

A machine learning-based framework for anomaly detection

RS Ball



orcid.org/0000-0001-7076-6157

Thesis accepted in fulfilment of the requirements for the degree
*Doctor of Philosophy in Computer and Information Sciences
with Computer Science and Information Systems* at the North-
West University

Promoter: Prof HA Krüger

Co-promoter: Prof L Drevin

Graduation May 2023

35484446

Acknowledgements

Firstly, I would like to thank my Promoter, Prof Hennie Kruger, as well as my Co-promoter, Prof Lynette Drevin, for their tireless efforts and support of this study. I am extremely grateful for Prof Kruger's willingness to revise countless versions of this work. This study would not have been possible without his endless guidance and support.

I would also like to thank my family for all their patience with and support of my studies, especially my wife, Ricki-lee, my son, Carter, and my daughter, Mila. Thank you for allowing me to undertake this commitment. I love you all very much.

Lastly, I'd like to thank my parents; my father, Richard, and my mother, Lynette, for instilling in me the importance of a good education. Thank you for all your sacrifices and support over the years.

Abstract

With the expansion of modern technology and the increased adoption of digital payment methods, fraudsters are becoming more sophisticated in their approach. Organisations attempting to mitigate fraudulent attacks can leverage their data to drive systems dedicated to detecting anomalous behaviour. Unfortunately, the techniques used to detect anomalies are often fragmented, focussing on solving a very specific detection problem. In this study, a machine learning-based framework for anomaly detection is proposed that aims to combine multiple machine learning approaches to detect anomalies in transactional systems.

The proposed framework includes a unified approach for combining aspects of the anomaly detection process into a singular pipeline. The approach begins with a neural architecture search, where multiple candidate autoencoder architectures are randomly simulated to determine an optimal architecture configuration. An optimal architecture is determined by evaluating each model with a proposed thresholding algorithm, which calculates the optimal threshold, using a balanced score. The output of the optimal architecture is then transformed from raw anomaly scores into a more manageable score in the range $[0,1]$ through the application of Gaussian scaling. This approach is validated on a public data set for illustration purposes, followed by the application of the approach to a real-world transactional data set.

Social network analysis is then introduced to the study, by taking network data and calculating network metrics to generate a feature set for augmenting the initial real-world transactional features. These network metrics capture the behaviours that link users together within a transactional system. Computing the Shapley values of the autoencoder output returns the feature contribution of the network metrics to determine their impact on the model's ability to detect anomalies. The combined network metrics and transactional feature set are then used to train a self-organising map to surface clusters of anomalous activity not detectable through the autoencoder, as well as to provide an additional approach to feature contribution through the exploration of the anomalous cluster weights.

Finally, the various modelling approaches are combined into a proposed anomaly detection framework. The framework includes aspects of visual analysis and what-if analysis, which provides practitioners with an interface to analyse the outputs of the various machine learning techniques and to perform sensitivity analysis by inputting various classification costs and conducting threshold changes. The final proposed framework was implemented in a real-world setting to illustrate the practical applicability. The obtained results of the implementation successfully achieved the main objective, which was to improve the detection of anomalies in a transactional setting.

Keywords

Anomaly detection, fraud detection, autoencoders, neural architecture search, social network analysis, self-organising maps, Shapley values

Contents

Acknowledgments	i
Abstract	ii
List of Figures	ix
List of Tables	xii
List of Algorithms	xiv
List of Acronyms	xv
1 Introduction and problem contextualisation	1
1.1 Chapter introduction	1
1.2 Research topics	1
1.2.1 Anomaly detection	2
1.2.2 Fraud detection	3
1.2.3 Social network analysis	4
1.2.4 Self-organising maps	4
1.2.5 Anomaly detection frameworks	4
1.3 Problem statement and research question	5
1.4 Research aims and objectives	6
1.5 Data collection	7
1.6 Computing environment	8
1.7 Research methodology	8
1.8 Ethical considerations	8
1.9 Thesis layout and summary of contributions	9
1.9.1 Chapter 1: Introduction and problem contextualisation	9
1.9.2 Part I: Literature review	10
1.9.3 Part II: Development and illustrative applications	10

1.9.4	Part III: Model extension and real-world validation	10
1.9.5	Part IV: Final framework and conclusion	11
1.9.6	Summary of the contributions of the study	11
1.10	Chapter conclusion	12

I Literature review 13

2 Literature review: Fraud and anomaly detection 14

2.1	Chapter introduction	14
2.2	Introduction to credit card fraud	15
2.3	Fraud detection systems	18
2.3.1	System design	18
2.3.2	System architecture	18
2.3.3	Information feedback loop	20
2.3.4	Relevant features	20
2.3.5	Performance metrics for general statistical classification	21
2.3.6	Performance metrics for fraud detection systems	24
2.4	Supervised vs unsupervised learning	28
2.5	Algorithmic approaches to fraud detection	30
2.5.1	Supervised learning methods for fraud detection	31
2.5.2	Unsupervised learning methods for fraud detection	33
2.6	Framing the fraud detection problem	34
2.7	Trade-off between accuracy and interpretability	35
2.8	Class imbalance	36
2.9	Concept drift	37
2.10	Streaming data	39
2.11	Autoencoders	39
2.11.1	Undercomplete autoencoders	42
2.11.2	Regularised autoencoders	42
2.11.3	Variational autoencoders	47
2.11.4	Network architecture	47
2.11.5	Applications of autoencoders	48
2.11.6	Related works on autoencoders for anomaly detection	49
2.12	Chapter conclusion	56

3 Literature review: Self-organising maps and social network analysis 57

3.1	Chapter introduction	57
3.2	Self-organising maps	58
3.2.1	Introducing self-organising maps	58
3.2.2	Building self-organising maps	59

3.2.3	Training self-organising maps	59
3.2.4	Evaluating self-organising maps	61
3.2.5	Interpreting self-organising maps	63
3.2.6	Related works on self-organising maps	67
3.3	Social network analysis	73
3.3.1	Introducing social network analysis	73
3.3.2	Motivation for network metrics in anomaly detection	75
3.3.3	Centrality	77
3.3.4	Clique	84
3.3.5	Clustering	86
3.3.6	Link Analysis	88
3.3.7	Assortativity	89
3.3.8	Alternative approaches to determining network metrics	90
3.3.9	Related works on social network analysis	90
3.4	Chapter conclusion	98

II Development and illustrative applications 99

4 Autoencoders for anomaly detection in a transaction-based system 100

4.1	Chapter introduction	100
4.2	Description of illustration data	102
4.2.1	Latent space representation using t-SNE	102
4.2.2	Sampling approach	104
4.3	Neural architecture search	106
4.4	Threshold optimisation	108
4.4.1	Model loss	113
4.4.2	Latent feature representation	113
4.5	Gaussian scaling	115
4.6	Performance evaluation of optimal architecture	117
4.7	Results and discussion	118
4.8	Extension of proposed approach	119
4.9	Chapter conclusion	119

III Model extension and real-world validation 121

5 Practical real-world application of autoencoders for transactional anomaly detection 122

5.1	Chapter introduction	122
5.2	Description of the industry data	123

5.2.1	Latent space representation using t-SNE	126
5.2.2	Sampling approach	128
5.3	Neural architecture search using industry data	130
5.4	Threshold optimisation using industry data	133
5.4.1	Latent feature representation	136
5.5	Applying Gaussian scaling on the industry data	137
5.6	Evaluating performance of optimal architecture	139
5.6.1	Performance metrics	139
5.6.2	Shapley values for feature contribution	141
5.7	Results and discussion	145
5.8	Chapter conclusion	146
6	Social network analysis extension of the unified approach for anomaly detection	148
6.1	Chapter introduction	148
6.2	Description of the network data	149
6.2.1	Introducing the network metrics	150
6.2.2	Calculating the network metrics	151
6.3	Unified approach including network metrics	163
6.3.1	Neural architecture search including network features	163
6.3.2	Threshold optimisation using industry data with network features	166
6.3.3	Applying Gaussian scaling on the industry data with network metrics scores	168
6.4	Evaluating performance of optimal architectures	170
6.4.1	Performance metrics	170
6.4.2	Shapley values for feature contribution	171
6.5	Results and discussion	173
6.5.1	Practical considerations	175
6.5.2	Recommendations	176
6.6	Chapter conclusion	178
7	Self-organising maps for anomaly detection	179
7.1	Chapter introduction	179
7.2	Self-organising map model building	180
7.3	Self-organising map model training	180
7.4	Self-organising map model evaluation	182
7.4.1	Node activation frequencies	182
7.4.2	Class distribution	184
7.4.3	Class proportion	185
7.4.4	U-matrix	186
7.4.5	Clustering maps	187

7.5	Interpreting features via component planes	189
7.6	Results and discussion	194
7.7	Chapter conclusion	196
IV	Final framework and conclusion	197
8	An anomaly detection framework	198
8.1	Chapter introduction	198
8.2	The proposed framework	199
8.3	The social network analysis component	201
8.4	The anomaly detection component	203
8.5	The self-organising map component	204
8.6	The visual analysis component	205
8.7	The what-if component	206
8.8	Visualisation of output results	208
8.9	Chapter conclusion	214
9	Conclusions and recommendations	215
9.1	Chapter introduction	215
9.2	Synopsis of the study	215
9.2.1	Introduction to the study	216
9.2.2	Part I: Literature review	217
9.2.3	Part II: Development and illustrative applications	218
9.2.4	Part III: Model extension and real-world validation	219
9.2.5	Part IV: Final framework and conclusion	220
9.3	Contributions of the study	221
9.4	Limitations and directions for future research	225
9.5	Chapter conclusion	226
	References	227
	Appendix A Summary of published articles	250
	Appendix B Research output 1	251
	Appendix C Research output 2	264
	Appendix D Research output 3	266
	Appendix E Research output 4	332
	Appendix F Language editing declaration	334

List of Figures

2.1	Supervised learning	29
2.2	Unsupervised learning	29
2.3	Example of an autoencoder	41
3.1	Example of a clustering map (Vettigli, 2018)	64
3.2	Example of a u-matrix (Vettigli, 2018)	65
3.3	Examples of component planes (Vettigli, 2018)	66
3.4	An example of a simple undirected graph	74
3.5	An example of a simple directed graph	74
3.6	An example of a weighted graph	75
3.7	Example of an adjacency matrix	78
3.8	Simple network example of degree centrality (Lizardo, 2021)	78
3.9	Complex network example of degree centrality (Rocchini, 2006)	79
3.10	Complex network example of eigenvector centrality (Rocchini, 2006)	80
3.11	Simple network example of closeness centrality (Lizardo, 2021)	81
3.12	Complex network example of closeness centrality (Rocchini, 2006)	81
3.13	Simple network example of betweenness centrality (Lizardo, 2021)	82
3.14	Complex network example of betweenness centrality (Rocchini, 2006)	83
3.15	Complex network example of harmonic centrality (Rocchini, 2006)	84
3.16	Example of a clique (Barabasi, 2016)	85
3.17	Example of the number of cliques	86
3.18	Example of clustering coefficient (Phuong <i>et al.</i> , 2017)	87
3.19	Simple network example of PageRank centrality	89
4.1	Visualising latent space of credit card data using t-SNE	104
4.2	Optimising threshold selection using the balanced score	110
4.3	Reconstruction errors (MSE)	110
4.4	Simulating 50 architectures with NAS	111
4.5	Illustrating model loss	113
4.6	Visualising latent representation in two dimensions	114

4.7	Anomaly scores with linear scaling	115
4.8	Anomaly scores with Gaussian scaling	116
5.1	Visualising latent space using t-SNE on industry data	127
5.2	Sampling process	129
5.3	Simulating 100 architectures with NAS using industry data	130
5.4	Optimising threshold selection using the balanced score	134
5.5	Reconstruction errors (MSE) using industry validation data	135
5.6	Reconstruction errors (MSE) using industry test data	136
5.7	Visualising latent representation in two dimensions	137
5.8	Anomaly scores with linear scaling for industry data set	138
5.9	Anomaly scores with Gaussian scaling for industry data set	138
5.10	Shapley values for industry data	144
6.1	Small cluster with predominantly normal nodes	154
6.2	Small cluster with predominantly locked nodes	155
6.3	Small cluster with a mix of anomalous, locked and normal nodes	156
6.4	Largest cluster in the overall network. The red nodes indicate confirmed instances of fraud, while the orange and green nodes indicate locked accounts and non-fraudulent accounts respectively.	157
6.5	The entire network. The largest cluster in the network is found in the top left corner, with the clusters becoming smaller towards the bottom of the figure.	158
6.6	Visualising latent space using t-SNE on industry data with network features	162
6.7	Simulating 100 architectures with NAS including network features	163
6.8	Visual depiction of the optimal architecture	165
6.9	Optimising industry data with network features threshold selection using the balanced score	166
6.10	Reconstruction errors (MSE) using industry validation data with network features	167
6.11	Reconstruction errors (MSE) using industry test data with network features	167
6.12	Anomaly scores with linear scaling for industry data set with network metrics	168
6.13	Anomaly scores with Gaussian scaling for industry data set with network metrics	169
6.14	Shapley values for industry data with network metrics	171
7.1	Training the SOM	181
7.2	The node activation frequencies of the SOM	183

7.3	The distribution of the classes across the SOM	184
7.4	The proportion of the classes across the SOM	185
7.5	The u-matrix	186
7.6	The clustering map of the most significant features per node	188
7.7	Component planes for features 1 to 25 (transactional features)	191
7.8	Component planes for features 26 to 47 (transactional features)	192
7.9	Component planes for features 48 to 61 (network metrics features)	193
8.1	High-level overview of the anomaly detection framework	200
8.2	Overview of the SNA component within the anomaly detection framework	202
8.3	Overview of the AD component within the anomaly detection framework	203
8.4	Overview of the SOM component within the anomaly detection framework	205
8.5	Overview of the VA component within the anomaly detection framework	206
8.6	Overview of the WI component within the anomaly detection framework	207
8.7	Example of the toggle at the default threshold	208
8.8	Example of the toggle at the updated threshold	209
8.9	Threshold visualisation with baseline threshold of 5 (left subplot) and adjusted threshold of 4.35 (right subplot)	209
8.10	Performance metrics for default threshold	210
8.11	Performance metrics for updated threshold	211
8.12	A sample of the anomalies detected	211
8.13	Displaying top n features for a sample of the anomalies detected	212
8.14	Calculating the total cost of fraud	213

List of Tables

1.1	Layout of thesis	9
2.1	Example of a confusion matrix for binary classification	22
2.2	Recent studies on the industrial and maintenance application of autoencoders	50
2.3	Recent studies on the fraud detection application of autoencoders	51
2.4	Recent studies on the security application of autoencoders	52
2.5	Recent studies on building and IoT applications of autoencoders	53
2.6	Recent studies on the process application of autoencoders	54
2.7	Recent studies on the computing application of autoencoders	55
3.1	Examples of recent studies on the medical application of SOMs	68
3.2	Examples of recent studies on the industrial application of SOMs	69
3.3	Examples of recent studies on the geographical application of SOMs	70
3.4	Examples of recent studies on risk and anomaly detection application of SOMs	71
3.5	Examples of recent studies on the computing application of SOMs	72
3.6	Examples of recent studies on the healthcare application of SNA	91
3.7	Examples of recent studies on the fraud detection application of SNA	92
3.8	Examples of recent studies on the geographical application of SNA	93
3.9	Examples of recent studies on the finance and organisational application of SNA	94
3.10	Examples of recent survey studies of SNA	95
3.11	Examples of recent studies on the networking application of SNA	96
3.12	Examples of recent studies on the computing application of SNA	97
4.1	Details of top 20 architectures	112
4.2	Confusion matrix of optimal architecture	117
5.1	First 10 samples of industry data	125
5.2	Details of top 20 architectures using industry data	131

5.3	Confusion matrix of optimal architecture with industry data	139
6.1	Description of the original network metrics	150
6.2	Source network data for first five edges	151
6.3	Description of the manual network features	152
6.4	Description of the final network features	159
6.5	Raw network metrics for first five nodes	160
6.6	First 25 samples of industry data network metrics features	161
6.7	Details of top 20 architectures using industry data with network metrics	164
6.8	Confusion matrix of optimal architecture including network features .	170
7.1	An example of the node weightings for feature $v5$	194
8.1	Cost matrix	213

List of Algorithms

3.1	Generalised approach to training a SOM (Ponmalai and Kamath, 2019)	60
4.1	Simple version of t-Distributed Stochastic Neighbor Embedding (Maaten and Hinton, 2008)	103
4.2	Neural Architecture Search algorithm	107
4.3	Anomaly score threshold optimisation algorithm	109
4.4	Gaussian scaling of anomaly score algorithm	116
5.1	Autoencoder based anomaly detection algorithm (An and Cho, 2015)	133
5.2	Calculate Shapley values for the top M features (Antwarg <i>et al.</i> , 2021)	142
5.3	Prediction of the i^{th} feature (Antwarg <i>et al.</i> , 2021)	143
5.4	Divide Shapley values into those contributing and offsetting an anomaly (Antwarg <i>et al.</i> , 2021)	143
6.1	Dijkstra’s algorithm (Dijkstra, 1959)	153

List of Acronyms

AD	Anomaly Detection
AP	Average Precision
AUC	Area Under the Curve
AUPRC	Area Under the Precision-Recall Curve
BMU	Best Matching Unit
CSV	comma-separated values
E_q	Quantization error
E_t	Topographical error
FP	False positive
FN	False negative
KL	Kullback-Leibler
MAE	Mean Absolute Error
MCC	Matthews Correlation Coefficient
MSE	Mean Square Error
NAS	Neural Architecture Search
PCA	Principal Component Analysis
ROC	Receiver Operating Characteristic
SHAP	SHapley Additive exPlanations
SNA	Social Network Analysis
SOM	Self-Organising Map
t-SNE	t-Distributed Stochastic Neighbor Embedding
TN	True negative
TP	True positive
VA	Visual Analysis
VAE	Variational Autoencoder
WI	What-if

Chapter 1

Introduction and problem contextualisation

1.1 Chapter introduction

Chapter 1 serves as an introduction to the study. Its contents will include a problem statement to contextualise the research, as well as the research aim and objectives for the study. The data collection, methods of investigation, and ethical considerations are also discussed. The following sections are presented in Chapter 1:

- Research topics;
- Problem statement and research question;
- Research aims and objectives;
- Data collection;
- Computing environment;
- Research methodology;
- Ethical considerations; and
- Thesis layout and summary of contributions.

1.2 Research topics

The aim of the thesis is to study the ever-prevalent issue of anomaly detection in a transaction-based system. Anomalous actors are becoming increasingly sophisticated

both in methodology and technical ability (Robertson, 2018; Wei *et al.*, 2013). Fortunately, the companies impacted by anomalous behaviour are also generating more data on and insights into potential anomalous cases than ever before.

Cynthia and George (2021) suggest that detecting anomalies in a transaction-based system is often solved, using either a supervised or unsupervised learning approach. Supervised approaches are used more frequently, but do not fare well in cases where labelled instances are few or non-existent. Instead, the use of unsupervised learning methods to both detect changes in behaviour and uncover unusual transactions is proposed (Gomes *et al.*, 2021).

An anomaly detection system can be structured in several ways (Gogoi *et al.*, 2010; Yamanaka *et al.*, 2019). The first and most popular approach would be the generation of a probability score of a user or transaction being an anomaly. This has traditionally been achieved by using supervised classification techniques (Pang *et al.*, 2021). A second approach which is less popular and therefore warrants further attention is the generation of interpretable clusters which issue the user with a high-level category (high-risk, low-risk, *etc.*), as well as an indication of what features contributed to the clustering decision (Pang *et al.*, 2016). Thirdly, it has been proposed that network modelling techniques can be used to inform a user whether an action is likely to be an anomaly, based on its relationship with other actions, events or parties within a broader network (Anand *et al.*, 2017; Kaur and Singh, 2016). It has also been proposed that the measures that these network modelling techniques generate could be used to inform a machine learning algorithm by being added to the feature set (Bonilla, 2018). The work of Bonilla (2018), Fire and Puzis (2016), and Nurek and Michalski (2020) confirms these issues on anomaly detection and can be consulted for further in-depth analysis.

To address the problem of anomaly detection efficiently, the study includes different research topics that each form an objective on their own. These areas of interest form the focus of the research topics and include anomaly detection, fraud detection, social network analysis, self-organising maps, and anomaly detection frameworks.

1.2.1 Anomaly detection

Anomaly detection is the process of determining the patterns in a dataset whose behaviour is not normal or expected (Agrawal and Agrawal, 2015). The abnormal behaviours are also termed anomalies or outliers. The study of anomaly detection is important because anomalies indicate significant but rare events, prompting critical action to be taken in a variety of application areas (Ahmed *et al.*, 2016). The financial implications of anomalous transactions are also significant, as they tend to

indicate fraudulent behaviour, which in turn leads to financial losses for individuals and organisations alike (Huang *et al.*, 2018).

The field of anomaly detection relates to dealing with uniqueness in data, and the challenges associated with it are therefore distinctive. Firstly, there are many unknowns involved in anomaly detection, namely unknowns related to the emerging behaviours and distributions of the data. They remain unknown entities until they occur in practice and/or a new type of fraud is detected (Pang *et al.*, 2021). Secondly, there are heterogeneous anomaly classes (Pang, 2019). Anomalies are irregular, and so one class of anomalies may differ significantly from another class of anomalies. Finally, anomalies are rare and therefore few in number (Chen *et al.*, 2016). Normal instances of the data, therefore, tend to significantly outnumber the anomalies in the data. This phenomenon is known as the class imbalance problem (Feng *et al.*, 2021), which makes it difficult to collect a large number of labelled anomalies to be used for modelling purposes.

Traditional anomaly detection approaches, especially unsupervised learning methods, tend to report a high number of false positives (Campos *et al.*, 2016). While these traditional approaches tend to fare well in detecting anomalies in a low-dimensional space, they fail to excel in high-dimensional feature sets (Zimek *et al.*, 2012). Neural networks, and autoencoders, in particular, have been proposed as an essential anomaly detection method for overcoming some of the shortcomings of traditional machine learning approaches (Erfani *et al.*, 2016; Xu *et al.*, 2015; Yu *et al.*, 2018). Autoencoders can assist in high-dimensional contexts, as they aim to learn a low-dimensional feature representation of a high-dimensional space (Jiang *et al.*, 2014). Explainability is another concern in traditional anomaly detection, as black-box models tend to lack the ability to explain anomaly detection decisions. Pang *et al.*, (2021) claim that anomaly explanations are still a largely unsolved problem, although some attempts have been made to explain the decisions made by an autoencoder using Shapley values (Antwarg *et al.*, 2021).

1.2.2 Fraud detection

A common problem within the anomaly detection domain is that of fraud detection. According to Chalapathy and Chawla (2019), fraud relates to the deliberate act of deception to access valuable resources, while fraud detection is the process whereby methods are employed to detect and prevent fraudulent acts from occurring. Credit card fraud is an important problem in the financial services industry, where stolen card details are used as a fraudulent source of funds in a transaction (Varmedja *et al.*, 2019).

The types of fraud that occur in the industry are numerous, with little consistency in their patterns, which ensures that fraud detection approaches need to adapt

continuously. The fraud detection subtype is the anomaly detection application area of choice for this study, with a variety of anomaly detection techniques being applied to detect suspicious patterns within the data.

1.2.3 Social network analysis

Social network analysis forms an important part of the research topics presented in this study, where some of the network approaches are used to enhance the anomaly detection modelling process. Fraudulent behaviours tend to differ from normal behaviours within a financial system, where fraudulent actors often act as a collective, fraudulent syndicate (Bodaghi and Teimourpour, 2018). The data relating to the transactions between fraudsters are therefore inherently linked. Social network analysis excels at capturing relationships between data in graphical form and is a vital technique for detecting anomalies in linked data.

The use of social network analysis in this study is twofold. The methodology will be used to generate several network metrics to highlight the proximity of any record to the confirmed anomalies in the data. The network metrics (for example, average degree, closeness centrality, betweenness centrality and PageRank) will then be used to help construct a feature set for the above autoencoder approach to assess whether they are significant in detecting anomalies (Bonilla, 2018; Fire and Puzis, 2016; Nurek and Michalski, 2020).

1.2.4 Self-organising maps

In practice, while an anomaly score might be effective as a stand-alone prevention method against anomalous behaviour, analysts are generally involved in the actual decision-making process and are likely to benefit from visual indicators of changing and unknown anomalous behaviour (Leite *et al.*, 2018). This presents an opportunity to incorporate self-organising maps, an unsupervised neural network technique (Chen *et al.*, 2021) that assists in identifying emergent behaviour through the formation of clusters.

1.2.5 Anomaly detection frameworks

Using and interpreting all the aforementioned techniques in isolation can be confusing for a prospective decision maker. For a decision to be reached in the shortest time possible, a well-defined and appropriately designed anomaly detection framework should be used to consolidate results. Whereas most anomaly detection systems focus on generating only an anomaly score (Lavin and Ahmad, 2015), the proposed framework developed in this research study will provide insights into the larger anomaly

detection problem, which are complex problems that need to be understood at the management level (Riveiro *et al.*, 2018).

1.3 Problem statement and research question

Reports suggest that the worldwide total credit card fraud losses realised were \$28.65 billion in 2020 (Nilson, 2020). Rampant rates of fraud in the industry are confirmed by the professional consulting and services network company, PricewaterhouseCoopers (2022), which reports that more than half of the global organisations surveyed suffered from fraud in the last 24 months. In the case of credit card fraud, losses impact the banks who issue the cards, the merchants who accept them for payment, and the consumers who make use of the cards. The problem of fraud is a growing concern for financial services organisations across the world that are incurring greater costs due to fraudulent activity each year (Hakak *et al.*, 2020). The introduction of remote working due to the COVID-19 pandemic has also contributed to the increase in the rate of transactional fraud, as more business is conducted online than ever before (Wan *et al.*, 2022).

Anomaly detection is important because it is a proven technique for detecting anomalous patterns in transactional data (Pourhabibi *et al.*, 2020). Transactional systems are the financial mechanisms that facilitate transactions through the use of credit cards and are therefore susceptible to fraud. Numerous studies have been conducted on fraud detection, although the area is constantly evolving as fraudulent practices become more sophisticated (Chen and Lai, 2021; Engels *et al.*, 2019; Sadgali *et al.*, 2020). Therefore, there are areas of improvement for fraud detection within the literature. These areas of opportunity involve extending existing machine learning techniques to detect fraud in numerous ways (Cacheda *et al.*, 2019; Lopez *et al.*, 2019).

Traditional transactional features can be combined with non-traditional sources of data, such as social network analysis metrics, to produce features that capture information about transactions from a variety of perspectives. The features can be used to train an autoencoder model to detect anomalies in an unsupervised manner. Self-organising maps can also be employed to study the nature of fraudulent signals which are expected to change over time. Lastly, the approaches can be combined into a formal framework to detect and prevent fraudulent transactions from occurring.

Based on the opportunities highlighted above, the problem investigated in this study deals with particular machine learning techniques and extensions to provide new insights and make a contribution to the field of anomaly and fraud detection. The primary research question for this study is therefore formulated as follows: *How can*

an anomaly detection framework, comprised of multiple machine learning techniques, improve anomaly detection in a transaction-based system?

1.4 Research aims and objectives

The main objective of this study, which is based on the formulated research question, is to *design, develop* and *propose* a machine learning-based anomaly detection framework that combines multiple approaches to inform whether a transactional behaviour is likely to be anomalous.

To achieve the main objective, the following sub-objectives are formulated:

1. *Conduct* a literature review on the main aspects related to the study, namely anomaly and fraud detection, autoencoder neural networks, social network analysis, and self-organising maps;
2. *Establish* an appropriate autoencoder architecture to detect anomalous behaviour;
3. *Extend* the formulated autoencoder model by *identifying* and *incorporating* applicable social network analysis metrics into the proposed machine learning-based anomaly detection framework;
4. *Develop* a self-organising map functionality to form part of the proposed machine learning-based anomaly detection framework. More specifically, the new functionality should assist with *categorising* behaviours into clusters and provide an additional intuitive understanding of which anomaly-related variables contributed to the decision-making process;
5. *Demonstrate* the adequacy and validity of the developed techniques, using existing example data sets;
6. *Identify* and *implement* a real-world application data set to demonstrate and validate the performance of the proposed machine learning-based anomaly detection framework and its underlying techniques;
7. *Develop* and *propose* a final machine learning-based anomaly detection framework that provides a user-friendly interface that is capable of visually presenting results, as well as answering appropriate what-if questions; and
8. *Recommend* applicable future research opportunities, based on the work presented in this thesis.

1.5 Data collection

Data related to anomaly detection is notoriously difficult to obtain (Phua *et al.*, 2010). Organisations are traditionally conservative when it comes to publishing data related to cases of in-house anomalies. This is because exposing their internal anomalies might pose a reputational risk, or the data may present an opportunity for prospective bad actors to capitalise on potential perceived gaps in the system.

The study will be conducted using a combination of data from an industrial partner (a real-world data set), as well as a publicly available anomaly data set that has been cited in the literature (Dal Pozzolo, 2015). The publicly available data set is used to illustrate the efficacy of the proposed solutions, whereas the private data set, sourced from an industrial partner, is used to prove that the solution makes a positive, real-world contribution.

The public data set is sourced from Kaggle, a website dedicated to hosting machine learning competitions (Kaggle, 2017). The data set was collected by Worldline S.A.¹, a payments provider, and has already been anonymised. Therefore, there is no way in which to infer identities or additional user information from within the data. The same approach is taken when preparing the private data from the industrial partner.

The context of the real-world problem is important, and a description of how the business partner operates is provided. Explaining the context will highlight how the data is generated and will detail the special considerations that the machine learning models or the proposed anomaly detection framework need to make. The industrial partner from whom the private data set is sourced is an active and major participant in the financial services industry, providing financial services to over 10 million customers. The business ecosystem involves buying, selling, making and receiving deposits, as well as the use of credit cards to fund user accounts, with over 50 000 unique payments and 400 000 unique trading transactions facilitated by the organisation each day.

The business aims to combat finance-related crimes by effectively understanding what behaviour, as measured through data, might indicate instances of anomalies in the form of fraud and phishing, *etc.* Typically, the data that contributes to these behaviours are aggregated and sorted into features that can be used to develop machine learning models. The output of any model in this scenario would be a probability score which represents the likelihood that the corresponding user or transaction is anomalous. The scores are then fed into internal rule-based systems that adjust the customer’s experience, based on the static rules that are created, using these scores.

¹<https://worldline.com/>

This endeavour intends to give good customers a streamlined experience and to limit the potentially negative impact that bad customers can inflict on the company and/or on other customers.

1.6 Computing environment

The Jupyter notebook infrastructure was used for performing all the computational experiments throughout this thesis. All instances were run on a Dell Inspiron 15 7000 gaming laptop computer with an Nvidia GeForce 1060 GPU and CUDA configured. Python 3, as well as the *Keras* framework, were used to build the various autoencoder models in Chapters 4, 5, and 6. The social network analysis metrics generated in Chapter 6 were prepared by leveraging the *NetworkX* library, while the self-organising map work in Chapter 7 was conducted using *MiniSOM*. Lastly, *Streamlit* was used to build the final proposed anomaly detection framework in Chapter 8.

1.7 Research methodology

This current chapter serves as an introduction to the study. The remainder of the research project will be executed and documented in four consecutive main parts. In Part I of the thesis, the literature study is presented. It consists of two chapters (Chapters 2 and 3) and sub-objective 1 is addressed. In the next part, Part II, sub-objectives 2 and 5 are addressed in Chapter 4. In this part, the development and formulation of the autoencoder model are detailed and an illustrative application to validate the model is presented. In Part III, the model extensions (the incorporation of social network analysis metrics and the use of self-organising map analyses) are described and a real-world application validation is presented. Part III is made up of three chapters (Chapters 5, 6, and 7) in which sub-objectives 3, 4, and 6 are addressed. In the final part, Part IV, sub-objectives 7 and 8 are addressed. The proposed anomaly detection framework (Chapter 8) is presented and described. The study is also concluded in Part IV (Chapter 9) by showing how all the objectives were achieved, giving an appraisal of the contributions, and making recommendations on future research opportunities.

1.8 Ethical considerations

The research proposal for this thesis was presented to the ethics committee of the Faculty of Natural and Agricultural Sciences, North-West University for ethical clearance. The study was approved as a no-risk study and ethics number NWU-01456-20-A9 was issued on 1 February 2020. A non-disclosure agreement has been entered into with the industrial partner from the financial services industry to enforce the

full anonymity of the industrial partner’s user base. The illustrative data set used in this study is available online² and contains anonymised transactional data only. There are no other human participants involved in the study.

1.9 Thesis layout and summary of contributions

This thesis comprises an introductory chapter (Chapter 1) and a further eight chapters, grouped as four distinct parts. The four parts are labelled Literature review, Development and illustrative applications, Model extension and real-world validation, and Final framework and conclusion. The layout is summarised in Table 1.1.

Table 1.1: Layout of thesis

Thesis layout		
Thesis section	Chapters	Description
Introduction	Chapter 1	Introduction and problem contextualisation
Part I	Chapter 2	Literature review: Fraud and anomaly detection
	Chapter 3	Literature review: Self-organising maps and social network analysis
Part II	Chapter 4	Autoencoders for anomaly detection in a transaction-based system
Part III	Chapter 5	Practical real-world application of autoencoders for transactional anomaly detection
	Chapter 6	Social network analysis extension of the unified approach for anomaly detection
	Chapter 7	Self-organising maps for anomaly detection
Part IV	Chapter 8	An anomaly detection framework
	Chapter 9	Conclusions and recommendations

The four main parts of the thesis, as well as their corresponding chapters are presented below.

1.9.1 Chapter 1: Introduction and problem contextualisation

Chapter 1 serves as an introduction to the study. Its contents include background information, a problem statement, research objectives, as well as the research methodology.

²<https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud>

1.9.2 Part I: Literature review

The literature review involves a thorough investigation of research related to autoencoders for anomaly detection, social network analysis, self-organising maps, and combining machine learning approaches in anomaly detection systems. More specifically, in *Chapter 2*, an investigation of research related to fraud and anomaly detection, including fraud detection systems, different machine learning approaches for fraud detection, and class imbalance problems is presented. The chapter is concluded with a discussion on the various types of neural network autoencoder, and how these are used for anomaly detection purposes.

Chapter 3 follows with a literature review on self-organising maps and social network analysis. The self-organising maps review covers topics related to building, training, evaluating, and interpreting self-organising maps, as well as related works from the literature. The social network analysis review motivates the use of network metrics in anomaly detection. Following this, various network metrics related to centrality, clique, clustering, link analysis and assortativity are presented. Chapter 3 ends with a review of works from the literature that are related to social network analysis.

1.9.3 Part II: Development and illustrative applications

In *Chapter 4*, a discussion is presented on the use of autoencoders for anomaly detection in a transaction-based system. This serves as a practical illustration of autoencoders, using a public data set sourced from *Kaggle*. A description of the data used in the chapter is initially presented, followed by the introduction of a unified approach for anomaly detection. The unified approach comprises a three-part process, involving a neural architecture search, threshold optimisation, and Gaussian scaling process. Chapter 4 is concluded with an evaluation of the results, as well as a proposal for an extension of the unified approach.

1.9.4 Part III: Model extension and real-world validation

In *Chapter 5*, a real-world dataset sourced from the industrial partner is used to validate the approaches presented in Chapter 4. The real-world case study is then followed, in *Chapter 6*, by a comprehensive description of how the original model can be extended through the inclusion of social network analysis metrics in the feature space. A second autoencoder model is trained on a combined feature set, including both the transactional and network features to enhance the prediction capability of the original model. In the final chapter in Part III (*Chapter 7*) the development of a self-organising map model that is utilised to provide additional insight into the contribution of the variables used in the modelling process is examined.

1.9.5 Part IV: Final framework and conclusion

The final part of the thesis, comprising Chapters 8 and 9, concludes the study. In **Chapter 8**, the final proposed machine learning-based anomaly detection framework is presented. The facets related to the design, development and implementation of the framework are discussed, and a computer-based representation is illustrated to the reader. In the final chapter, **Chapter 9**, there is a demonstration of how the set goals of the study have been achieved. An appraisal of the contributions, as well as recommendations for further and follow-up studies are provided.

1.9.6 Summary of the contributions of the study

A comprehensive appraisal of the contributions is given in Chapter 9. The primary contribution is based on the main objective of the study, which is the proposal of a new machine learning-based framework for anomaly detection. In addition to the achieved main objective, additional contributions are made by the study, as listed below.

- The literature review has produced a valuable framework, beyond that in which this study was grounded, serving as a useful resource and reference framework for future research related to the field of machine learning-based anomaly detection models and frameworks.
- The proposal of a novel neural architecture search methodology, which was empirically tested and illustrated, using an example data set. The methodology contributes to determining an optimal architecture for autoencoder models in a real-world setting
- Social network analysis extends the feature space of the initial anomaly detection model, by capturing the relationships between transactions within the system. The inclusion of the social network analysis metrics leads to improved prediction capability of the model.
- The introduction of self-organising map analysis provided additional techniques to analyse variables and detect different anomalous behaviours.
- Applying the proposed models and techniques in a real-world setting validated the approaches.
- The proposal to combine the above-mentioned work into a comprehensive machine learning-based framework for anomaly detection.
- Suitable recommendations of ideas for follow-up work and further research opportunities.

The contributions of the study are in line with the primary and sub-objectives formulated earlier and will be revisited in Chapter 9.

1.10 Chapter conclusion

In Chapter 1, the study was introduced and the research project was contextualised by presenting information, such as a problem statement, research question and objectives, as well as the research methodology. The thesis layout was presented, and the contributions of the study were highlighted.

Part I

Literature review

Chapter 2

Literature review: Fraud and anomaly detection

2.1 Chapter introduction

Chapter 2 is the first of two chapters in Part I of the study. The purpose of Part I is to provide a thorough investigation of research related to autoencoders for anomaly detection, social network analysis, and self-organising maps. The literature review on fraud and anomaly detection aims to provide the necessary background for the study and to contextualise the study in terms of other studies. Anomaly detection is typically applied in a variety of different transaction-based data systems. One major area of concern is the detection of anomalies within financial transactions in general. Financial data is used in the study, but with a focus on credit card transactions. The reasons for using credit card transactions are twofold; firstly, because detecting anomalies in credit card transactions is an important problem faced by many in the industry, and secondly, because access to credit card transactions data is available both online and via the industrial partner with whom this research is closely aligned. Several techniques can be used for this purpose. The study will focus on autoencoders for fraud and anomaly detection. The reasons why these techniques will be used are because they are relatively recent approaches, are popular from an industrial perspective, and are appropriate for anomaly detection.

The review starts with an introduction to anomalous credit card transactions and how these are typically managed from an operational perspective. Topics related to the types of financial anomaly, their cost to the organisation impacted by financial anomalies, and anomaly detection system architecture components are included. Following this, general machine learning concepts related to anomaly detection are then

discussed. These concepts include the definition of the anomaly detection problem from a fundamental machine learning perspective, what algorithmic approaches are applied in studies related to anomaly detection, and considerations around implementation. Finally, the focus in this chapter will be on studies related to autoencoders and their application to anomaly detection.

2.2 Introduction to credit card fraud

Fraud by definition is the obtaining of goods, services or money by unethical means (Delamaire *et al.*, 2009). It is becoming an ever increasingly popular means of financial crime. The topic of fraud detection is a notoriously difficult problem to solve, and the topic has therefore garnered a considerable amount of attention in the literature, as well as in the machine learning community at large (Chalapathy and Chawla, 2019). Fraud does not only occur in credit card usage, but it can also apply to several different financial products, such as loans and retail accounts. Those conducting fraudulent activities are becoming increasingly sophisticated in their approaches; therefore organisations need to employ robust methodologies to prevent fraud. This important distinction between prevention and detection is made by Anderson (2007), where organisations are prompted to take steps to prevent fraud from occurring in the first place, as well as to take effective and efficient action when fraud does slip through the first line of defence and an instance is confirmed.

As discussed by Anderson (2007) in the author’s classification of the types of fraud, credit card fraud falls under the general fraud umbrella that contains many other types of fraudulent activity, such as ATM fraud and card skimming. For this research, the focus will be on one specific case of fraudulent activity, namely credit card fraud. Credit card fraud is becoming more prevalent as more industries move online, involving easier methods for purchasing via a credit card, further perpetuating the issue. This is confirmed by the Office for National Statistics in the UK that states that digital fraud had risen by 23% between 2019 and 2020 alone (Stripe, 2020).

Types of credit card fraud

An important consideration to take into account is that within the realm of credit card fraud, there exist several types in terms of how and when the fraudulent activity took place. As explained by Delamaire *et al.*, (2009), there are four distinct types of fraud related to credit cards. The first proposed type is bankruptcy, in which a user makes use of his/her credit card while being insolvent. Theft or counterfeit fraud, which entails using a card that is not yours, is the second proposed type. Then there is application fraud, where a credit card is applied for, using false information. Finally, the fourth proposed type is behavioural fraud, where details of legitimate

cards have been obtained fraudulently. These usually occur in instances where a vendor assumes that the user has the card and therefore only the card details are necessary to transact, for example in an e-commerce environment.

Sadgali *et al.*, (2018) offer a contrasting view of credit card fraud types. Here the authors argue that there are two main types, namely application fraud and behavioural fraud, with behavioural fraud being broken down further. The proposed sub-categories of behavioural fraud include mail theft, stolen or lost cards, counterfeit card, and ‘cardholder not present’ fraud. Regardless of the breakdown, behavioural fraud is essentially used as an umbrella term to encapsulate all cases of card fraud where fraudulent activity takes place post-application. This is opposed to application fraud where the fraudulent activity technically took place during the application process, via the submission of false information.

In the context of an e-commerce transaction, it is clear that multiple types of fraud can occur, each with its own distinct attributes. For example, a fraudster could initiate an interaction with an e-commerce company where the card may have been obtained either via application or behavioural fraud. The fraud could also be considered from multiple perspectives too, namely the perspective of the issuing bank, the e-commerce vendor, and the customer affected.

This introduces the concept of account takeovers into the discussion. Khan *et al.*, (2013) describe account takeover as when a fraudster illegally obtains access and control over a legitimate account. Again, from the perspective of a bank, this would be the bank account itself, but in the context of an e-commerce company, this would entail a fraudster obtaining access to the e-commerce login account itself. The definitions can also become somewhat diluted as an account takeover could also eventually involve the use of a card obtained via theft or other means. This illustrates that quite often there are multiple parties involved, each with its own perspective and level of exposure to the fraud case.

Impact of fraud

As discovered above, credit card fraud usually impacts three distinct parties. This is confirmed by Bhatla *et al.*, (2003) who state that the impact is realised by the cardholder, the merchant, as well as the bank. The merchant is generally deemed to be the party affected the most by credit card fraud. In some cases, particularly in chargeback scenarios, the merchant absorbs the cost of the fraudulent transaction as he/she is in no position to challenge a chargeback dispute issued by a bank. These costs are not merely the value of the good or service purchased, but also include potential shipping costs, card association fees, merchant bank fees, administration costs, as well as loss of reputation and goodwill.

In other cases, the cost of credit card fraud can be realised by the banks involved. These also entail a combination of the direct cost of the fraud itself, as well as certain administration and manpower costs. The onus is therefore on both merchants and banks to apply processes and technological enhancements to counter the impact of fraud, particularly in the face of ever exceedingly complex methods of conducting fraud. The costs of developing these solutions are significant, but as suggested by Bhatla *et al.*, (2003), both banks and merchants are creating innovative solutions that are reducing the number of fraud instances and lowering chargeback rates.

Balancing the total cost of fraud

The performance of any fraud prevention mechanism can be measured by considering the total cost of fraud (Bahnsen *et al.*, 2013). The total cost is essentially the realised cost of all incurred fraud instances, as well as the cost of the systems employed to detect and manage the fraud. These systems could include software or human screening and therefore involve a monetary and time element, both of which need to be quantified to determine the total cost of fraud. To minimise the total cost, organisations are required to find the optimal balance between the cost of the fraud and the cost of the fraud prevention (Beneish and Vorst, 2021). If one does not maintain an optimal balance, two extremes could be realised. The first extreme would be a scenario where an organisation does not employ enough resources to detect fraud. Here the cost of fraud prevention would be low, but the instances of fraud would be numerous and outweigh the savings made by not implementing strict fraud detection principles, thereby increasing the total cost of fraud for the organisation. The converse situation is one in which fraud is reduced by having overly strict fraud detection processes and reviews, but not enough to cover the costs of having excessively strict processes in place.

Human in the loop

One particularly important factor in balancing the total cost of fraud is the review process. Reviewing is essentially when an experienced fraud analyst verifies whether a potential fraud case is fraudulent or not (Veeramachaneni *et al.*, 2016). The human in the loop acts as the source of truth in the detection process. While fraud detection systems help guide the human in the loop toward knowledge, the fraud detection system cannot conclude with absolute certainty whether the transaction in question was fraudulent or not. The human analyst is therefore still an important part of the screening process. A secondary benefit of having a human in the loop is that he/she is effectively labelling the data for the organisation. Labelling in this context is essentially flagging records as fraudulent or not, which can be used further down the line for classification and other machine learning approaches.

2.3 Fraud detection systems

As discussed in the previous section, credit card fraud detection is a complex problem, requiring a complex solution in the form of a system that identifies patterns in transactional data, sends suspicious patterns and data points to an analyst who can verify the outcome of the pattern, and then for these verifications to be fed back into the system for it to update its knowledge. Finally, the system needs to cater for the cost of misclassification and how that impacts the total cost of fraud. In this section, the properties of such a system are discussed.

2.3.1 System design

In practice, a functioning fraud detection system typically consists of multiple layers positioned at different stages of the credit card purchasing flow (Wilhelm, 2004). These are divided into real-time, close to real-time, and offline layers. Dal Pozzolo (2015) argues that the distinct stages where a layer exists include when the transaction is attempted by the user (terminal), when the transaction request is sent (transaction blocking), when the transaction seeks authorisation (scoring rules and data-driven model), and finally when a fraud investigator is alerted of potentially suspicious activity. Only the final layer in this system involves human intervention, while the rest are fully automated. The layered approach to fraud detection and prevention is confirmed by Litan (2012) who settled on a five-layer approach, including endpoint-centric, navigation-centric, channel-centric, cross-channel-centric, and entity link-related layers. The layers proposed by Dal Pozzolo (2015) will be discussed in more detail in the next section.

2.3.2 System architecture

According to Dal Pozzolo (2015), the first layer of the typical fraud detection system is referred to as the *terminal*. Here card-specific checks take place in real-time, and if any of the checks fail, the transaction attempt is declined (Van Vlasselaer *et al.*, 2015). Examples of terminal layer checks include whether the incorrect pin code was used, the number of attempts on the card and whether any spending limit was reached.

The second layer can be referred to as the *transaction blocking* layer. In this layer, transactions are blocked before being authorised while operating in real-time and checking the transactions against very specific blocking rules designed by fraud investigators. An example of a transaction blocking rule might be something regarding two transactions (for the same customer or card) attempted from two very distinct Internet Protocol (IP) address locations within a very short space of time. In reality, the rules are not public knowledge, as potential fraudsters might be able

to circumvent this layer if they were to know what rules were in place. The second layer is the last in the real-time analysis phase, and transactions that pass through these blocking rules are then sent for authorisation. In the case where a transaction passes through the blocking rules, aggregate data is collected and used to inform decisions in the next layer. The data acts as additional features for the next layers, *i.e.*, scoring rules and a data-driven model.

Scoring rules put in place by experienced fraud investigators can be considered the next layer. These rules run in parallel with the data-driven models, namely models that use predictive modelling to assign fraud scores to transactions and report on potential cases in near real-time. Scoring rules are a type of expert system and are generally comprised of a set of if-else conditions. They are easy to create and understand by general users in a business context, although they do require revision and manual intervention. This is due to the ever-changing nature of fraudulent behaviour. While very useful, these rules can become quite expensive for the business to maintain in terms of the time it takes to understand their performance and make adequate adjustments to them.

Data-driven models run in parallel with scoring rules and are therefore also operating in near real-time. They essentially make use of machine learning algorithms to estimate the likelihood that a transaction or user is operating fraudulently. The benefit of data-driven models over scoring rules is that they can consider many data points when learning what fraud looks like, although they can tend to become black boxes and hence not easily interpretable by the business. The ability to extract trends from the data-driven models and implement those trends as simple rules would be ideal. Therefore it would be beneficial to consider very interpretable and less interpretable models when designing this layer while considering the balance between model accuracy and interpretability and what that means for the business overall.

The final layer of the fraud detection system is the *fraud investigator* layer. Here a team of fraud investigators receives alerts from both the scoring rules and data-driven model layers. These alerts get queued, with each alert containing a set of information related to the case, the probability of fraud score, and whether the alert was derived from the scoring rules or data-driven model layer, with the investigator ultimately marking the alert as fraud or non-fraud. Realistically it is impractical for a team of investigators to evaluate every alert that is generated, so in practice often only the highest scoring alerts are investigated. Part of the overall optimisation process is to detect as many true fraudulent cases as possible while limiting the number of false positives, *i.e.*, alerts for fraud are concluded as non-fraudulent. This investigation process is vitally important as the evaluations and their corresponding labels are fed back into the data-driven model to further learn and improve its performance. Were the data-driven model to inform the scoring

rule layer, then these labels would indirectly also impact the decisions regarding rule evaluation and performance monitoring.

2.3.3 Information feedback loop

Information feedback is vital to the accuracy and efficacy of the fraud detection system as a whole (Dal Pozzolo *et al.*, 2015). The primary source of information feedback in a fraud detection system is the outcome feedback received from the fraud investigators, although they do represent a small sample of the entire set of transactions processed. A secondary set of information comes from the users themselves reporting fraudulent behaviour on their accounts (Van Vlasselaer *et al.*, 2015). These would be cases where the fraud detection system does not flag a transaction as fraudulent, but the user confirms that it is fraudulent. There are not many of these cases, as the users often do not check their statements and therefore they seldom pick up the fraudulent transaction from their side. Nonetheless, it is common practice to assume that after a certain period has passed, all transactions that have been processed and have not been flagged as fraudulent can be considered genuine transactions. At this point, the fraudulent and non-fraudulent transactions and data points can be used to further train the data-driven models.

The scoring rules previously discussed do not learn from the data in the same way that the data-driven models do, and therefore the scoring rules are typically updated manually by an investigator, based on the perceived performance of the existing rules. In comparison, the data-driven models use the alerts that they generate on a specific day to train their algorithms to detect fraud the following day. Therefore there is a strong feedback loop between the alerts generated and how those alerts are scored, to retrain the data-driven models.

Dal Pozzolo (2015) proposes that these data-driven models require a relatively large set of data to train the model, hence the algorithms are typically trained at the end of a day (or any other period) once a sufficiently large set of feedback data is available. As the models run throughout the day, fraud investigators give feedback on each of the alerts received, therefore all feedback information would be received by the end of the day. In the day-to-day scenario, the model could collect feedback data at the end of the day, retrain the model at night, and use the retrained model from which to make new alerts the following day.

2.3.4 Relevant features

After any transaction is authorised in the fraud detection system, several transaction-specific data points are captured. These include variables such as the time of the transaction, the amount, through which bank the transaction was processed, *etc.*

To provide a better description of customer behaviour, further variables can be derived and added to the existing feature space at the time of the transaction. This process of feature augmentation involves adding aggregated variables to the feature space of the customer. Typical examples of aggregated features include the average transaction size of the customer, the number of transactions from the same card, the number of transactions per a specified period, *etc.* Additional feature types, such as network-related features, have also been proposed, as they help facilitate an understanding of the user within the greater network of transactions (Van Vlasselaer *et al.*, 2015).

The additional features can be considered aggregated features in that they provide an aggregated view of user behaviour. Including these aggregated features is vital to understanding how user behaviour progresses over time and can be used to detect anomalies related to users or transactions (Krivko, 2010). It must be noted that calculating aggregates of transactional data can be a computationally expensive exercise, thus they are often computed offline. The aggregated features are then brought together with the original transaction features to form a feature vector to be used for training the data-driven models. At this point, post-authorisation and feature augmentation, the scoring rules and data-driven models both score each vector.

Dal Pozzolo (2015) states that in practice, there is a problem with the inclusion of the unique identifier, namely the user or card ID in the feature vector. The issue is a sampling problem, where if the card ID is included in the feature vector, then the model might be biased towards subsequent transactions from a previously recognised card. It needs to be noted that in a real-world setting, a card will be blocked as soon as it is flagged for fraud. The post-fraudulent transactions could be removed from the historical data to train the model on a feature set more reflective of reality, but then it is highly likely that the class imbalance problem (to be discussed in greater detail in Section 2.8) will worsen. The proposed solution is to remove the unique identifier from the feature vector once the aggregated features are calculated. Each case of fraud will therefore be treated independently and the model will not be able to leverage the unique identifier for detecting cases of fraud. There is no perceived significant loss of information in removing the unique identifier, as the aggregated features provide adequate additional information for training.

2.3.5 Performance metrics for general statistical classification

Flach (2019) argues that using a single performance measure to evaluate the effectiveness of a model is inadequate. This is confirmed by Powers (2014) who suggests considering multiple performance metrics for classification. For general classification problems, a confusion matrix is typically used to evaluate the performance of

the classification by matching the predictions of each class with their corresponding predictions (Caelen, 2017). An example of a confusion matrix for two classes is presented in Table 2.1. The left axis of the table indicates the true values for each class, whereas the top axis indicates the predictions returned by the trained algorithm for each class.

Table 2.1: Example of a confusion matrix for binary classification

		Predicted class		Total
		Positive	Negative	
True class	Positive	TP	FN	$RP = TP + FN$
	Negative	FP	TN	$RN = FP + TN$
Total		$PP = TP + FP$	$PN = FN + TN$	N

The confusion matrix can be used to compute a variety of different performance measures. In the binary classification context, the confusion matrix is a matrix with two rows and two columns. This matrix reports on the False positive (FP), False negative (FN), True positive (TP), True negative (TN), Real Positives, Real Negatives, Predictive Positives, and Predictive Negatives after classification, where:

$$RP = TP + FN \quad (2.1)$$

$$RN = TN + FP \quad (2.2)$$

$$PP = TP + FP \quad (2.3)$$

$$PN = TN + FN \quad (2.4)$$

These count measures can be used to determine the proportion of correct classifications which is also known as classification accuracy. The mathematical representation of accuracy is given in (2.5).

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \quad (2.5)$$

Other relevant metrics that can be derived from a confusion matrix include precision and recall. Precision (see (2.6)), or positive predictive value, is the fraction of relevant instances among the retrieved instances. For anomaly detection purposes,

precision can be considered as the number of correctly classified anomalies divided by the total number of records classified as anomalies (whether true or false).

$$\text{Precision} = \frac{TP}{TP + FP} \quad (2.6)$$

Recall, as expressed by (2.7), is the fraction of the total relevant instances that were retrieved. In anomaly detection, recall can be considered the number of correctly classified anomalies divided by the total number of true anomalies in the data.

$$\text{Recall} = \frac{TP}{TP + FN} \quad (2.7)$$

Another binary classification metric is the F -score or F -measure (see (2.8)). It is derived from precision and recall, where F_1 equates to the harmonic mean between precision and recall.

$$F_1 = 2 * \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}} \quad (2.8)$$

An F_β metric (see (2.9)) can also be used, where β applies a weighting such that recall is considered β times as important as precision (Busa-Fekete and Dembczynski, 2015). This approach can be useful when compared with F_1 as it allows the decision maker to place more emphasis on either precision or recall, providing a more customisable measure.

$$F_\beta = (1 + \beta^2) * \frac{\text{Precision} * \text{Recall}}{(\beta^2 * \text{Precision}) + \text{Recall}} \quad (2.9)$$

The F -score is widely acknowledged as a useful metric which is used in many studies (Manning *et al.*, 2008); however, it has also faced some criticism. Powers (2014) states that F -scores do not consider true negatives and that alternative classification measures, such as the Matthews Correlation Coefficient (MCC), would be preferable in assessing the performance of an algorithm in a binary classification context. Powers (2007) explains that because F -scores ignore true negatives, they may be misleading for unbalanced classes. A preferable measure would be symmetrical and therefore assesses both directions of predictability. Chicco and Jurman (2020) share this sentiment, when the authors claim that the F -score is less truthful and informative when compared with the MCC. Hand and Christen (2018) maintain the same view while criticising F_1 -scores specifically, stating that the equal importance

placed on precision and recall is problematic, as in practice, different misclassifications would have different cost implications. Misclassification costs are particularly relevant in fraud detection where it is often more expensive to misclassify a positive instance of fraud as negative, than a non-fraudulent instance as positive (Sammut and Webb, 2010).

In this study, fraud detection is the main objective and because of the imbalanced real-world data and the significant role cost of misclassification will have, the criticism mentioned above plays a very significant role and the F -score will not be used as an exclusive measure of model performance. The F -score will be discarded where the context of the problem requires it. This will be elaborated on in Chapter 6 where the experimental results of the study are presented.

2.3.6 Performance metrics for fraud detection systems

Selecting the appropriate performance metrics for the fraud detection system is a challenge. There are several considerations to take into account when deciding on the ideal performance metric to use, including the cost of fraud, how long it takes to detect fraud, unreported cases of fraud, company reputation, and the problem of class imbalance. It has been suggested by Provost (2000) that traditional classification measures are misleading in the context of highly imbalanced class problems. This finding is known as the accuracy paradox. Provost *et al.*, (1998) and Valverde-Albacete *et al.*, (2013) confirm that classification accuracy is not effective in measuring the performance of models with a class imbalance. This is problematic, as severe class imbalances are typical of the anomaly detection problem (Zimek and Schubert, 2017).

Bahnsen *et al.*, (2013) argue that cost-based metrics should be used to quantify the financial impact of fraud. A form of this cost-based metric would be a matrix that assigns a specific cost to each record in the confusion matrix. Further studies by Elkan (2001) and Zhou and Liu (2010) indicate that focussing cost-sensitive metrics on the benefit rather than the cost of fraud should be considered, as there is the risk that different baselines will be used when assessing the total cost. Bahnsen *et al.*, (2015) provide a solution to this by means of normalising the costs or benefits with regard to the maximum loss. When it comes to the cost and how this is associated with the transaction amount, two distinct cases can be considered, *i.e.*, fixed cost of false negatives or dependent on the transaction amount. In the fixed scenario, all cases of fraud are assigned an equal cost. The reasoning behind this approach is that seemingly small transactions would undoubtedly lead to much larger transactions as fraudsters become more comfortable in their methods. Alternatively, a transaction-dependent approach could be used where the real loss that would be incurred is accounted for.

In the transaction-dependent scenario above, the cost of a false negative (missed fraud) can be assumed to be equal to the transaction amount. The cost of true positives (correct alerts of fraud) and false positives (incorrect alerts of fraud) would be the time and effort it takes for the investigator to assess whether the case is indeed fraudulent or not. The costs associated with verifying whether a transaction is fraudulent, *i.e.*, the cost of a verification phone call is very small in comparison to the cost of an actual fraudulent transaction. One should also consider the potential loss of revenue due to blocking a non-fraudulent customer, who would undoubtedly have made additional genuine purchases in future were he/she not blocked.

Another factor in the cost determination is the reaction time of the fraud detection system. If the fraud detection system were able to react in a shorter time frame, then overall more cases of fraud could be prevented. The potential for losses needs to be considered as well, warn Duman and Ozcelik (2011). Blocking an account that has a higher potential to incur cost, *i.e.*, a card with a higher spending limit, can be considered a greater cost saver than blocking an account with a lower spending limit.

Modelling the task of detection via classification is therefore not as straightforward as modelling traditional classification problems. Fan and Zhu (2011) claim that classification and detection are somewhat different problems, and that popular classifiers built for detection often mistakenly optimise a criterion suited to classification and not detection. In detecting rare items, misclassification rates and other measures are often irrelevant. What is important, however, is the ability to rank the important rare events ahead of the rest. In a fraud detection context, investigators will simply not have enough time and resources to investigate every case that has the slightest indication of fraud. Rather, they will investigate only the cases with the highest perceived risk of fraud as per the fraud detection system’s output score. The reason for this is that organisations will seldom block an account without a thorough investigation by the fraud investigator taking place. This illustrates why the detection of rare events is a somewhat dissimilar problem to that of a typical binary classification problem with balanced classes. Now that the use of typical performance metrics has been motivated against, in the following subsections, a brief overview of the metrics used in a combined form during this study is presented.

Area Under the Curve

An alternate measure is the Area Under the Curve (AUC) proposed by Chawla (2009) although Hand (2009) deems the standard implementation of the AUC to be inappropriate as it effectively averages the misclassification costs for the two classes. Rather, an adapted implementation, using the Mann-Whitney statistic is proposed where transactions are ranked according to the *posterior* probability of being fraudulent,

with the likelihood that a case of fraud ranking higher than a genuine transaction being measured. The approach handles the class imbalance problem as discussed, and has been effectively used in many studies related to neural networks and other classification techniques (Johnson and Khoshgoftaar, 2019).

As its name suggests, the AUC is the area under the Receiver Operating Characteristic (ROC) curve. The ROC curve is obtained by plotting the True Positive Rate (TPR) against the False Positive Rate (FPR) at various threshold configurations. A threshold T assigns a class label against the prediction continuous random variable X , where instances are classified as positive where $X \geq T$ or negative where $X < T$.

Average Precision

The approach to compute the average precision where only a few relevant results, k , are required is known as the Precision at k and is indicated by P^k .

Fan and Zhu (2011) propose the use of the Average Precision (AP) measure to help rank fraudulent items ahead of non-fraudulent items. The approach to compute the average precision where only a few relevant results, k , are required, is known as the Precision at k and is indicated by P^k . AP can be expressed as a finite sum over every position in a ranked sequence of transactions. The formula for P^k is presented in (2.10) where k is the rank in the sequence of retrieved transactions, n is the number of retrieved transactions, $P(k)$ is the precision at cut-off k in the list of transactions, and $\Delta r(k)$ is the change in recall from items $k - 1$ to k (Zhu, 2004).

$$P^k = \sum_{k=1}^n P(k) \Delta r(k) \quad (2.10)$$

Precision at k can be expanded into a broader form of AP, summarising the precision-recall curve as the weighted mean of precisions achieved at each threshold. This is in turn weighted by the increase in recall from the previous threshold. The formula for this form of AP is displayed in (2.11), where P_n and R_n are the precision and recall values at the n th threshold. The values for AP range between 0 and 1 where a higher score is more desirable. Where random predictions are made, the AP corresponds to the fraction of positive classes.

$$AP = \sum_n (R_n - R_{n-1}) P_n \quad (2.11)$$

For a single-figure measure of precision across recall levels, the mean of the AP ($\overline{AP}$) can be taken. The formula for $\overline{AP}$ is defined in (2.12), where AP is averaged

over the number of classes N . In the context of this study, AP can be thought of as the $\overline{AP}$ across the relevant classes, providing a single performance metric score that can be utilised as a comparative metric against other single-figure measures throughout the study.

$$\overline{AP} = \frac{1}{N} \sum_{i=1}^N AP_i \quad (2.12)$$

Success in the detection context means that the relevant items are ranked higher ahead of the rest of the items. Therefore the higher the AP, the better the algorithm performs at ranking the relevant items. An algorithm that can rank all relevant items higher than the non-relevant items would attain an AP of 1 or 100%. AP also acts as the Area Under the Precision-Recall Curve (AUPRC), asserts Zhu (2004) and has been used effectively as a classification metric in several recent studies (He *et al.*, 2018; Mohapatra *et al.*, 2014; Revaud *et al.*, 2019).

Fraud detection systems are generally constrained by the number of fraud investigators who can verify whether a prediction is indeed fraudulent or not. Therefore within the limited range of predictions that can be presented to the fraud investigators, it is imperative that the best ranking exists within this limited number of predictions. It is important to have the highest possible precision for the first k predictions. This measure will be referred to as P_k . As discussed above, the Area Under the Precision-Recall Curve and the average precision both give a good indication of the overall ranking on the dataset. P_k , however, determines the desired best measure of ranking on the subset of k predictions only. While also known as alert precision, P_k is considered a widely used and important accuracy measure in fraud detection systems, according to Bhattacharyya *et al.*, (2011). Key to the calculation of the Area Under the Precision-Recall Curve is that the calculation does not make use of true negatives, and is therefore only concerned with the correct prediction of the minority class.

Matthews Correlation Coefficient

As presented by Chicco and Jurman (2020), the Matthews Correlation Coefficient (MCC) is a metric that is more useful for imbalanced class classification when compared to other metrics, such as Accuracy and F_1 score, which can be misleading in the imbalanced case. The MCC considers the proportion of each class in the confusion matrix, therefore yielding a high score only if the classifier does well on both the negative and positive elements. The MCC is defined as follows:

$$MCC = \frac{TP * TN - FP * FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \quad (2.13)$$

When assessing the formula for the MCC, it is apparent that higher MCC scores are only obtained in instances where good results are obtained for all four of the confusion matrix elements (true positives, false negatives, true negatives, false positives). The MCC acts as a correlation coefficient between the observed and predicted binary classifications, and it returns a value in the range $[-1, +1]$. The results of the MCC can be interpreted as follows: a coefficient of $+1$ would be considered perfect prediction, whereas coefficients of 0 or -1 represent predictions no better than random guessing and complete disagreement between the observations and predictions, respectively. The metric has been used in several studies related to classification (Boughorbel *et al.*, 2017; Liu *et al.*, 2015; Saqlain *et al.*, 2019).

2.4 Supervised vs unsupervised learning

Machine learning involves using statistical algorithms to perform a specific task without being explicitly programmed to perform this task. As mentioned by James *et al.*, (2013), most machine learning problems fall into one of two categories, namely supervised or unsupervised. Supervised learning entails having both predictor variables and response variables. In these cases, the point of interest is in fitting a statistical model that relates the response variable to the predictor variables, to make accurate predictions on new observations of predictor variables in future (Mohri *et al.*, 2012). Some examples of supervised learning techniques include decision trees, support vector machines, k-nearest neighbours, generalised linear models and Naive Bayes.

The second category of machine learning is what is known as unsupervised learning. Here, no associated response variables are available (Carcillo *et al.*, 2019). By definition, this can be considered a more challenging problem, as there is effectively nothing to predict. Instead of predicting response variables, unsupervised learning involves understanding the relationship between the variables or observations. A typical example tool used in unsupervised learning is clustering (Sun *et al.*, 2018). The idea of clustering is that an understanding as to whether the observations fall into natural groupings or not is gained. Examples of unsupervised learning include k-means clustering and principal component analysis.

The distinction between supervised and unsupervised learning is evident when examining the difference between Figures 2.1 and 2.2. In the supervised learning diagram, illustrated in Figure 2.1, there are two distinct classes, and an algorithm is effectively attempting to discriminate between the two classes by constructing an optimal boundary between them. In the unsupervised learning diagram, depicted in

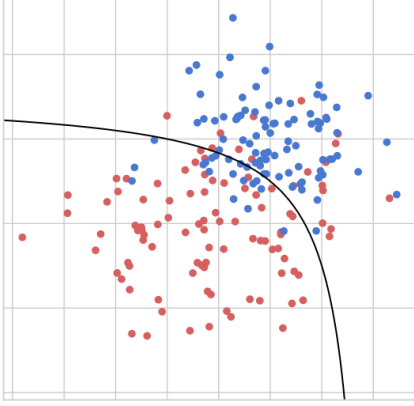


Figure 2.1: Supervised learning

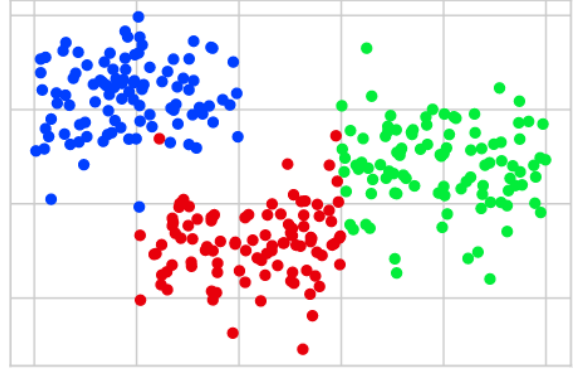


Figure 2.2: Unsupervised learning

Figure 2.2, there are no established classes of data; rather, an algorithm is attempting to create clusters of the data.

A third potential category is what is known as semi-supervised learning. These are cases where one has a complete set of predictor variables, but attached to these are a relatively incomplete set of response variables (Burkhart and Shan, 2020). This is a less familiar category and therefore falls beyond the scope of this research.

Different approaches can be formulated across more than one of the three categories described above. For example, anomaly detection approaches can be defined as supervised, unsupervised, or semi-supervised learning. Variables in a machine learning context can be considered to be either quantitative or qualitative (James *et al.*, 2013). Quantitative variables are essentially numerical values obtained through a measuring process, whereas qualitative variables involve properties that are observed and can generally not be measured with a numerical result. Modelling scenarios that involve a quantitative response variable are referred to as regression problems, whereas those with a qualitative response are known as classification problems. From a model perspective, some are strictly considered regression or classification models, while others can be adapted to perform in either domain. An example is logistic regression which is typically considered a binary classification model, but considering that it estimates class probabilities it could also be thought of as a regression method (James *et al.*, 2013).

Rodrigues and Omar (2014) explain that classification problems can be further broken down into binary and multi-class classification problems. These can be described as models with two distinct response variables and models with more than two

distinct response variables, respectively. In the machine learning domain, algorithms are selected based on whether the response variables are quantitative or qualitative. Less importance is placed on whether the predictor variables are quantitative or qualitative.

2.5 Algorithmic approaches to fraud detection

Supervised and unsupervised learning have both been proposed as approaches to solving the fraud detection problem (Carcillo *et al.*, 2019). As discovered previously, supervised learning involves training data with labels or target variables, whereas unsupervised learning does not involve the use of labels. Carcillo *et al.*, (2019) contend that unsupervised learning includes anomaly and outlier detection which attempt to discriminate from transactions and behaviour that deviate significantly from the majority, without any knowledge of the classes themselves. Krivko (2010) suggests making use of both approaches when attempting to detect fraudulent activity. This is because unsupervised techniques often produce false positives, although unsupervised techniques need not only be used for the prediction tasks, but could also be used from a clustering perspective to understand the general nature and different types of fraudulent behaviour.

As noted by Dal Pozzolo (2015), the fraud detection problem can be framed from either the transaction level or the user level. Phrasing the problem from a card perspective, modelling would be used to understand the behaviour of individual cards. Unsupervised methods are typically used in these scenarios, where only the previous history of the specific card is taken into account. While they do well to detect changes in behaviour, they do not determine the overall patterns of fraudulent behaviour that change over time. This approach introduces a problem when attempting to understand a behaviour change only, that is, the change may not always be attributed to fraudulent activity, *i.e.*, there may be a seasonal element influencing the model.

Transaction-level models operate at the transaction level and do not necessarily consider the behaviours attached to the card itself. Generally, the customer-specific behaviour models can be used as the initial model in the fraud detection system, with the transaction-level model acting as a secondary model (Van Vlasselaer *et al.*, 2015). Traditionally, the transaction-level models only consider each transaction in isolation, while not accounting for the previous transactional history of the user. A seemingly more comprehensive approach would, however, be to include customer behaviour features as discussed previously.

2.5.1 Supervised learning methods for fraud detection

Following on from the process of framing the fraud detection problem from the desired perspective and point in time, an understanding of the types of supervised or unsupervised approach that can be applied is required. Dal Pozzolo (2015) groups supervised learning approaches into four distinct groups, namely i) supervised profiling, ii) classification, iii) cost-sensitive, and iv) network model approaches.

Supervised profiling methods

Supervised profiling involves profiling the distribution of features used to detect fraudulent or non-fraudulent transactions or customers. The idea is to create a profile for each prospective class, and then compare new transactions and customers to these profiles to see how similar they are. One approach to supervised profiling involves using Weight of Evidence to determine the similarity between the classes (Siddiqi, 2012). Weight of Evidence indicates the predictive power of an independent variable over the dependent variable. While the approach was traditionally proposed to work in a credit risk environment, it can also be extended to fraud detection (Sudjianto *et al.*, 2010).

Besides using variable distribution to segment profiles, rule-based methods can also be used to profile customers (Kovach and Ruggiero, 2011). Because the nature of fraud changes constantly, any statistical-based rules need to be updated consistently to capture the latest patterns of fraudulent behaviour. Therefore it is ideal for these approaches to learn online instead of offline, unless updated frequently.

Classification methods

Classification approaches consistently appear as the go-to method for detecting fraudulent activity in the literature (Nejem and Kadeem, 2021). Common underlying classification algorithms proposed to solve the fraud detection problem include logistic regression, support vector machines, decision trees, association rules, and neural networks (Lata *et al.*, 2015; Suryanarayana *et al.*, 2018). In general, the data used for fraud detection is usually highly imbalanced and skewed towards the non-fraudulent class (Sammur and Webb, 2010). Interestingly, some studies have shown that neural networks have performed well with a high level of class imbalance (Buda *et al.*, 2018; Li *et al.*, 2021).

While neural networks are reported to achieve good predictions, they are notoriously difficult to interpret and are widely considered to be black box models (Oh *et al.*, 2019). Because of this, fraud detection implementations often favour tree-based methods due to their ease of model interpretation (Bussmann *et al.*, 2020). Interpretation not only refers to the data scientist’s interpretation of the model output, but

also the business understanding output that is presented to the fraud investigators and analysts who need to understand why something was classified as fraudulent or not.

Bayesian methods as well as other probabilistic graphical modelling approaches are also reported as popular modelling alternatives (Sudjianto *et al.*, 2010). Studies consistently show that random forest classifiers achieve superior performance when compared to alternative classification approaches (Khare and Sait, 2018; Van Vlasselaer *et al.*, 2015).

Cost-sensitive methods

The problem of fraud detection is inherently a business or commercial problem; therefore there are always cost complications that need to be factored into any solution proposed (Bahnsen *et al.*, 2014). Cost-sensitive algorithms take into account any costs that span from a misclassification; that is, from false positives and false negatives. Elkan (2001) explains that these algorithms can help deal with the unbalanced class distribution problem due to their ability to assign larger costs to the minority class. They are largely boosting algorithms which means that they are ensemble algorithms used to reduce bias and variance. The premise is that a group of individually weak classifiers could be used to create a single strong classifier.

AdaCost is a popular cost-sensitive algorithm, but one in which costs are assumed to be fixed (Fan *et al.*, 1999). In a fraud context, costs are not necessarily fixed and are rather proportional to the transactional amount of the fraud. Other algorithms, such as cost-sensitive decision trees, are transaction dependent and therefore represent the fraud problem better (Bahnsen *et al.*, 2015). They are beneficial when the aim is to reduce the costs associated with fraud detection, where the loss function takes the cost of fraud into account.

In practice, however, these algorithms have shown to favour producing false positives due to the cost associated with misclassifying fraudulent transactions (Dal Pozzolo, 2015). This means that the burden placed on fraud investigators to evaluate predictions is increased unnecessarily so. This is not useful to a business that is only interested in evaluating more precise predictions.

Network model methods

Network model approaches consider the notion that fraudulent behaviours can in many ways be linked together (Akoglul *et al.*, 2015). Some examples of this include fraudsters sharing devices or credit cards. Fraud is not always as clear-cut, however, and often the links between transactions and devices or cards are more complex. Network modelling aims to detect the patterns in these complex relationships by

using principles of link analysis and graph theory. APATE, the algorithm proposed by Van Vlasselaer *et al.*, (2015), includes network-relevant features, represented as long, medium and short-term network scores, in the feature space of the modelling problem. Their work highlights an improvement in model performance when including these network features.

2.5.2 Unsupervised learning methods for fraud detection

Unsupervised learning methods can be used to understand the nature of unlabelled fraud-related data. They can be used to determine the types of fraud not yet understood by human fraud investigators. Unsupervised techniques also have the benefit of not being impacted by a class imbalance or mislabelled data, yet according to Bolton and Hand (2001) and Carneiro *et al.*, (2017), these methods are still underrepresented in the fraud detection literature. The techniques that are generally applied involve detecting outliers or profiling customers. Outlier detection, in a fraud detection context, may be used to identify a class of records which may or may not be fraudulent. Anomaly or outlier detection relates to the identification of rare items (Zimek and Schubert, 2017), and according to Chandola *et al.*, (2009), can be conducted using an unsupervised, supervised, or semi-supervised approach.

Kou *et al.*, (2011) describe the process of identifying outliers as one in which a baseline distribution of normal behaviour is established, where observations that exhibit the greatest distance from normal are detected. One such example is the peer group analysis method proposed by Bolton and Hand (2001) where customers are profiled and placed into certain groups or clusters. Transactions that are somewhat departed from the standard clusters are candidates for fraud evaluation. Other examples include self-organising maps and one-class support vector machines (Harwani *et al.*, 2020).

Dal Pozzolo (2015) argues that unsupervised learning methods are not without their challenges. Firstly, creating a normal region that covers all possible cases of normal customer behaviour is a challenge. The behaviour exhibited by fraudulent users is also constantly evolving and therefore the current idea of fraud will likely not be sufficient for future use. Once defined, the boundary between fraud and non-fraud is often fuzzy. The actions taken by fraudulent users are also constantly evolving towards what would traditionally be deemed as normal behaviour, therefore making the identification of truly normal behaviour over time a more difficult task to achieve. Finally, the behaviour of genuine users can also be somewhat erratic at times and is prone to containing noise, which could be classed along with the fraudulent outliers.

Outlier detection can be broken down into two distinct types, namely local and global outlier detection approaches (Cheng *et al.*, 2019). Local detection approaches

attempt to identify outliers that can be considered anomalous relative to a local subgroup of the full data set. These methods consider how much a data point deviates from the rest of the data set while considering the density of the local neighbourhood relative to that data point. Kim and Sohn (2012) propose the use of peer group analysis as an approach to model local outliers. This approach summarises the behaviour of a subgroup of data and attempts to detect data points that differ significantly from this peer group, where these data points previously behaved similarly to the peer group. The use of breakpoints is also proposed to mark the points where data differs significantly from previous behaviour. This comparison between old and new behaviours and their subsequent features, is not necessarily always captured by traditional outlier detection methods.

Global detection approaches model observations that deviate from the entire data set as a whole. Zimek *et al.*, (2013) propose several evolutionary algorithms for outlier detection in a higher-dimensional space, namely the brute force, outlier search, selection criterion, crossover and mutation algorithms. By studying a typical fraudulent data set holistically, one is likely to encounter a high degree of sparsity in the data. The idea is to project these data points into a lower-dimensional space to reduce the effects of this sparsity and then study the density of the underlying distributions. A potential upside of these techniques is that missing data can be handled effectively when using projection methods. The downside would be that visually inspecting the results would be very difficult.

2.6 Framing the fraud detection problem

The proposed problem can be framed from multiple diverse and competing perspectives. Firstly, if the problem exists within a closed ecosystem or an organisation, such as, for instance, an e-commerce store, then the problem can be viewed from the customer perspective or at the account level (Xie *et al.*, 2019). Account-level models incorporate both transactional variables, as well as high-level behavioural indicators related to the account or cardholder. Generally speaking, these models are unsupervised because they need to attempt to detect a behaviour change, although they are not used to identify major overall patterns or trends in fraudulent behaviour.

A second perspective from which to view the problem is at the transaction level (Bahnsen *et al.*, 2016). Here account-level data is not entirely important, rather each user’s transactions are viewed and screened for potential fraudulent behaviour in isolation. Dal Pozzolo (2015) suggests that transaction-level models be used as a second line of defence against fraud, where the first line of defence is models, based on account or customer-specific behaviour. For transaction-level models, traditionally the historical behaviour from the account level or previous transactions is not

taken into account, although research shows that it is useful to include account-level variables in the modelling process (Whitrow *et al.*, 2008).

2.7 Trade-off between accuracy and interpretability

Machine learning models, regardless of whether they are considered to be a classification or regression model, have varying degrees of flexibility (James *et al.*, 2013). Flexibility refers to the shapes that the model can construct to estimate a function f . An example of a model with low flexibility would be linear regression which is only able to generate linear functions, such as lines or planes. Alternatively, models, such as support vector machines, can provide a far wider range of possible shapes to estimate the function f . Naturally, a trade-off occurs as the more flexible a model is, the less interpretable it becomes. Conversely, the more restrictive a model is the more the interpretability of the model increases.

Interpretability in this context refers to the ability of the decision maker working with the model to understand how the predictor variables are associated with the response variable. James *et al.*, (2013) advise that the use case of the model determines the degree of interpretability required. In the instance where only the prediction accuracy is required, then a highly complex and flexible model might be used, sacrificing the need to interpret the model's results. Such a case might be in the financial services industry where the focus is purely on predicting the price movement of a particular stock.

Alternatively, an organisation might be interested in a good enough model in terms of accuracy, but also one where the contribution of the predictor variables can be interpreted. An example of this case would be in a fraud detection setting where the contributing variables are vital to understanding, as they either inform an expert system or inform users of the changing and dynamic behaviour of the persons conducting fraudulent activities. Surprisingly, more complicated models do not always outperform less flexible models in terms of prediction accuracy. Often, more accurate predictions are reached when using less flexible models. This likely occurs due to the overly complex model exhibiting what is known as overfitting (Claeskens and Hjort, 2008).

When applying machine learning algorithms, it is good practice to consider several different and competing methods. The reason for this is that there is generally no single model that dominates all other models in any particular real-world application. While an algorithm might perform exceptionally well in a particular scenario, there is no guarantee that this same algorithm would be the preferred algorithm in another domain or using other data. This is known as the no free lunch theorem in statistics (Bhaskara, 2017).

2.8 Class imbalance

In the context of a binary classification problem, a class imbalance occurs when one of the two classes represents a far greater proportion of the sample set. A typical example of this would be in the credit card fraud detection space. Generally, fraud levels are relatively low compared with valid transactions, therefore by default the organisation being impacted collects far more records on valid transactions compared with fraudulent transactions. This means that there is far less fraudulent data from which a model can learn.

Burnaev *et al.*, (2015) propose that in the case of a significant class imbalance the data be resampled. The reasoning behind this is that algorithms have better performance when trained on a balanced training set. A potential risk in applying sampling techniques is that they do not take any class information into account when oversampling or undersampling, although the techniques are relatively simple to implement and interpret.

Undersampling is the practice of decreasing the size of the majority class. This involves removing these observations at random. The practise assumes that the majority class has so many observations that removing some of them at random will not influence the distribution significantly. This is a valid assumption in many cases, although there is still a resulting risk that observations of significant value could be removed, therefore influencing the distribution (Dal Pozzolo, 2015). Regardless, this approach is popular in practice, as it is simple to implement.

An alternative sampling approach is oversampling. This involves increasing the number of minority samples at random, to bring the overall imbalance closer to an overall state of balance. A potential downside of oversampling the minority class until it is equally represented when compared to the majority class is the risk of overfitting, due to an increased bias towards the minority class (Drummond and Holte, 2003). Other potential downsides are that oversampling the minority class does not add any information value, and that oversampling increases the training time of the model. These problems are inevitably compounded when the data set itself is large.

A popular implementation of oversampling the minority class is called Synthetic Minority Oversampling Technique (SMOTE). Chawla *et al.*, (2002) proposed the use of synthetic examples of the minority class in the neighbourhood of observed examples. The intention behind SMOTE is to generate new samples by interpolating between samples of the minority class. The technique shows favourable results in the performance improvement of baseline classifiers, although certain drawbacks to this technique do exist. One drawback is that observations are generated without considering neighbouring examples, thereby increasing the likelihood of overlapping samples. The Adaptive Synthetic (ADASYN) sampling approach to oversampling

the minority class, is one potential solution to the drawback realised by the SMOTE technique. He *et al.*, (2008) explain that ADASYN uses a weighted distribution for different minority class examples according to the level of difficulty of their learning. Therefore more synthetic data is generated for the minority class that is more difficult to learn compared to those minority class examples that are easier to learn.

Stratified sampling is another alternative approach to both under- and oversampling. According to Parsons (2017), it is used to divide a target population’s elements into distinct groups or strata. The intention is to create strata with homogeneous elements that act as partitioned subgroups of the population. This can be useful in stratifying imbalanced data based on the target variable. In this way, the appropriate number of positive or negative classes can be controlled throughout the model training process. There are several benefits to stratified sampling over simple random sampling. These include a smaller error in estimation if measurements in the strata have a lower standard deviation, measurements are generally more manageable when grouped into strata, and it is useful to have estimates of population parameters for groups within the population (Thompson, 2012).

The mean and variance of stratified sampling are given by the following equations:

$$\bar{x} = \frac{1}{N} \sum_{h=1}^L N_h \bar{x}_h, \quad (2.14)$$

$$s_{\bar{x}}^2 = \sum_{h=1}^L \left(\frac{N_h}{N}\right)^2 \left(\frac{N_h - n_h}{N_h}\right) \frac{s_h^2}{n_h}, \quad (2.15)$$

where L is the number of strata, N the sum of all of the strata sizes, and N_h the size of stratum h . Additionally, $\bar{x}_h$, n_h and s_h represent the sample mean, number of observations, and sample standard deviation of stratum h , respectively.

2.9 Concept drift

Often a difference in the distribution of the training and testing sets occurs due to an adaptation of how the data has been generated. This difference or change in the distribution is referred to as concept drift in the literature (Souza *et al.*, 2020). In a non-stationary environment, such as the fraud detection environment, the general reliance is on current data only and irrelevant or seemingly outdated training samples are not considered. A problem is presented in cases where concepts that were learned previously could occur again in time. If the outdated training data has been removed

from consideration, then the information relevant to these previously learned concepts is lost. The loss of information is referred to in the literature as the stability-plasticity dilemma (Mermillod *et al.*, 2013).

Concept drift can generally be categorised into one of two types, namely active and passive adaptation (Liu *et al.*, 2021a). Active adaptation involves conducting a change detection test and making the necessary changes based on whether or not a change was detected in the test. This type of adaptation works well when the distribution of the data changes drastically instead of gradually. In practice, if a significant change is detected, the underlying model is retrained on more recent samples that would be representative of the changes detected.

Change detection, according to Hoens *et al.*, (2012), works by either checking the distribution of the features used or by detecting errors due to misclassification. Checking the distribution of features helps to detect potential changes in the distribution that may indicate a class prior change or a change within the class. Alternatively, misclassification errors would be used to help detect a change in the class boundary. However, the severe class imbalance that is typically present in fraud detection problems makes change detection a difficult exercise due to the lack of data corresponding to the minority class. Once enough data has been collected on the minority class, the process of change detection will yield better results.

Passive adaptation does not work on the same premise as that of the active adaptation approach; that is, change detection tests are not used in the passive adaptation approach. Here the model is updated as soon as new supervised samples are generated and are available to use for training. These approaches work well in instances where changes are realised more gradually. A popular approach is to use an ensemble of classifiers to determine what has been learned historically, as well as what knowledge has been acquired from more recent samples (Liu *et al.*, 2021a). Each classifier within the ensemble has a relative weighting that is used to determine the influence of the classifier on the model. If this weighting or contribution to the ensemble drops below a certain threshold, then that classifier is deemed to not be reflective of the current concepts and is therefore removed from the ensemble.

Bifet *et al.*, (2015) claim that another approach to passive adaptation involves training a classifier, using a sliding window of the incoming samples. The difficulty with this approach is calibrating the sliding window so that it coincides with the rate of change within the distribution. Some methods propose using a fixed window size (Liu *et al.*, 2021a), whilst others propose an expanding window in scenarios where changes are not detected or a contracting window, *i.e.*, where changes in the distribution are detected (Liu *et al.*, 2017).

2.10 Streaming data

As is often the case with live streaming applications, data is received with a high frequency and therefore presents significant challenges around the storage of the data samples. For a live fraud detection system, data will need to be processed in real-time and therefore it may not be feasible for the system to revisit previous transactional samples. Aggarwal (2007) refers to this dilemma as the one-pass constraint.

Domingos and Hulten (2000) propose the use of an algorithm known as Very Fast Decision Tree for mining streams of data. Other algorithms are also mentioned in the literature, namely derivatives of the VFDT (Gama *et al.*, 2006; Hulten *et al.*, 2001), but few are focussed on the class imbalance problem that is so prevalent in the fraud detection space. Current suggestions to solving the imbalance problem on streaming data involve rebalancing the data through oversampling the minority class or undersampling the majority class (Gryzb *et al.*, 2021; Leevy *et al.*, 2018). These approaches both have the potential to introduce significant risks into the process. Undersampling the majority class could potentially lead to the removal of negative samples with a high information value, whereas oversampling the minority class involves replicating the existing positive samples which could lead to overfitting.

An alternative method for addressing the class imbalance problem in streaming data is proposed by Gao *et al.*, (2007). This method involves accumulating the minority class samples in real-time, while downsampling the majority class samples until the minority class samples represent around 40% of the total sample. The older majority class samples are effectively replaced until this aforementioned balance is reached. Other variations involve not only accumulating the minority class, but also misclassified observations from the majority class to enhance the definition of the boundary between the two classes (Lichtenwalter and Chawla, 2009). Further approaches choose to accumulate minority class samples via several algorithmic approaches, namely using k-nearest neighbours, naive Bayes and Mahalanobis distance methods (Perez *et al.*, 2016; Purwanto *et al.*, 2018; Zhu *et al.*, 2020).

2.11 Autoencoders

Within the field of machine learning, there is a subset of techniques that falls within the term neural networks. Neural networks are used to solve several machine learning problem types, *i.e.*, supervised, unsupervised and semi-supervised. In this study, an autoencoder, a type of neural network trained in an unsupervised manner, will be used to detect anomalies within transactional data. Goodfellow *et al.*, (2016) declare that an autoencoder is a neural network that is trained to try to reconstruct its input to its output. Autoencoders can be used for anomaly detection (Ribeiro *et al.*, 2018; Zhou and Paffenroth, 2017), where the model is used to learn the most salient

features in the data, to reproduce these salient features. The reconstructed features for anomalies should be worse than the reconstructed features for the non-anomalous data, as the model is trained on non-anomalous data only. This makes autoencoders of interest and in this section, they will be described in detail.

The following explanation of the main elements of an autoencoder is based on the work of Schreyer *et al.*, (2018). Consider training an autoencoder neural network using a set of N aggregated user records

$$X = \{x^1, x^2, \dots, x^n\}, \quad (2.16)$$

where each aggregated user record consists of a tuple of k transactional attributes

$$x^i = (x_1^i, x_2^i, \dots, x_j^i, \dots, x_k^i), \quad (2.17)$$

with x_j^i representing the j^{th} transactional attribute of the i^{th} aggregated user record.

In terms of structure, an autoencoder has a hidden layer h that generates a function used to describe the model's input. It also contains two distinct parts, namely an encoder function $h = f_\theta(x)$ and a decoder function $r = g_\theta(h)$, which act as nonlinear mappings. The encoder function maps an input vector x^i into a compressed representation z^i within the latent space Z . The latent representation is then mapped back onto a reconstructed vector $\hat{x}^i$ by the decoder function, where $\hat{x}^i$ is a reconstruction of the original input space. The encoder and decoder are typically symmetrical architectures, each containing several layers of neurons. The encoder mapping of the autoencoder can be defined as

$$f_\theta^l(.) = \sigma^l(W^l(f_\theta^{l-1}(.) + b^l), \quad (2.18)$$

while the decoder is defined as

$$g_\theta^l(.) = \sigma'^l(W'^l(g_\theta^{l-1}(.) + d^l). \quad (2.19)$$

The non-linear activations are denoted by σ and σ' , *e.g.*, the Rectified Linear Unit (ReLU) activation function. θ denotes the parameters of the model, while W , b and d represent the model weights and biases, respectively. Finally, l denotes the number of hidden layers in the model.

The decoder function r attempts to produce a reconstruction of the input, hence it is termed reconstruction. It is not feasible nor useful for the autoencoder to simply learn to copy the input x perfectly. Instead, the model is restricted in the sense that it can only copy input that resembles the training data, and that it is only allowed to copy approximately. Because of these restrictions, the model is likely to learn useful representations of the data.

To achieve an accurate representation of the input, the model is trained to learn model parameters θ^* that minimise the reconstruction error between the input and its respective reconstruction as best as it can. The objective of the autoencoder training process is therefore to train a model that optimises

$$\arg \min_{\theta} \|X - g_{\theta}(f_{\theta}(X))\|. \quad (2.20)$$

Bengio (2009) argues that training an autoencoder is generally easier than training a deep neural network. Therefore autoencoders have been used as kinds of building blocks for deep neural networks, where each layer in the deep network can be represented by an autoencoder that can be trained in isolation. An example of an autoencoder is illustrated in Figure 2.3.

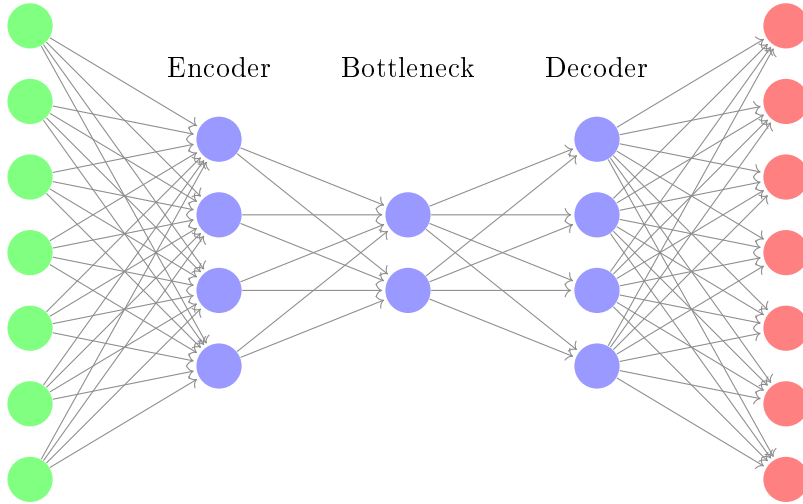


Figure 2.3: Example of an autoencoder

Several different autoencoder variations exist, each of which has its unique approach to model construction and training. These variations include undercomplete, regularised and variational autoencoders, and each is discussed in Sections 2.11.1, 2.11.2 and 2.11.3, respectively. The autoencoder variations are introduced to under-

stand which architectural configurations are more useful for transactional anomaly detection purposes.

2.11.1 Undercomplete autoencoders

The basic premise of autoencoders is to generate an encoder function h that learns the properties of the input. An approach to generating useful features from the encoder function is to limit h so that it is smaller than input x . This is what is known as an undercomplete autoencoder, *i.e.*, one in which the dimension of the encoder function h is smaller than the dimension of the input x . The architecture forces the encoder to learn the most informative features from the data.

According to Goodfellow *et al.*, (2016), the learning process for an autoencoder is described as minimising a loss function

$$L(x, g_{\theta}(f_{\theta}(x))), \quad (2.21)$$

where L is the loss function. The loss function penalises the function $g_{\theta}(f_{\theta}(x))$ for being too different from the input x . An example metric used to penalise would be the mean squared error. In the case where the autoencoder is linear (has no activation function) and L is the mean square error, then the autoencoder is effectively acting in much the same way as Principal Component Analysis (PCA). Therefore, in this case, the autoencoder learns the principal components of the data.

Alternatively, an autoencoder with nonlinear encoder and decoder functions learns a much more powerful nonlinear generalisation of PCA. A danger of this approach is that if the encoder and decoder are given surplus capacity then they can simply learn to copy the data without learning any useful information about the distribution of the data.

2.11.2 Regularised autoencoders

Very similar issues arise with autoencoders whose input layer is of the same dimension as the hidden layers, as well as when an autoencoder is overcomplete, *i.e.*, when the hidden layers have a greater dimension than that of the input (Goodfellow *et al.*, 2016). In these two cases, only a linear encoder and decoder are required for the model to potentially learn to copy the input only, without learning about the distribution of the data.

In practice, it is prohibitive to limit the scale of the model upfront to prevent the copying issue. Using a regularised approach allows the use of more complex models by introducing a loss function that encourages the model to learn other properties of the

data (Zhao *et al.*, 2018). Regularised autoencoders are not simply copying their input to their output, but are rather factoring in additional properties such as the size of the derivative of the representation, the robustness of the model to noise or missing data, and the sparsity of the representation. Therefore, even overcomplete and nonlinear models can learn useful information about the data, even though the models have enough capacity to simply learn to copy their inputs to their outputs.

There are several approaches to building an autoencoder in a regularised fashion, including options such as sparse, denoising, and contractive autoencoders. These approaches, as well as some specifics around enforcing sparsity in the case of the sparse autoencoder, are presented below.

Sparse autoencoders

When sparsity is encouraged in the learning of feature representations, then improvements in classification performance are observed (Makhzani and Frey, 2013). The premise of sparse autoencoders is that a network may contain more hidden units when compared to the units of the input layer, but that only a fraction of the hidden units are active at once. The sparsity constraint $\Omega(h)$ imposed in this scenario essentially forces the model to learn unique representations of the input data. The loss function for a sparse autoencoder is therefore represented as follows:

$$L(x, g_{\theta}(f_{\theta}(x))) + \Omega(h). \quad (2.22)$$

Sparse autoencoders are generally used to learn features for a secondary task, such as classification (Goodfellow *et al.*, 2016). When a sparsity penalty is introduced into the loss function, the autoencoder is forced to learn unique statistical features of the data. These learned features have therefore been assessed for statistical relevancy and can be used as input into a classification task.

According to Ng (2011), a hidden unit can be considered active when its output value is closer to 1 or inactive when the output value is closer to 0. This assumes that a sigmoid activation function is used. In instances where a tanh activation function is used, an output value close to -1 would rather be representative of an inactive unit. For the autoencoder to discover the usual structures within the data, the autoencoder is constrained in such a manner that the hidden units are mostly inactive.

Advantages of sparsity

There are several advantages to implementing sparsity constraints (Glorot *et al.*, 2011). One benefit is the disentanglement of information. The activated neurons in sparse representations are generally well preserved and robust against small changes

to the input. Another benefit is that sparse representations are more likely to be linearly separable due to their representation in high-dimensional space.

Enforcing sparsity

The sparsity constraint $\Omega(h)$ can simply be considered as a regularisation term which can be enforced on the loss function in several ways. These sparsity enforcing methods include exploiting the Kullback-Leibler (KL) divergence or applying L1 or L2 regularisation terms to the activation.

KL-divergence

The Kullback-Leibler divergence is a standard function for measuring the difference between two distributions (Ng, 2011). Given an activation $a_j^{(l)}$ of hidden unit j within layer l of an autoencoder, the average activation of said hidden unit (averaged over m training examples) can be expressed as:

$$\hat{\rho}_j = \frac{1}{m} \sum_{i=1}^m [a_j^{(l)}(x^{(i)})]. \quad (2.23)$$

To encourage most neurons to be inactive, $\hat{\rho}_j$ is expected to be as close to 0 as possible. This method enforces the constraint $\hat{\rho}_j = \rho$ where ρ is the sparsity parameter. Here ρ typically takes on a value close to zero (typically 0.05) where the intention is for the average activation of each hidden unit j to be close to this value. To meet this constraint, the activations of the hidden units must mostly be small values close to 0.

To apply the above, a penalty term is then added to the loss function to penalise instances where $\hat{\rho}_j$ deviates significantly from ρ . As argued by Ng (2011), a good choice for the penalty term is the following:

$$\sum_{j=1}^s \rho \log \frac{\rho}{\hat{\rho}_j}, \quad (2.24)$$

where s is the number of neurons in the hidden layer, while j is summing over the hidden units in the network. $KL(\rho||\hat{\rho}_j)$ is then the KL-divergence between a Bernoulli random variable with mean ρ and a Bernoulli random variable with mean $\hat{\rho}_j$. This sparsity regularisation term can therefore be enforced in the following manner:

$$L(x, g_{\theta}(f_{\theta}(x))) + \lambda \sum_j KL(\rho||\hat{\rho}_j). \quad (2.25)$$

L1 and L2 regularisation

Another approach to enforcing sparsity involves applying L1 or L2 regularisation terms to the activations. L1 regularisation adds the absolute value of coefficients as a penalty, whereas L2 regularisation adds the squared magnitude of coefficients as a penalty term (Arpit *et al.*, 2016).

Regularisation attempts to reduce model overfitting by decreasing the magnitude of the weights. The regularisation term is added to the cost function, allowing the network to develop a preference for learning smaller weights, which ultimately assists with negating the impact of model overfitting.

The main difference between the two is that L1 regularisation tends to shrink the penalty coefficient to zero, whereas L2 regularisation moves coefficients towards zero, but will never reach zero. For this reason, L1 regularisation is often preferred for feature extraction (Schmidt *et al.*, 2007). A term can therefore be added to the loss function to penalise the absolute value of the vector of activations a in layer h for observation i , scaled by a tuning parameter λ :

$$L(x, g_{\theta}(f_{\theta}(x))) + \lambda \sum_i |a_i^{(h)}|. \quad (2.26)$$

Alternatively, an L2 regularisation term can be added to the loss function, where the square of the errors is penalised:

$$L(x, g_{\theta}(f_{\theta}(x))) + \lambda \sum_i (a_i^{(h)})^2. \quad (2.27)$$

L1 and L2 regularisation norms are also known as Mean Absolute Error (MAE) and Mean Square Error (MSE), respectively (Chai and Draxler, 2014). The MAE is a statistical approach that is used to determine the quality of an estimator by measuring the absolute value of the errors, whereas the MSE measures the average of the squares of the errors. The errors are the difference between the estimator's predictions and the true values. In the context of autoencoders, MAE and MSE can be used during model training by measuring the difference between the original input and the consequent reconstruction. When used for measuring the loss of the autoencoder training process, they are frequently referred to as the reconstruction error or reconstruction loss. The formula for the reconstruction loss is expressed as:

$$l(x, \hat{x}) = \frac{1}{n} \sum_{i=1}^n (x_i - \hat{x}_i)^2. \quad (2.28)$$

Bengio *et al.*, (2007) found that when training overcomplete autoencoders with stochastic gradient descent that useful representations are yielded. An explanation for this is offered by Collobert and Bengio (2004) who claim that stochastic gradient descent with early stopping is similar to L2 regularisation of the parameters, where stochastic gradient descent is an iterative approach for optimising an objective function. The findings generated by Bengio *et al.*, (2007) suggest that the optimisation results in encodings that only work well for data similar to those in the training data, which is ideal. In addition to constraining the encoding via regularisation, another strategy is to make use of a sparsity constraint in the loss function. A further strategy is to add noise to the encoding.

Denoising autoencoders

Denoising autoencoders follow a different approach when compared with sparse autoencoders. Goodfellow *et al.*, (2016) state that denoising autoencoders learn something useful in the data by changing the reconstruction error terms of the cost function. These autoencoders, therefore, take a somewhat corrupted input and are trained to reconstruct the original uncorrupted version of the input. Denoising autoencoders instead minimise

$$L(x, g_{\theta}(f_{\theta}(\tilde{x}))), \quad (2.29)$$

where $\tilde{x}$ is a version of x that has been corrupted through the introduction of a form of noise. In this case, the input data is corrupted slightly, while the target output remains uncorrupted. Assumptions are made using this approach, including the fact that higher-level representations are robust to the corruption of the input data, and that the autoencoder captures features that represent the structure in the distribution of the input data (Vincent *et al.*, 2010). Some examples of corruption noise that are added to the training process include Gaussian noise, Salt-and-pepper noise, and Masking noise (Vincent *et al.*, 2010).

Contractive autoencoders

Contractive autoencoders, another variety of autoencoder, adds an explicit regulariser on the $h = f_{\theta}(x)$ in order to encourage the derivatives of f to be as small as possible (Goodfellow *et al.*, 2016). The regulariser is the squared Frobenius norm (the sum of squared elements) of the Jacobian matrix of partial derivatives (encoder activations) associated with the encoder function. The regulariser is only applied to the training data and therefore forces the model to learn useful information about the distribution of the training data. This regularisation term is therefore enforced in the contractive autoencoder in the following manner:

$$L(x, g_\theta(f_\theta(x))) + \lambda \sum_i \|\nabla_x h_i\|^2. \quad (2.30)$$

A detailed mathematical discussion on contractive autoencoders, Frobenius norm, and the Jacobian matrix concepts do not form part of this discussion, however, details of these concepts may be obtained in Rifai *et al.*, (2011).

2.11.3 Variational autoencoders

A Variational Autoencoder (VAE) provides a probabilistic manner for describing an observation in latent space. They are essentially generative models and are only associated with generic autoencoders due to the shared nature of their architectural styles. An and Cho (2015) note that variational autoencoders are directed probabilistic graphical models where the *posterior* is approximated by a neural network. The models intend to simulate how the data is generated to understand the underlying causal relations.

2.11.4 Network architecture

Autoencoders are typically only constructed of a single layer encoder and a single layer decoder (Goodfellow *et al.*, 2016). There are benefits of expanding the simple architecture presented above to a deeper architecture. Some advantages of using deeper encoders and decoders are that a deeper network can exponentially reduce the cost of representing some functions, the amount of training data required to learn some functions decreases, and in an experimental setting, deeper autoencoders are shown to yield better compression when compared to shallow encoders (Hinton and Salakhutdinov, 2006).

The universal approximation theory states that networks with a linear input and at least one hidden layer with an activation function can approximate a function from one finite-dimensional space to another (Goodfellow *et al.*, 2016). The implication of this is that regardless of the function, the function can be represented whether or not the network learned the function. Reasons why the function might not be learned, yet are represented include incorrect hyperparameter selection and overfitting. Another claim of the universal approximation theorem is that in theory there exists a network large enough to achieve any degree of accuracy required; however, it is not stated just how large the network would be.

According to Bengio (2009), the problem of training deep network architectures can be decomposed into several sub-problems, each representing a different level of abstraction within the network architecture. Because unsupervised learning is used, it can construct important information about the input distribution for the

layer in question. It is proposed (Bengio, 2009) that this approach can be extended sequentially across the network; that is, using unsupervised learning to construct important features from the first layer, and subsequently using the output of this first layer *i.e.* the features learnt, as input into the second layer and so on. The premise is that higher levels of abstraction are learnt, ultimately characterising the input data. This approach also involves managing the constituent parts of the network *i.e.* training each subsequent layer locally. The proposed benefit of such an approach is that the gradient diffusion issue that arises is avoided when using gradient descent by optimising for a single global criterion.

2.11.5 Applications of autoencoders

There are several different applications of autoencoders. Some of these applications include reducing the dimensionality of high-dimensional data and detecting anomalies in data.

Dimensionality reduction

Dimensionality reduction is arguably one of the first proposed applications of autoencoders (Goodfellow *et al.*, 2016). The intention of using autoencoders for dimensionality reduction is to find a method of projecting a high-dimensional feature space down into a lower-dimensional space. The concept was introduced by Hinton and Salakhutdinov (2006) where the authors showed that using an autoencoder for dimensionality reduction yields superior results when compared to simpler dimensionality reduction approaches, such as PCA.

One of the benefits of reducing the dimensionality of high dimensional data is the performance improvements realised in tasks such as classification (Goodfellow *et al.*, 2016). The approach is known as feature learning or representation learning, and it automatically detects convenient features for classification purposes (Bengio *et al.*, 2013). Feature learning is important in deep learning, as it replaces the manual task of feature engineering.

Relationship with principal component analysis

As established, PCA is another approach for dimensionality reduction (Axenie *et al.*, 2019). Chicco *et al.*, (2014) warn that simple autoencoders with single sigmoid hidden layers or those using linear activation functions are strongly related to PCA. However, it is noted that the non-linearity inherent in more complex autoencoders can determine more powerful generalisations when compared with PCA (Hinton and Salakhutdinov, 2006). The reconstructions have also shown a lower loss of information when compared with PCA.

Anomaly detection

A promising field for the application of autoencoders is anomaly detection. In this context, the autoencoder finds the lower dimensional embeddings of the original data where it is expected that anomalies and normal data are separable (An and Cho, 2015). These lower-dimensional embeddings are then transformed (reconstructed) back into the higher dimensional space of the original data. An error, known as the reconstruction error, is determined by calculating the error between the original data points and the data points reconstructed from the lower dimension back to the higher dimension.

Using autoencoders for anomaly detection is considered a semi-supervised or unsupervised learning approach. The autoencoder is trained only with normal instances of the data. The same loss functions used for training the sparse autoencoder, MAE and MSE, can also be used to evaluate the effectiveness of the reconstruction by determining the reconstruction error. Considering that the autoencoder is trained only on normal instances, it will reconstruct the normal data well, hence achieving a low reconstruction for these instances. Conversely, it is expected that anomalies will achieve a high reconstruction error, as the autoencoder had not learned these instances during the training process.

2.11.6 Related works on autoencoders for anomaly detection

Using autoencoders for anomaly detection and other related studies forms an important part of the background knowledge for this study. Therefore, additional studies related to using autoencoders for anomaly detection are summarised in Tables 2.2 through 2.7. The anomaly detection applications of autoencoders are vast, spanning important areas, such as industrial and maintenance, fraud detection, security, building and Internet of Things (IoT), process, and computing applications. Reported fraud detection applications show that autoencoders are an effective approach for detecting anomalies in a financial setting.

Table 2.2: Recent studies on the **industrial and maintenance** application of autoencoders

Summary of related work		
Author	Title	Research theme
Liu <i>et al.</i> , (2021b)	Semi-supervised anomaly detection with dual prototypes autoencoder for industrial surface inspection	A semi-supervised approach, the dual prototype encoder (DPAE), is presented. The autoencoder is used to detect potential surface quality defects of industrial products in instances where there is low availability of defective samples.
Alfeo <i>et al.</i> , (2020)	Using an autoencoder in the design of an anomaly detector for smart manufacturing	The authors present an anomaly detection approach for detecting smart manufacturing anomalies.
Lutz <i>et al.</i> , (2020)	Evaluation of anomaly detection of an autoencoder based on maintenance information and Scada-data	The authors propose a new method for monitoring the health of a wind turbine using an under-complete autoencoder. The model is trained to detect anomalies before an event occurs, which aids in predicting failures before they occur.
Wu <i>et al.</i> , (2020)	Fault-attention generative probabilistic adversarial autoencoder for machine anomaly detection	A fault-attention generative probabilistic adversarial autoencoder (FGPAA) is proposed for predictive maintenance purposes.

Table 2.3: Recent studies on the **fraud detection** application of autoencoders

Summary of related work		
Author	Title	Research theme
Alazizi <i>et al.</i> , (2020)	Dual sequential variational autoencoders for fraud detection	A generative model that takes into account the sequential nature of data is presented. The model consists of two variational autoencoders which are used to generate a condensed representation of the input sequential data.
Chang (2020)	Managing credit card fraud risk by autoencoders	In this paper, an anomaly detection approach is applied to identify fraudulent credit card transactions.
Schultz and Tropmann-Frick (2020)	Autoencoder neural networks versus external auditors: detecting unusual journal entries in financial statement audits	The authors propose using autoencoder neural networks to detect anomalous journal entries in financial accounts.
Schreyer <i>et al.</i> , (2018)	Detection of anomalies in large-scale accounting data, using deep autoencoder networks	The authors apply deep autoencoder networks to detect anomalies in financial journal entries, to identify fraud during financial statement audits or forensic investigations.

Table 2.4: Recent studies on the **security** application of autoencoders

Summary of related work		
Author	Title	Research theme
Chang <i>et al.</i> , (2020)	Clustering driven deep autoencoder for video anomaly detection	Implementing a novel convolutional autoencoder for detecting anomalies in surveillance videos. The approach separately captures the spatial and temporal informative representations.
Muhammad <i>et al.</i> , (2020)	Stacked autoencoder-based intrusion detection system to combat financial fraudulent activity	An intrusion detection system for Internet of Things devices is proposed, comprised of a stacked autoencoder together with a deep neural network.
Salahuddin <i>et al.</i> , (2020)	Time-based anomaly detection using autoencoder	The authors propose a novel approach to time-based anomaly detection using an autoencoder. Different time-windows are used to detect multiple DDoS attacks that are typically difficult to identify using standard flow-based features.
Chawla <i>et al.</i> , (2019)	Bidirectional LSTM autoencoder for sequence-based anomaly detection in cyber security	An autoencoder is used to detect anomalies in kernel sequence data of cybersecurity systems. The model was trained on normal call sequences found in the ADFA-LD data set.

Table 2.5: Recent studies on **building and IoT** applications of autoencoders

Summary of related work		
Author	Title	Research theme
Chow <i>et al.</i> , (2020)	Anomaly detection of defects in concrete structures with the convolutional autoencoder	Training a convolutional autoencoder using defect-free images, in order to detect defects in concrete structures.
Kim <i>et al.</i> , (2020)	APAD: autoencoder-based payload anomaly detection for industrial IoE	Detecting payload attacks in industrial Internet of Everything (IoE) applications. The approach employs an autoencoder to distinguish between normal and abnormal behaviour in low-performance devices.
Yin <i>et al.</i> , (2020)	Anomaly detection based on convolutional recurrent autoencoder for IoT time series	In order to improve service quality in IoT systems, an integrated model of the convolutional neural network and recurrent autoencoder is proposed for anomaly detection purposes.
Fan <i>et al.</i> , (2018)	Analytical investigation of autoencoder-based methods for unsupervised anomaly detection in building energy data	The construction of an ensemble of autoencoders with different network architectures and training schemes is proposed. The method is intended to assist in detecting operational faults in building energy data.

Table 2.6: Recent studies on the **process** application of autoencoders

Summary of related work		
Author	Title	Research theme
Elsayed <i>et al.</i> , (2020)	Network anomaly detection, using LSTM-based autoencoder	A combined anomaly detection approach using a Long Short Term Memory (LSTM) autoencoder and one-class support vector machine. The LSTM autoencoder is used to learn a compressed representation of the normal traffic data, which is then fed to the one-class support vector machine to discriminate between normal and anomalous data.
Nolle <i>et al.</i> , (2018)	Analysing business process anomalies using autoencoders	Applying autoencoders to sequential data to model internal business processes, to detect fraud and other inefficiencies. The approach does not rely on any prior knowledge of the underlying process and may be trained using noisy data that contains the anomalies of interest.
Xu <i>et al.</i> , (2018)	Unsupervised anomaly detection via variational autoencoder for seasonal KPIs in web applications	The use of an unsupervised variational autoencoder for detecting anomalies in web metrics data. A novel KDE interpretation of the reconstruction is developed.

Table 2.7: Recent studies on the **computing** application of autoencoders

Summary of related work		
Author	Title	Research theme
Merrill and Eskandarian (2020)	Modified autoencoder training and scoring for robust unsupervised anomaly detection in deep learning	This paper introduces three methods that overcome the shortcomings of unsupervised autoencoders, where the assumption is that normal inputs will produce lower reconstruction errors in instances when the training data contains anomalies.
Wang <i>et al.</i> , (2020b)	adVAE: A self-adversarial variational autoencoder with Gaussian anomaly prior knowledge for anomaly detection	A self-adversarial variational autoencoder is proposed, to overcome the data distribution recovery and overfitting properties of generative models.
Borghesi <i>et al.</i> , (2019)	A semisupervised autoencoder-based approach for anomaly detection in high-performance computing systems	Using an autoencoder to detect anomalies in a high-performance computing setting, including a comparison with traditional semi-supervised anomaly detection approaches.
Chen <i>et al.</i> , (2017)	Outlier detection with autoencoder ensembles	In this paper, autoencoder ensembles for unsupervised outlier detection are used.

The aim of the summaries presented in Tables 2.2 to 2.7 was to show the importance and versatility of autoencoders across many different application areas. These summaries also assist with putting the modelling approach used in the study into context, as well as concluding the discussion with a formal reference to literature where autoencoders were used.

2.12 Chapter conclusion

In this chapter, the first chapter in Part I, an overview of recent literature on anomaly detection, fraud detection, and autoencoders was provided. This included discussions related to measuring performance in an anomaly detection context, designing systems to detect fraud, and constructing sparse autoencoders to detect anomalies in a transactional setting.

In the next chapter, which is the second chapter in Part I of the study, a literature review of self-organising maps and social network analysis, and how these approaches can be used to enhance the anomaly detection process will be presented.

Chapter 3

Literature review: Self-organising maps and social network analysis

3.1 Chapter introduction

This chapter, which is the second chapter in Part I of the study, serves as an introduction to a variety of concepts related to self-organising maps (SOMs) and social network analysis (SNA). While the autoencoder models discussed in the previous chapter can be used to train an anomaly detection system, SOMs and SNA are believed to be able to describe the nature of the data itself and to determine network-related relationships in the data respectively.

Using a SOM to model the problem provides an unsupervised view of the information in the feature space. With this view in mind, the behaviour of the contributing features can be better understood. This concept will be illustrated in detail in Chapter 7. Introducing network data into the study opens up the possibility for social network analysis through a variety of network metric techniques. These network metrics are introduced in this chapter and are applied to the real-world data in Chapter 6, where they feature as additional inputs into the anomaly detection problem space.

Following on from an introduction to SOMs, a discussion on the various aspects and applications of SOMs is presented. Some of the important recent studies related to SOMs are then covered. This is succeeded by an introduction to SNA, as well as the concept of network metrics. In the penultimate section of the chapter, a summary

of the key studies associated with SNA is included. The chapter is then concluded with an overview of the key findings from the literature review.

3.2 Self-organising maps

In this section, an overview of the Self-Organising Map (SOM) model and the associated visualising techniques used in the study, beginning with an introduction to the algorithm are provided. Several steps in the model creation process are then discussed, including the building, training, evaluating, and interpreting of these models. The interpretation section includes a discussion on the major visualisation techniques used with SOMs, namely the clustering map, u-matrix, and component plane visualisations. Finally, the section is concluded by reflecting on recent studies and related applications.

3.2.1 Introducing self-organising maps

According to Kohonen (2001), SOMs are a type of artificial neural network that is trained in an unsupervised manner. It is typically used to produce a two-dimensional representation of higher-dimensional space, a process known as dimensionality reduction. While they are considered a class of artificial neural networks, they do not employ the same learning techniques as most other neural networks. SOMs make use of competitive learning to learn a feature map, based on the topology of the input space, rather than applying traditional neural network learning techniques, such as back-propagation with gradient descent.

SOMs are first trained whereafter they can be used for inference or mapping purposes. The training process is referred to as vector quantization, which is a technique for modelling probability density functions by the distribution of prototype vectors (Vellido *et al.*, 2021). The model inference process involves classifying a new input vector onto the map space. The mapped space is defined before training and involves constructing a two-dimensional space consisting of nodes arranged in a rectangular or hexagonal grid form.

Ponmalai and Kamath (2019) argue that high-dimensional data is difficult to analyse, as humans are only able to visualise data in two or three dimensions. The authors claim that SOMs are a dimensionality reduction technique that can be used to help interpret high-dimensional data without extensive computational requirements. A two-dimensional lattice of nodes is created, whereby a structure is imposed giving each node a location in $\mathbb{R}^2$ as represented by a vector $\mathbf{l}_i$ where i is the index of the node. The nodes in the two-dimensional lattice also have a corresponding position in the higher dimensional input space, as represented by $w_i \in \mathbb{R}^d$ where d denotes the dimension of the input space. Therefore each node within the lattice acts as a

connection between $\mathbb{R}^d$ and $\mathbb{R}^2$. A key concept of SOMs is that nodes that are close to one another in the lattice space should also be close to one another in the input space.

Training a SOM involves updating the locations of its associated weight vectors. The intention is to move the weight vectors to be representatives of the centres of clusters in the data and to mirror the order of the lattice (Ponmalai and Kamath, 2019). Starting with random weight vectors, iterating through the training process essentially moves the lattice nodes along with their associated neighbours. The magnitude of the move by the neighbour of each node is determined by a neighbourhood function and is proportional to the distance between the node and its neighbour in the lattice space.

3.2.2 Building self-organising maps

Two main designs have been used to define the structure of the lattice in SOMs, namely the rectangular and hexagonal designs. Kohonen (2014) states that the hexagonal structure is superior to the rectangular structure, both in terms of its visualisation abilities, as well as its performance in generating clusters of data. The improved clustering performance is because each hexagonal lattice has six neighbours, while each rectangular lattice has only four neighbours. This likely leads to a higher density of weight vectors where data points are densely populated. The size of the lattice is another consideration that needs to be made before training. According to Kohonen (2001), $5\sqrt{n}$ nodes, where n is the number of data points, is a feasible choice for lattice size determination.

Before initiating the training of a SOM, the weight vectors need to be initialised. While the weight vectors are bound to change throughout the training process, proper initialisation can assist in the algorithm converging sooner. Examples of initialisation approaches include random initialisation, random sampling initialisation (Haykin, 2009), best candidate sampling initialisation (Kamath and Fan, 2018), and principal component initialisation (Akinduko *et al.*, 2016). Ponmalai and Kamath (2019) state that principal component initialisation tends to find an equilibrium point in the fewest number of iterations when directly compared with the other initialisation approaches, while random initialisation typically takes the most number of training iterations to find an equilibrium point.

3.2.3 Training self-organising maps

As with some traditional machine learning algorithms, it is advisable to rescale the input into an acceptable range, such as between 0 and 1 (Misbahulmunir *et al.*, 2020). The generalised approach to training a SOM is given by Algorithm 3.1, where

training requires an initialised lattice of nodes and weight vectors. Each of the nodes within the lattice has a position within both the lattice, as well as within the input space. These positions are represented by two vectors, namely the lattice vector l_i and weight vector w_i , respectively, where $i \in \{1, 2, 3 \dots n\}$ with i being the index of the node and n the number of nodes within the lattice.

Algorithm 3.1: Generalised approach to training a SOM (Ponmalai and Kamath, 2019)

input : number of N training epochs, initialised lattice nodes L , initialised weight vectors W_1
output: trained SOM with weight vectors W_2
for $i = 1$ **to** N **do**
 $x \leftarrow$ pick a random point in the data set;
 $c \leftarrow$ the lattice node closest to x ;
 move the weight vector of c closer to x ;
 move the weight vectors of the neighbours of c slightly closer to x ;
end

There are two approaches to training SOM models, namely training via an online update or a batch update. The batch update approach proves to be beneficial due to it not requiring a learning rate function, therefore eliminating the number of required parameters (Ponmalai and Kamath, 2019). Other authors have argued in favour of the online update approach due to its proposed improved stability (Cottrell *et al.*, 2018). For practical applications of SOMs, Kohonen (2013) suggests that the batch update approach be taken as a result of the lack of the learning rate parameter, allowing for a simpler model convergence process.

The batch update approach considers the entire data set during each iteration of the training process. Given $w_i(k)$ for $i \in \{1, 2, 3 \dots n\}$ where k is the iteration of the training process, the Best Matching Unit (BMU) needs to be determined for all data points. The BMU of a data point can be defined as the node that is closest to the data point in the input space. The number of data points that have a particular node j as their BMU also needs to be determined. This value n_j is calculated as follows:

$$n_j = \sum_{x \in D} \left(\begin{cases} 1 & c(x) = j \\ 0 & \text{otherwise} \end{cases} \right), \quad (3.1)$$

where $c(x)$ is the index of the BMU of the point x and is expressed mathematically as follows:

$$c(x) = \arg \min_i \| x - w_i \| . \quad (3.2)$$

The batch update step is then described as:

$$w_i = \frac{\sum_{j=1}^n n_j h_{j,i} \bar{x}_j}{\sum_{j=1}^n n_j h_{j,i}} . \quad (3.3)$$

Within equation (3.3), $\bar{x}_j$ is the average of the closest input data points to the j -th lattice node, while $h_{j,i}$ is the neighbourhood distance from node j to node i . Ponmalai and Kamath (2019) found that the training process for batch updates still reaches an equilibrium, regardless of the lack of a learning rate. Equilibrium is determined by calculating the average displacement of the nodes from the previous training iteration, and if this displacement drops below some predetermined threshold ϵ , then the equilibrium is reached.

The neighbourhood function $h_{j,i}$ is another important consideration within the training process. This function effectively maps the input space and the lattice space together. $h_{j,i}$ denotes the neighbourhood distance for a winning node j to node i . According to Haykin (2009) and Kohonen (2001), the neighbourhood function has a few requirements to satisfy. The first requirement is that its maximum value should be reached when $\| l_i - l_j \| = 0$, where l_i and l_j are the lattice vectors for node i and the winning node j , respectively. This needs to be symmetric around the winning node j . The second requirement is that as $\| l_i - l_j \|$ increases, $h_{j,i}$ should decrease.

There are several different neighbourhood functions, namely the Gaussian and Bubble functions (Natita *et al.*, 2016), as well as the Triangle and Gaussian with cut-off functions (Ponmalai and Kamath, 2019). As reported by Ponmalai and Kamath (2019), the choice of the neighbourhood function is largely irrelevant. During the organising or initialising phase of the training process, it is unlikely that the map will self-organise. The authors then found that once convergence begins within the training process, the choice of neighbourhood function does not matter as long as the size of the neighbourhood only captures one or zero neighbours of a particular node.

3.2.4 Evaluating self-organising maps

It is inherently difficult to validate whether a trained SOM that has reached an equilibrium point represents the higher dimensional shape of the data set. Essentially

it is therefore difficult to evaluate the quality of the mapping between the lattice and the input space. A good quality mapping would preserve the topology of the high-dimensional input space.

As with supervised learning techniques, error metrics are vital for assessing the performance of a trained SOM. Erwin *et al.*, (1992) claim that while each of the various reported error metrics for SOM is useful for evaluation purposes, no single error metric provides a holistic explanation of the model’s quality. Rather, each of the metrics measures something different about the SOM that could collectively provide the required holistic explanation of quality.

The two main approaches for assessing the quality of a SOM are the quantization error (E_q) and the topographical error (E_t) (Tu and Hoan, 2016). If the values of these measurements are small, then the quality of the SOM is deemed to be good. Achieving a good quality SOM according to multiple measures is challenging, however, as there is no guarantee that a solution will simultaneously decrease Quantization error (E_q) and Topographical error (E_t) for any given data set.

E_q is the average distance of the input data points to the nearest node in the lattice (Breard, 2017). The formula for E_q is represented by:

$$E_q = \sum_i \|x_i - w_c\|^2, \quad (3.4)$$

where

$$c = \arg \min_i \|x - w_i\|. \quad (3.5)$$

Within equation (3.5), every competitive node i corresponds to a cluster and its weight vector w_i corresponds to a centroid of the cluster. In each epoch, a vector x is presented as input to the network. The distances from each centroid w_i are calculated, and finally, the winning node c with the minimum Euclidean distance is selected. w_c represents the BMU’s weight vector of sample x_i , where x_i is the input sample for training time i .

While E_q measures how well the SOM mapping fits the distribution of the input data, it does not measure how well the lattice fits the topology of the higher dimensional input space. E_q is therefore useful for determining whether a map has nodes near dense clusters. If more nodes are added to the lattice, a lower E_q value would be achieved, although this could lead to a model that overfits the input data.

E_t attempts to capture information that the quantization error does not capture, namely how well the shape of the high dimensional input space is preserved in the lower dimensional output space (Breard, 2017). E_t is calculated by taking the proportion of data points where the BMU and the second BMU are neighbours within the lower dimensional lattice space. The formula for E_t is represented by:

$$E_t = \frac{1}{|D|} \sum_{x \in D} te(x), \quad (3.6)$$

where

$$te(x) = \begin{cases} 1 & c_i(x) \text{ and } c_j(x) \text{ are neighbours,} \\ 0 & \text{otherwise.} \end{cases} \quad (3.7)$$

c_i is the index of the i -th BMU and D is the high dimensional input space. E_t indicates how well the SOM preserves the topology of the higher dimensional input space, and as stated by Ponmalai and Kamath (2019), it can be improved by using the shortest path distance between the first and second BMUs rather than simply returning 0 if they are not neighbours, as per equation (3.6). A lower value is preferred for E_t , where 0 indicates excellent topographic representation, meaning that all best and second-best matching nodes are neighbours. Conversely, a value of 1 indicates a case of maximum error, where the best and second-best nodes are never neighbours.

The Distribution Matching Error (DME) is another metric that attempts to measure whether a SOM's weight vectors appear to be drawn from the same distribution as the high dimensional input data. For each dimension within the input space, a two-sample Kolmogorov-Smirnov (KS) test is performed. In keeping with Jin *et al.*, (2019), a two-sample KS test can be used to determine if two samples are drawn from the same distribution. The DME is calculated by taking the proportion of dimensions where two-sample KS tests fail.

3.2.5 Interpreting self-organising maps

Olszewski (2014) claims that SOMs can be interpreted as a visualisation technique, therefore creating a map structure. The location of points in the two-dimensional map aims to represent the similarities between the data in the higher dimensional space. This allows for the visualisation of relationships between the data points in multi-dimensional space. Ponmalai and Kamath (2019) confirm this and argue that SOMs are intended to be used as visualisation tools and that SOM visualisation

approaches are either centred on analysing the location of nodes, or on analysing the input data that is mapped onto the nodes.

To illustrate the versatility of SOMs as a visualisation technique, three popular and well-known visualisation techniques will be discussed. The techniques are clustering maps, u-matrix and component planes.

Clustering map

The first visualisation technique is the clustering map, which indicates the different clusters of similar data points after running the SOM algorithm on a data set (Qian *et al.*, 2019). *MiniSom* is a Python-based implementation of SOMs that allows for the creation of multiple visualisation techniques (Vettigli, 2018). An example of a clustering map is presented in Figure 3.1. The technical details of clustering map implementation are omitted here as the goal is to give a high-level illustration of what the technique offers.

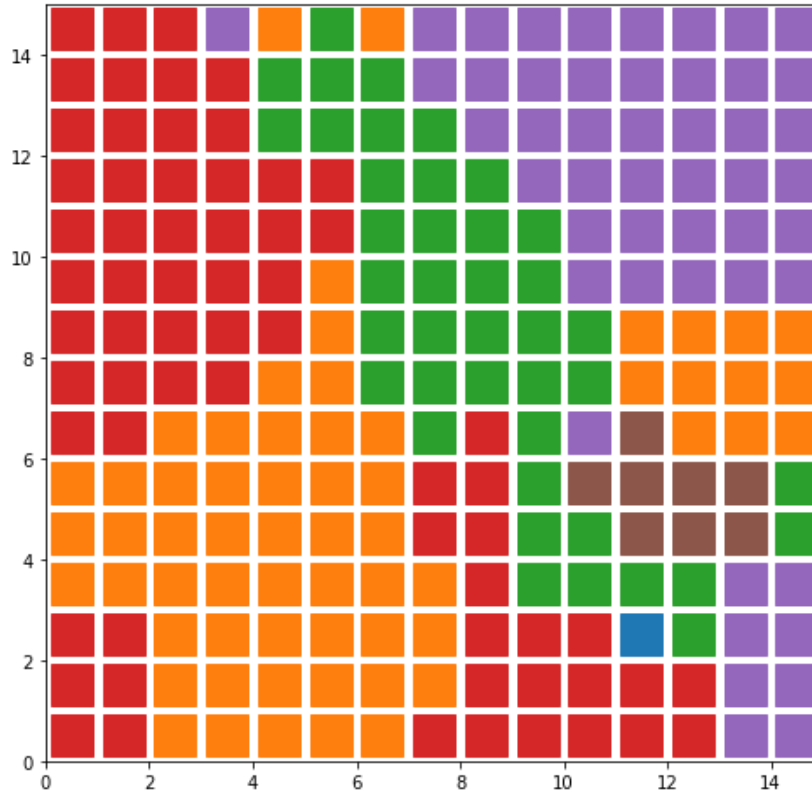


Figure 3.1: Example of a clustering map (Vettigli, 2018)

The clustering map in Figure 3.1 is mapped onto a 15x15 lattice, where six different colours are used to highlight the different clusters. Each node in the map is associated with the feature with the maximum weight, thereby segmenting the map into regions where specific features have high values.

U-matrix

In line with Qian *et al.*, (2019), the u-matrix is a map produced by mapping the average Euclidean distance between each node and its nearest neighbours in a high-dimensional space onto a two-dimensional lattice. An example of a u-matrix is depicted in Figure 3.2, where a pseudo-colour is used where the nodes of the maps are displayed as an array of cells and the colour represents the distance from the neighbour nodes (Vettigli, 2018). Markers have then been added to represent the samples mapped in the specific cells.

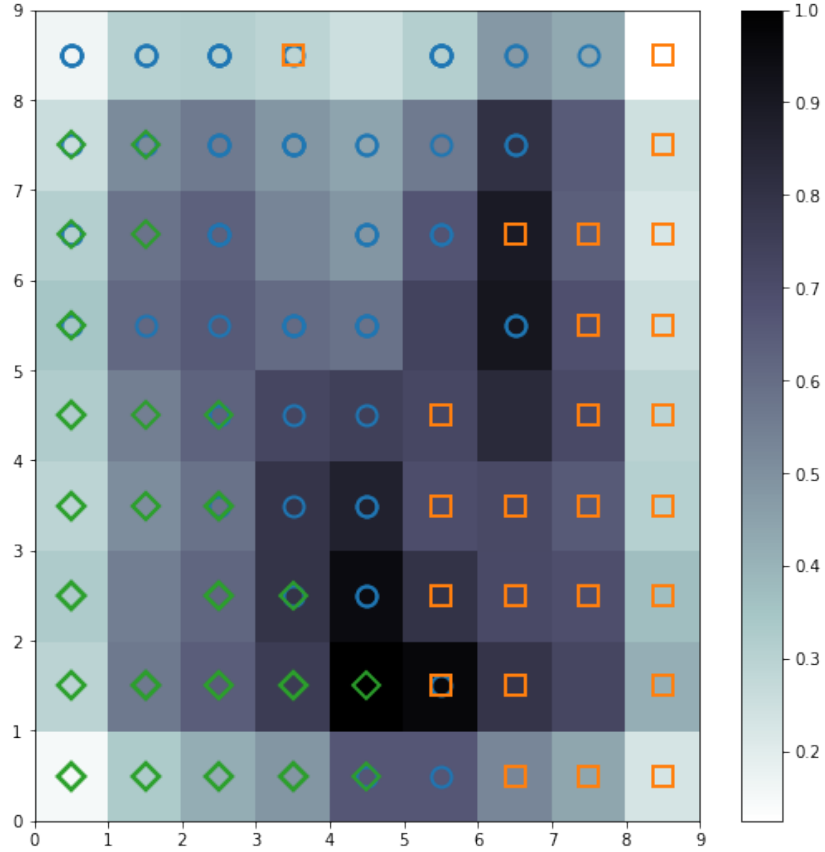


Figure 3.2: Example of a u-matrix (Vettigli, 2018)

The legend in Figure 3.2 indicates the similarity or distance of each node and its neighbours. The darker the node, the further this node is from its neighbours in the high-dimensional space. Conversely, the lighter the shade of the node, the closer or more similar each node is to its neighbours.

Component planes

Component planes are another visualisation technique that separately maps each feature of the node’s weights onto a two-dimensional lattice. As reported by Qian *et al.*, (2019), component planes are useful for producing a visual and quantitative understanding of the relationships between features, where features with a high correlation will have similar component planes. An example, taken from Vettigli (2018), of a component plane is illustrated in Figure 3.3.

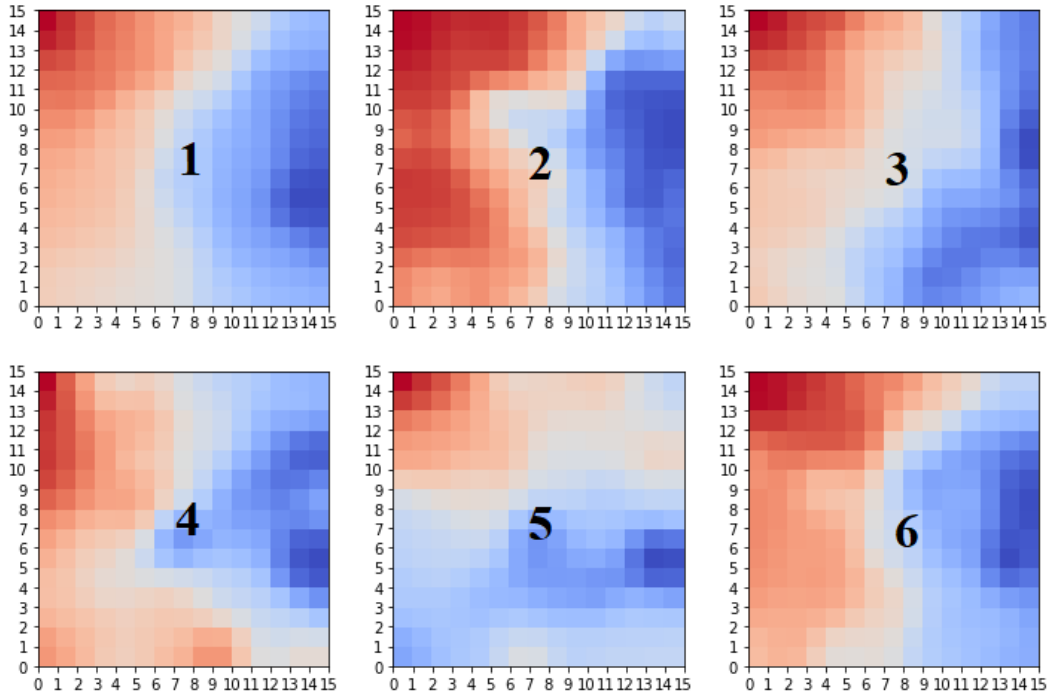


Figure 3.3: Examples of component planes (Vettigli, 2018)

In Figure 3.3, it is clear that the feature in the middle of the top row (*plane number 2*) and the feature to the right of the bottom row (*plane number 6*) have similar planes. This indicates that the features are strongly correlated. Were this SOM used for dimensionality reduction and feature selection purposes, then it can be concluded that including both of these two features in any further model training

process might be redundant. Ponmalai and Kamath (2019) also found that component planes can be used to determine what features contribute to what class. This can be achieved by comparing component planes for each feature to a clustering map, specifically noting where the clusters from the clustering map are activated on the feature plane. A particular class or cluster that has a higher value in its corresponding position within a feature plane lattice indicates that the feature is an important feature for that class.

Contrary to the output produced by some clustering algorithms, such as k-means, SOMs produce an output that is straightforward to visualise (Schreck, 2010). The three SOM interpretation techniques discussed in this section cover a variety of use cases. Cluster maps are useful for surfacing the features that have the highest weighting per node, although this summarised visual representation does not indicate the lower weighted yet potentially significant features per node or cluster. The u-matrix is useful for visualising the distances between nodes on the lattice, which assists with discriminating between regions of the SOM. For larger, high-dimensional problems, the clusters can be difficult to identify using the standard u-matrix. Finally, component planes allow practitioners to visualise multiple features simultaneously by arranging them in a matrix layout. By using a colour-coded visualisation for component planes, the contributions of the features towards the calculated distances between clusters can be determined, giving practitioners an understanding of the correlation between the cluster structure and the features.

3.2.6 Related works on self-organising maps

In this study, the use of SOMs forms an integral part of the final proposed anomaly detection framework. SOMs are an important field of study, and there are a large number of application areas and many research studies that can be found in the literature. Only a small number of examples is highlighted for illustration purposes, summarised in the form of tables in Table 3.1 through to Table 3.5. The research themes in these tables indicate the popularity of SOMs and show the wide variety of application areas for the approach.

The preceding sections introduce the concept of self-organising maps by explaining concepts such as the building, training, evaluation and interpreting of SOMs. These concepts are important in the context of this study and play a critical role in the final proposed anomaly detection framework.

Table 3.1: Examples of recent studies on the **medical** application of SOMs

Summary of related work in medical applications		
Author	Title	Research theme
Mazin <i>et al.</i> , (2021)	Identification of sarcomatoid differentiation in renal cell carcinoma by machine learning on multiparametric MRI	Utilising unsupervised self-organising maps and supervised learning vector quantizer (LVQ) machine learning to classify tumors in MRI images.
Abe <i>et al.</i> , (2020)	Batch-learning self-organising map identifies horizontal gene transfer candidates and their origins in entire genomes	As an alternative to phylogeny-based methods that rely on sequence alignments, the authors developed an alignment-free clustering method on the basis of an unsupervised neural network, batch-learning self-organising map (BLSOM).
Nilashi <i>et al.</i> , (2020)	Coronary heart disease diagnosis through the self-organising map and fuzzy support vector machine with incremental updates	The development of a predictive method for heart disease diagnosis using machine learning techniques.
Tervonen <i>et al.</i> , (2020)	Personalized mental stress detection with self-organising map: from laboratory to the field	To address the issues around using labelled data and poor model generalisation in stress detection applications, the research presents multiple personalised models, based on an unsupervised algorithm, the self-organising map.

Table 3.2: Examples of recent studies on the **industrial** application of SOMs

Summary of related work in industrial applications		
Author	Title	Research theme
Qi <i>et al.</i> , (2022)	Towards latent space based manipulation of elastic rods using autoencoder models and robust centerline extractions	A new methodology to manipulate elastic rods automatically into 2D desired shapes is presented. The methodology involves the use of a deep autoencoder, and leverages a self-organising map to compute the rod's centerline from raw visual data.
Lu and Yan (2020)	Deep Fisher autoencoder combined with a self-organising map for visual industrial process monitoring	This study proposes a Fisher autoencoder (FAE) to extract discriminative features for industrial process monitoring.
Bonelli-Toro and Gomez (2020)	Machine learning applied to acoustic emission for tool wear classification during milling of composite materials	In this paper, milling tools used in machining of sandwich structure panels for aerospace applications were studied.
Ramos <i>et al.</i> , (2019)	The forbidden region self-organising map neural network	A new self-organising map model which is guaranteed to keep all prototypes out of a set of prespecified forbidden regions, namely those regions of the input space where samples are not contained, is presented.

Table 3.3: Examples of recent studies on the **geographical** application of SOMs

Summary of related work in geographical applications		
Author	Title	Research theme
Athanasakis (2020)	Data-analysis in environmental and traffic data for Thessaloniki Greece	This study analyses traffic variables generated by Internet of Things devices. A self-organising map is used to illustrate the non-linear relationship between features and pollutants, and is also used for feature selection purposes.
Gardner <i>et al.</i> , (2020)	Self-organising map and relational perspective mapping for the accurate visualization of high-dimensional hyperspectral data	An optimisation of the toroidal self-organising map algorithm for the accurate visualisation of hyperspectral data is presented.
Gorgoglione <i>et al.</i> , (2020)	Application of the self-organising map to characterize nutrient urban runoff	A self-organising map method was used to simplify a complex dataset that contains precipitation, flow rate, and water-quality data. The model was used to identify possible patterns among these variables in order to help explain the main features that impact the processes of nutrient build-up and wash-off from urban areas.
Oliveira <i>et al.</i> , (2020)	Forecast opportunities for European summer climate ensemble predictions using self-organising maps	European weather patterns are analysed using a neural network-based classification of individual ensemble members.

Table 3.4: Examples of recent studies on **risk and anomaly detection** application of SOMs

Summary of related work in risk and anomaly detection applications		
Author	Title	Research theme
Chen <i>et al.</i> , (2020)	Self-organising map assisted deep autoencoding Gaussian mixture model for intrusion detection	A deep autoencoding Gaussian mixture model, including a compression and estimation network, is used for joint unsupervised training purposes. A self-organising map is then introduced to better preserve the topology of the input space.
Wang <i>et al.</i> , (2020a)	Imbalanced credit risk evaluation based on multiple sampling, multiple kernel fuzzy self-organising map and local accuracy ensemble	The proposal of a new ensemble model to evaluate class-imbalanced credit risk, which integrates multiple sampling, multiple kernel fuzzy self-organising map and local accuracy ensemble.
Chen <i>et al.</i> , (2019)	Regional disaster risk assessment of China based on self-organising map: clustering, visualization and ranking	Proposes a new multi-criteria decision making method to evaluate the natural disaster risk of China at the region scale.
Ferles <i>et al.</i> , (2018)	Denoising autoencoder self-organising map (DASOM)	Combining nonlinearities of neurons into networks for modelling increasingly varying and progressively more complex functions, using Boltzmann machines and denoising autoencoder self-organising map.

Table 3.5: Examples of recent studies on the **computing** application of SOMs

Summary of related work in computing applications		
Author	Title	Research theme
Aly and Almotairi (2020)	Deep convolutional self-organising map network for robust handwritten digit recognition	Development of a new efficient deep unsupervised network to learn invariant image representation from unlabelled visual data.
Mancini <i>et al.</i> , (2020)	XPySom: high-performance self-organising maps	The development of a high-performance self-organising map Python library, designed for optimal performance on multi-core CPUs and GPUs, and evaluated on the MNIST data set.
Notsu <i>et al.</i> , (2020)	Online state space generation by a growing self-organising map and differential learning for reinforcement learning	In this research, the authors developed a method integrating a growing self-organising map and differential learning system for online reinforcement learning which adaptively builds the state structure.
Lokesh <i>et al.</i> , (2019)	An automatic Tamil speech recognition system by using bidirectional recurrent neural network with self-organising map	Bidirectional recurrent neural network with self-organising map-based classification scheme is suggested for Tamil speech recognition.

Tables 3.1 to 3.5 illustrate the various applications of the modelling approach and provide a structured and summarised reference for further research in the area of SOM and related problems.

3.3 Social network analysis

In this section, an overview of social network analysis and the network metrics used in the study, as well as a discussion on attempts to use these network metrics in a fraud detection context are provided. Initially, an introduction to social network analysis is presented, which is followed by a reflection on the recent attempts at using network metrics for engineering fraud detection features. Several network metrics are then discussed, including centrality, clique, clustering, link analysis and assortativity metrics. Finally, the section is concluded, taking into account considerations of alternative approaches to determining network metrics, as well as examples of recent studies in the field of network analysis.

3.3.1 Introducing social network analysis

A social network is a concept referring to a structure made of social actors sharing interests or activities (Sanni and Nurcan, 2015). Social Network Analysis (SNA) also involves understanding the interdependencies in behaviour or attitudes related to configurations of social relations (O'Malley and Onnela, 2014). Both the relationships between individuals as well as their attributes are the observations of interest within a social network. Due to the connected nature of a social network, these observations are thought to be dependent on other observations as a result of the social connections within the network. The statistical techniques required to analyse the dependent data found in social networks, therefore, differ from traditional statistical techniques required for model-independent observations.

In SNA, a graph or network is formed by a finite set of nodes or vertices which are connected by a set of edges. In a real-world social network context, these nodes could represent users of a system, while the edges could represent the connections or relationships between specific users within the system. By convention, the number of nodes in the graph is denoted by n , while the number of edges in the graph is denoted by m . The graph itself is typically denoted as G .

A formal definition of a graph G is presented by Bender and Williamson (2010) as $G = (V, E)$ where V is a finite set, called the vertices of G and E is the set of edges. The edges are unordered pairs of vertices. An example of a simple graph is illustrated in Figure 3.4. There are a total of five vertices (or nodes) represented by the circles, and six edges represented by the letters indicating the relationships or links between the nodes.

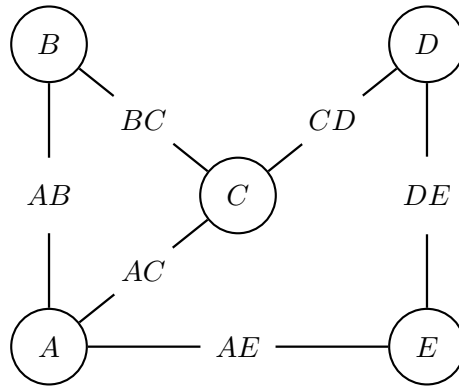


Figure 3.4: An example of a simple undirected graph

The edges between vertices can also contain a directional element to them, a concept known as a directed graph. A directed graph (or digraph) places importance on the order of the edge pairs. An example of a directed graph is displayed in Figure 3.5. The edge between nodes *A* and *B* flows in the direction from *A* to *B*, as indicated by the direction of the arrow. A directed graph can have edges that flow in both directions, as well as self-loops or edges from a node that flow back to the same node. The direction of a graph's edges is not always important, in which case the graph is formed as an undirected graph.

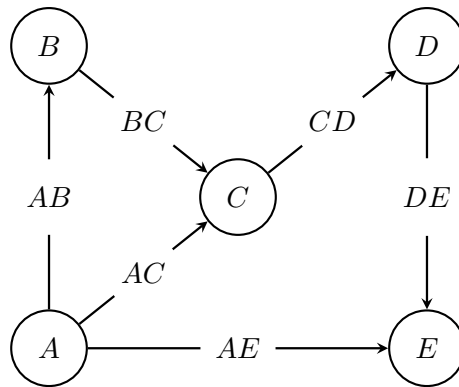


Figure 3.5: An example of a simple directed graph

Graphs can also take the form of a weighted graph or one where a numerical value is assigned to edges to represent the edges' importance within the graph. An example of a weighted graph is presented in Figure 3.6. The edges with the lighter, dashed lines represent a lower weighting in comparison to the thicker, bolder edge lines. The weight of an edge can represent several quantities, such as cost or distance

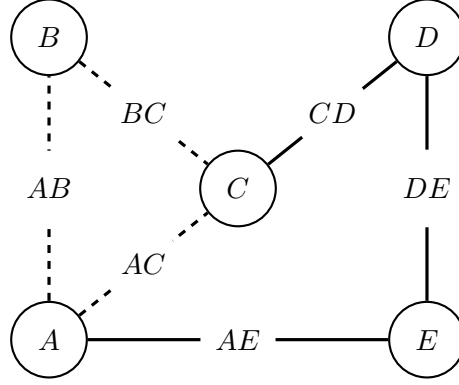


Figure 3.6: An example of a weighted graph

between nodes, or the capacity of the flow between them (Kang *et al.*, 2022).

Another important aspect of graphs, which is of particular interest to the study of fraud detection, is the concept of cliques. Given that $G = (V, E)$ is a simple undirected graph, a subgraph H of graph G is considered a clique if each node in the subgraph is connected by an edge to every other node in H (Newman, 2010). A subgraph H is said to be a maximal clique in G if H is a clique and no node exists only in G but not in H that sends an edge to every node of H .

Graph theory, and more specifically social network analysis, is a very broad field of study that consists of many sub-specialisations and application areas. It is a highly mathematical area of study, although a discussion of all of the mathematical details does not form part of the scope of this chapter. Readers interested in the mathematical foundations of graph theory are referred to Balakrishnan and Ranganathan (2012).

3.3.2 Motivation for network metrics in anomaly detection

Feature engineering is an important part of creating effective anomaly detection systems (Bahnsen *et al.*, 2016). Typical anomaly detection models only make use of raw transactional features and do not take into account the relationships between users within the transactional system. By expressing the relationships between users within a system as a graph, it is possible to determine the network metrics for each node within the graph. According to Negro (2021), the process of assigning a set of metrics to each node within a graph is called *featurisation*. This means that the graph data is essentially translated back into tabular form via the determination of network metrics.

Needham and Hodler (2021) refer to the process of creating features for machine learning using graph data as connected or graph feature engineering. The authors state that these connected features are features based on the structure of the graph data. These features have proved useful in expanding the universe of possible features for a model, reducing model overfitting, and significantly expanding the applicability of the models.

Another motivation for using connected features is that in scenarios that involve fraud, the activities of users are often hidden in several layers of obfuscation and network relationships. Traditional feature engineering approaches are likely to be unable to detect these behaviours without the contextual information that graphs provide (Needham and Hodler, 2021).

According to Hamilton (2020), graph features can be conducted in several ways. Node-level features relate to network metrics or other attributes that are attributed to each node within the larger graph, while graph-level features pertain to the attributes of the graph as a whole. Considering that this research intends to detect anomalies in a transactional setting, it makes sense to focus on node-level features that can help contribute to the anomaly detection decision.

Various attempts have been made to include network-based features when constructing predictive models in an anomaly detection context. Van Vlasselaer *et al.*, (2015) were successful in combining transactional features with network-based features in their *APATE* approach. The authors found that the two sets of features worked well together and complemented one another in improving the performance of their models. Bonilla (2018) also achieved superior performance of their models when including network metric features in their blockchain fraud detection research.

Fakhraei *et al.*, (2015) also showed success in combining network-based features with other feature types to improve the performance of spam account classification models, using methods, such as PageRank and degree centrality. These sentiments are echoed by Acevedo-Viloria *et al.*, (2021) who saw merit in combining traditional tabular data with alternative sources of data, such as graph data to improve their fraud detection strategy.

The field of network theory is vast and there are potentially hundreds of different network metrics that can be applied to a network. In the context of fraud detection, network metrics that relate to the influence that a node has on the graph, the location of the node in the network, and in what neighbourhoods within the graph the node is located, have proven to be useful (Bonilla, 2018). These concepts can be translated into several important network analysis typologies, namely density, centrality and modularity metrics. Focussing on these metrics helps to narrow down the multitude of metrics that could potentially be generated.

In terms of choosing on which network metrics to focus, related studies provide some assistance. Van Vlasselaer *et al.*, (2015) and Bonilla (2018) used centrality metrics successfully in detecting anomalies, and therefore it was decided to include centrality metrics in this study. Link analysis metrics, such as PageRank were effectively used by Fakhraei *et al.*, (2015), while Needham and Hodler (2021) determined that cliques are useful in determining groups. Assortativity metrics, according to Piraveenan *et al.*, (2008), were found to be strong indicators of anomalies. Finally, Yin *et al.*, (2019) argued that clustering metrics like the local clustering coefficient can be used as a powerful outlier detection signal, while Hamilton (2020) also motivates that the clustering coefficient is a significant measure of importance in graphs.

One of the objectives of this study is to establish a baseline of important network metrics to consider as possible significant features to use for anomaly detection problems. By identifying a set of network metrics that act as a strong baseline for anomaly detection purposes, researchers may be encouraged to include these significant network metrics in their anomaly detection feature sets.

3.3.3 Centrality

According to Borgatti *et al.*, (2009), centrality relates to the structural importance or prominence of a node in the network. Several centrality algorithms exist which can determine influence from the perspective of the node on the network. Popular examples include degree centrality, eigenvector centrality, closeness centrality, betweenness centrality, load centrality, and harmonic centrality.

Degree centrality

Degree centrality is the fraction of nodes to which a node is connected within a network. The values for degree centrality are normalised by dividing by the maximum possible degree in a simple $n - 1$ graph, where n is the number of nodes in a graph G . Grando *et al.*, (2018) define degree centrality for a vertex v as:

$$C_D(v) = \sum_{i=1}^n A(i, v), \quad (3.8)$$

where A is the adjacency matrix of graph G . Equation (3.8) essentially starts at row $i = 1$, and ending at the last row n , sums all of the values in the adjacency matrix designated by the possible (i, v) combinations. Figure 3.7 depicts an example of an adjacency matrix. The rows i and columns j of the adjacency matrix represent the connections in a simple graph. Each position in the matrix (v_i, v_j) contains a 0 or 1, depending on whether the nodes v_i and v_j are adjacent or not. A value

of 1 represents adjacent nodes, while a value of 0 indicates that the nodes are not adjacent.

$$\begin{pmatrix} 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

Figure 3.7: Example of an adjacency matrix

The degree centrality metric can be summarised as the number of ties that any node has to other nodes in a graph G . In practical terms, degree centrality caters for incoming ties or connections, as well as outgoing ties. Should the direction of the ties not be an important consideration, then the graph can be deemed undirectional, and therefore the degree centrality would be based on all related ties to the node in question.

Degree centrality can also be extended to the centrality of the graph as a whole (Newman, 2010), although for this research only node centrality is of interest. This is because nodes represent users within a graph, and the industry data set onto which the proposed network metrics will be augmented are aggregated at the user/node level.

A simple example of degree centrality is presented in Figure 3.8, where the degree centrality is indicated by node colour. In this example, the pink, red, blue, grey and orange nodes have degree centralities of 6, 3, 3, 2 and 1, respectively.

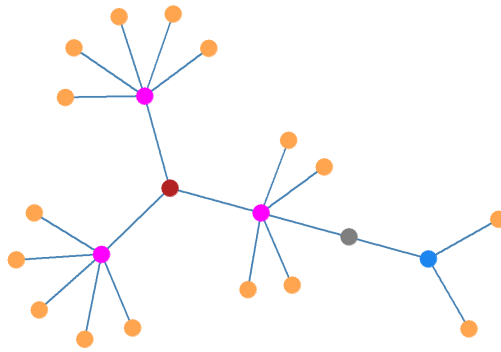


Figure 3.8: Simple network example of degree centrality (Lizardo, 2021)

A more complex graph structure to illustrate the degree centrality is presented in Figure 3.9.

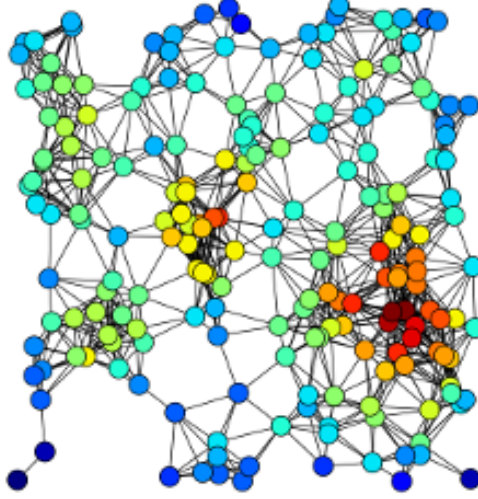


Figure 3.9: Complex network example of degree centrality (Rocchini, 2006)

The nodes with more connections to other nodes are redder in colour, whereas those nodes with a lower degree centrality are indicated by a darker shade of blue.

Eigenvector centrality

Eigenvector centrality is another measure of a node's influence within a given network. Newman (2010) states that eigenvector centrality differs from degree centrality by acknowledging that all connections to other nodes in a network are not equally important from an influential perspective. Eigenvector centrality is defined as follows:

$$x_v = \frac{1}{\lambda} \sum_{t \in G} A_{v,t} x_t, \quad (3.9)$$

where $A_{v,t}$ is an adjacency matrix, λ is a constant (typically the leading eigenvector of A (Hamilton, 2020)), and x_t is the centrality score for the node against which the node in question is measured. $A_{v,t} = 1$ in instances where node v and node t are connected, otherwise $A_{v,t} = 0$.

The premise of eigenvector centrality is that connections to nodes that are themselves influential contribute more to the centrality score of a node while being connected with less influential nodes lowers a node's centrality score. A high eigenvector

centrality score, therefore, means that the node in question is connected to many other nodes in the network that themselves have high scores. A complex network example of eigenvector centrality is presented in Figure 3.10.

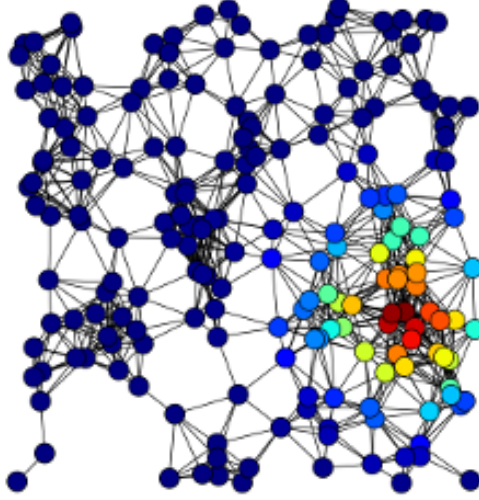


Figure 3.10: Complex network example of eigenvector centrality (Rocchini, 2006)

When compared with Figure 3.9, higher eigenvector centrality concentrates at the bottom right of the network in Figure 3.10. In comparison, the degree centrality is more widely distributed in Figure 3.9 even though it shares the main concentration location with Figure 3.10.

Closeness centrality

Closeness centrality measures the average length of the shortest path between the node in question and all other nodes in a graph G . Closeness centrality is calculated as follows (Newman, 2010):

$$C(x) = \frac{1}{\sum_y d(y, x)}. \quad (3.10)$$

The $d(y, x)$ component of the equation calculates the distance between the nodes y and x , and therefore the closeness centrality measures the reciprocal of the distance between the nodes. The normalised version of closeness centrality involves multiplying equation (3.10) by $n - 1$ where n is the number of nodes in graph G .

A simple network example of closeness centrality is presented in Figure 3.11, where node E, depicted in red, has the maximum closeness centrality.

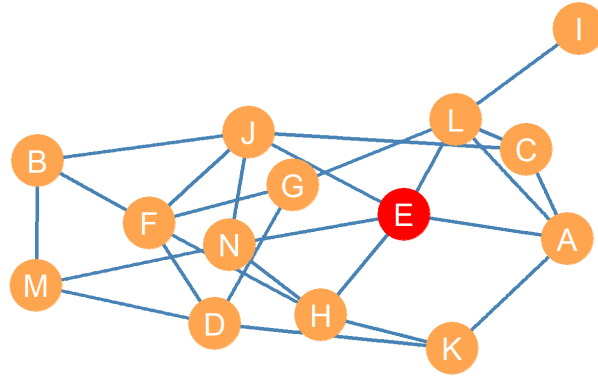


Figure 3.11: Simple network example of closeness centrality (Lizardo, 2021)

A second, more complex example of closeness centrality is presented in Figure 3.12.

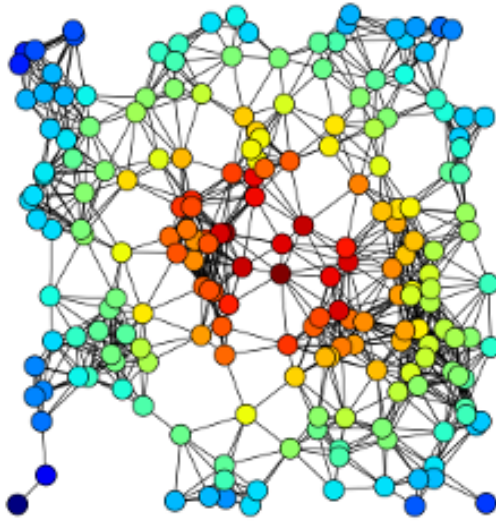


Figure 3.12: Complex network example of closeness centrality (Rocchini, 2006)

When comparing this example to Figures 3.9 and 3.10, the highest centrality values shift towards the centre of the network, thus illustrating how the relationship to all other nodes in the network influences this metric.

Betweenness centrality

Developed by Newman (2010), betweenness centrality is a node centrality measure that determines the number of times that a node acts as a bridge along the shortest path between two other nodes in the network. Nodes that are highly likely to occur on a randomly chosen shortest path between randomly chosen pairs of nodes will yield a high betweenness. The betweenness of a node in a graph G is defined as:

$$C_B(v) = \sum_{s \neq v \neq t \in V} \frac{p(s, v, t)}{p(s, t)}, \quad (3.11)$$

where $p(s, v, t)$ is the number of shortest paths from node s to node t that pass through node v , and $p(s, t)$ is the total number of shortest paths from node s to node t . This fraction is summed over all possible pairs of vertices (s, t) .

The computation involves computing the shortest path between each pair of nodes in the graph G , then determining the fraction of shortest paths that pass through the node in question, and finally summing this fraction over all pairs of nodes within graph G . A simple network example of betweenness centrality is presented in Figure 3.13, where node L, depicted in red, has the maximum betweenness centrality.

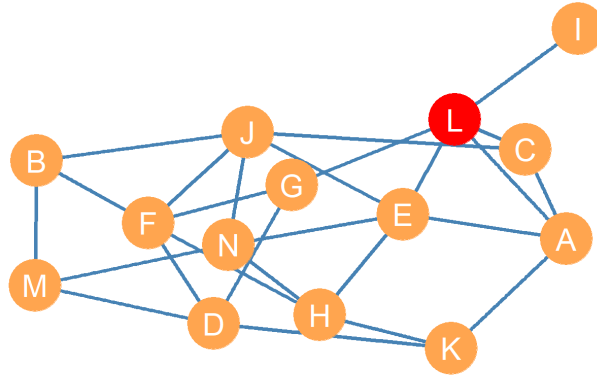


Figure 3.13: Simple network example of betweenness centrality (Lizardo, 2021)

A complex network example of betweenness centrality is displayed in Figure 3.14.

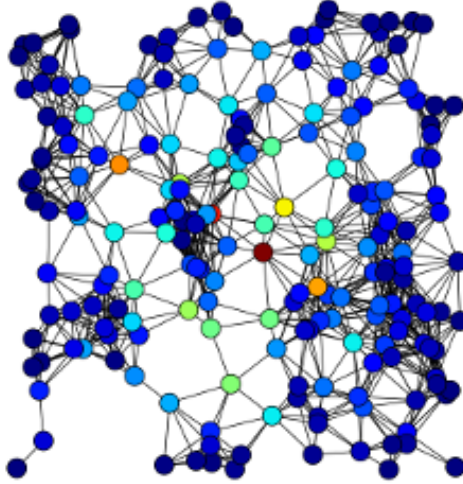


Figure 3.14: Complex network example of betweenness centrality (Rocchini, 2006)

The highest centrality values are positioned centrally in the network, but also positioned in a way that allows them to link the subgroups within the broader network.

Load centrality

This centrality metric was developed by Goh *et al.*, (2001) and is somewhat similar to the betweenness centrality measure introduced previously. The load centrality is the fraction of all shortest paths that pass through each respective node. Brandes (2008) found that load centrality is determined via the process whereby each node sends a unit amount of some commodity to each other node. After starting from a source node v , the commodity is always passed to the adjacent vertices closest to the target node. In instances where there is more than one node, then the commodity is divided up equally between the nodes. The sum of the commodity that passes through a node during these exchanges is then defined as the load centrality for node v .

Harmonic centrality

The harmonic centrality is a variant of closeness centrality depicted by equation (3.10) (Marchiori and Latora, 2000). Also known as valued centrality and invented

to solve the original problem with unconnected graphs, it can be considered a practical representation of an average shortest path (Needham and Hodler, 2021). The harmonic centrality equation is presented as follows:

$$H(v) = \sum_{u=1}^{n-1} \frac{1}{d(u, v)}, \quad (3.12)$$

where v is the node in question, u is the other node in the relationship, n is the number of nodes in the graph, and $d(u, v)$ is the shortest path distance between node v and node u . As with closeness centrality, harmonic centrality can also be normalised by dividing by $n - 1$ where n is the number of nodes in graph G . An example of harmonic centrality is shown in Figure 3.15.

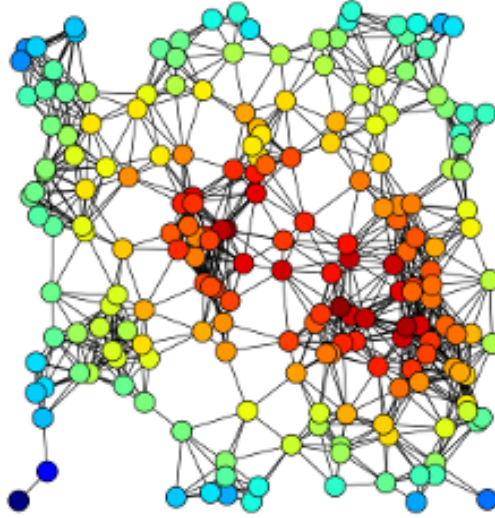


Figure 3.15: Complex network example of harmonic centrality (Rocchini, 2006)

The spread of the centrality values in Figure 3.15 appears similar to the closeness centrality values in Figure 3.12.

3.3.4 Clique

Assigning network participants into groups is an important process within SNA, as it is believed that individuals within a group are more alike than individuals in other groups (O'Malley and Onnela, 2014).

A clique is a complete graph, which is a graph where all nodes are connected. Newman (2010) states that a clique is a subset of nodes of a graph G , where every

two distinct nodes in the clique are adjacent. Therefore, the larger the clique, the stronger the evidence that all the connected individuals are in the same group. An example of a clique is presented in Figure 3.16.

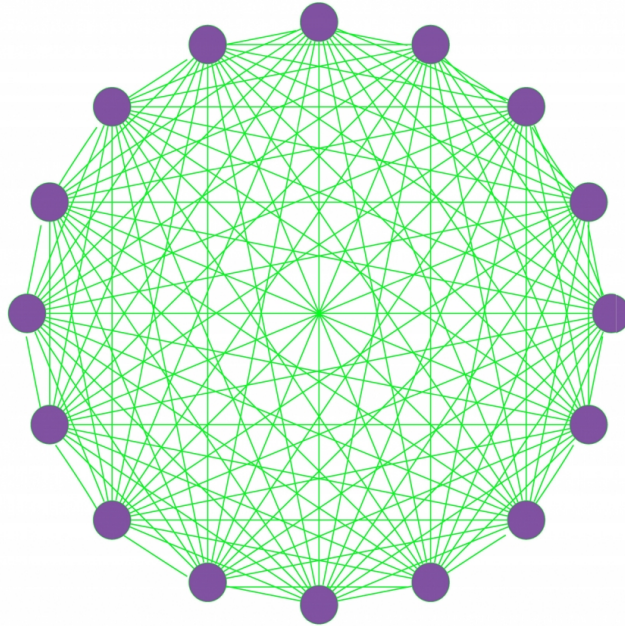


Figure 3.16: Example of a clique (Barabasi, 2016)

The clique in Figure 3.16 has 16 individuals or nodes, where each node is connected to every other node within the network. Extending the concept of a clique, a maximal clique is a clique that cannot be extended by including one more adjacent node.

Node clique number

In social network analysis terms, it can be concluded that nodes within a clique are connected to all other nodes. The node clique number metric finds the size of the largest maximal clique containing each respective node.

Number of cliques

The number of cliques metric finds the number of maximal cliques for each respective node. Applications typically require reporting on all maximal cliques, rather than just on a single large clique. Algorithms attempting to solve this would take exponential

time in the worst-case scenario, as graphs could contain an exponential number of cliques.

In practice, however, sparser graph structures are encountered, therefore algorithms may solve this in linear time (Eppstein *et al.*, 2013). An example of the number of cliques is displayed in Figure 3.17.

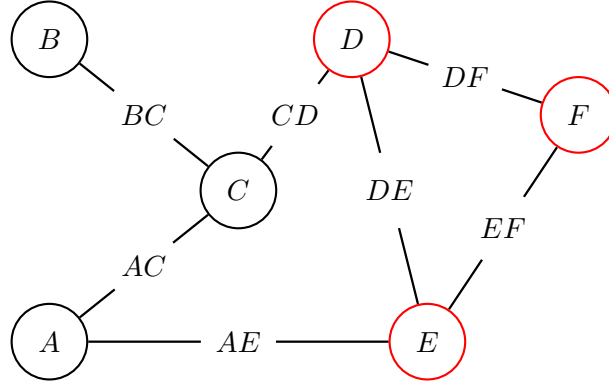


Figure 3.17: Example of the number of cliques

Figure 3.17 shows a graph with a single maximum clique, which is the red triangle $\{D, E, F\}$. There are also four more maximal cliques, which are the pairs $\{A, E\}$, $\{A, C\}$, $\{C, D\}$, and $\{B, C\}$.

3.3.5 Clustering

Clustering is the process of grouping sets of nodes in such a way that nodes in the same group are more similar to each other than to those in other groups. In network analysis, the term transitivity refers to an important property of the potential relations between nodes. In a network, there are various relations between pairs of nodes, the simplest of which is a group of nodes being connected by an edge. If the connected relation were transitive it would mean that if node u is connected to node v , and v is connected to w , then u is also connected to w (Newman, 2010).

Another important concept required to understand clustering metrics is the idea of a triangle. A triangle is a set of nodes where two of the nodes are connected by an edge of the graph. A triple, on the other hand, is a set of two edges connected to a node v .

Clustering coefficient

The clustering coefficient, in the context of unweighted graphs as used in this study, determines the number of triangles connected to a node v as a proportion of the total number of triples centred around node v . Onnela *et al.*, (2005) define the clustering coefficient as:

$$c_v = \frac{2t_v}{k_v(k_v - 1)}, \quad (3.13)$$

where k_v is the degree of node v and t_v is the number of triangles attached to the node.

The intention of this measure is based on the observation that nodes within social networks tend to create tightly knit groups with a high number of connections between the nodes within each group (Newman, 2010).

Figure 3.18 provides a visual explanation of the local clustering coefficient on an undirected graph. The local clustering coefficient of the red node is computed as the proportion of connections among its neighbours which are realised compared with the number of all possible connections.

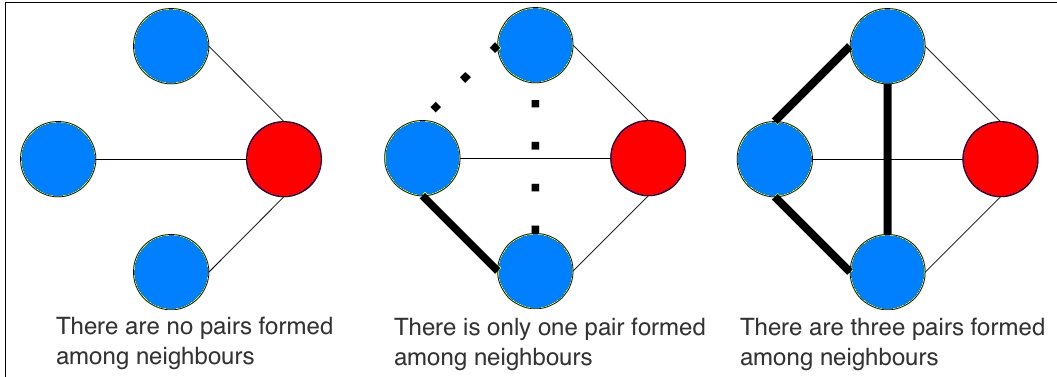


Figure 3.18: Example of clustering coefficient (Phuong *et al.*, 2017)

Negro (2021) suggests that a clustering coefficient could be important in a fraud detection context. If a node were to have a low clustering coefficient compared with the global average clustering coefficient, then that means that the node is connected to people who belong to independent groups. This anomalous activity of joining independent groups may reveal some intention to join an established grouping currently involved in fraudulent activity.

Square clustering coefficient

The square clustering coefficient determines the probability that two neighbours of node v share a common neighbour that differs from node v (Lind *et al.*, 2005). It is defined by:

$$C_{vu} = \frac{\sum_{u=1}^{k_v} \sum_{w=u+1}^{k_v} q_v(u, w)}{\sum_{u=1}^{k_v} \sum_{w=u+1}^{k_v} [a_v(u, w) + q_v(u, w)]}, \quad (3.14)$$

where

$$a_v(u, w) = (k_u - (1 + q_v(u, w) + \theta_{uv}))(k_w - (1 + q_v(u, w) + \theta_{uw})). \quad (3.15)$$

$q_v(u, w)$ are the number of common neighbours of u and w other than v . $q_v(u, w) = 1$ if the neighbours are connected, and 0 otherwise. θ_{uv} has the opposite value of $q_v(u, w)$, and k_u is the degree of node u (Zhang *et al.*, 2008).

3.3.6 Link Analysis

Link analysis is a data analysis technique used to measure relationships between nodes. One example is the Katz centrality metric, which measures influence by taking into account the total number of walks between pairs of nodes (Salehi and Masoumi, 2020).

A shortcoming of the Katz centrality metric, however, is that if a node with high Katz centrality links to many other nodes, then those others also get high centrality (Newman, 2010). Due to this shortcoming, the PageRank algorithm is often preferred.

PageRank

PageRank is a link analysis algorithm popularised by its extensive use at Google for ranking web pages in their search engine results. According to Page *et al.*, (1999), PageRank assigns a numerical weighting to each node within a connected set of nodes. The algorithm computes a ranking of the nodes in a graph G , based on the structure of the connections. The PageRank value for any node within a graph G is expressed as:

$$PR(u) = \sum_{v \in B_u} \frac{PR(v)}{L(v)}, \quad (3.16)$$

where the PageRank value for node u is dependent on the PageRank values for each node v contained within the set B_u of all nodes linked to the node in question. This is then divided by $L(v)$, the number of connections from node v . Figure 3.19 illustrates a simple network with nodes labelled by their PageRank centrality.

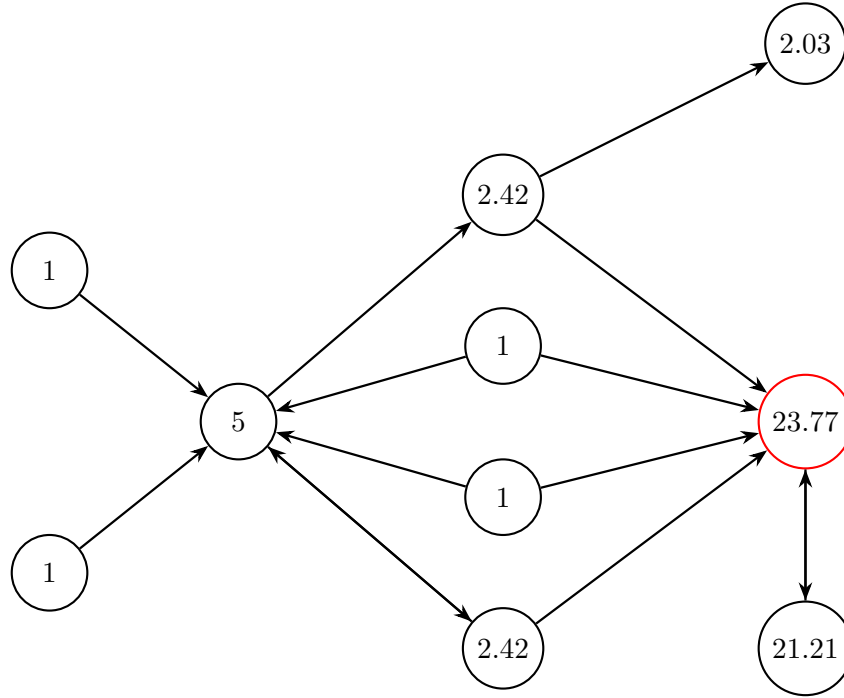


Figure 3.19: Simple network example of PageRank centrality

The nodes in Figure 3.19 display their respective PageRank values, where the most influential node in the network has a PageRank value of 23.77 (red outline). The second highest value of 21.21 belongs to the node below the largest node. There is a bidirectional relationship between the largest and second-largest nodes in the network.

3.3.7 Assortativity

Assortativity refers to a network where a significant fraction of the edges runs between vertices of the same type (Newman, 2010). A simplistic way of quantifying

the assortativity would be to measure this fraction, although this is not useful in non-trivial cases. A useful assortativity metric is, therefore, the average neighbour degree.

Average neighbour degree

Average neighbour degree, which according to Barrat *et al.*, (2004) determines the average degree of the neighbourhood for each respective node and can be determined by:

$$k_{nn,i} = \frac{1}{|N(i)|} \sum_{j \in N(i)} k_j, \quad (3.17)$$

where $N(i)$ are the neighbours of node i and k_j is the degree of node j from within $N(i)$.

3.3.8 Alternative approaches to determining network metrics

Beyond the traditional approaches to calculating network metrics that have been covered in this chapter, alternative methods have been proposed. One emerging method is via the use of Graph Neural Networks (GNN), which are a type of neural network which operates directly on graphs (Maurya *et al.*, 2019). While alternative approaches to determining networks have emerged, the focus of this study will be on calculating network metrics by using the social network analysis techniques presented in this chapter.

3.3.9 Related works on social network analysis

It is necessary to acknowledge related studies as part of this study, therefore additional studies related to using network analysis are summarised in Table 3.6 to Table 3.12. Social network analysis is a significant input, with a wide variety of applications. A portion of the example application areas from the literature is highlighted, including applications related to healthcare, fraud detection, geographical, organisational, survey studies, networking, and computing. The applications of social network analysis to anomaly detection are of particular interest, as they confirm that analysing data in a network can help to detect suspicious activity.

Table 3.6: Examples of recent studies on the **healthcare** application of SNA

Summary of related work in healthcare applications		
Author	Title	Research theme
Foutch <i>et al.</i> , (2021)	Protein conformational switch discerned via network centrality properties	The authors used four different network centrality metrics (betweenness, closeness, degree and eigenvector centralities) to study protein structure networks (PSNs).
Jeyasudha and Usha (2021)	An intelligent centrality measures for influential node detection in COVID-19 environment	This research proposes a method to identify influential nodes in social networks. The method involves detecting communities using community detection algorithms and finding the influential nodes in the community using social network analysis centrality measures.
Hamilton <i>et al.</i> , (2020)	Spiked-based graph centrality measures	This study derived several spike-based routines that compute or establish bounds on radial centrality measures for undirected graphs and trees without the use of matrix multiplication.
Hung <i>et al.</i> , (2020)	Social network analysis of COVID-19 sentiments: application of artificial intelligence	Social network analysis was used to determine the social network of dominant topics and whether tweets related to COVID-19 expressed positive, neutral, or negative sentiments.

Table 3.7: Examples of recent studies on the **fraud detection** application of SNA

Summary of related work in fraud detection applications		
Author	Title	Research theme
Bodaghi and Teimourpour (2018)	Automobile insurance fraud detection using social network analysis	Organised fraudsters were claimed to have caused significant revenue leaks for the insurance industry. Social network analysis was used to represent the relations between entities, with cycle detection being employed to locate suspicious activity.
Leite <i>et al.</i> , (2018)	Visual analytics for event detection: focusing on fraud	Social network analysis is shown to extend fraud detection techniques through a visual analytics approach. The authors confirm that social network analysis is useful in a fraud detection context, as it allows for the analysis of hidden network connections, and saves time in comparison with manual fraud investigations.
Colladon and Remondi (2017)	Using social network analysis to prevent money laundering	The financial operations of an Italian factoring company were analysed, with social network analysis being applied to detect money laundering.

Table 3.8: Examples of recent studies on the **geographical** application of SNA

Summary of related work in geographical applications		
Author	Title	Research theme
Napitupulu <i>et al.</i> , (2021)	Centrality measures in transportation networks for Unpad campus route	The authors conducted a study to understand which centrality measures were most important in a transport network. This included conducting a numerical experiment using Unpad campus route data to determine which centrality measures were most effective in modelling the network.
Bhogaram <i>et al.</i> , (2020)	Optimal and critical path analysis of state transportation network using Neo4j	A network model, created using the Neo4j software, mapped the Southern Californian transportation network. Critical nodes in the network were determined using centrality algorithms.
Valeri and Baggio (2020)	Social network analysis: organizational implications in tourism management	The authors conducted a study on how social network analysis has been used and can be used to improve the competitiveness of tourism destinations. Social network analysis was used to demonstrate that tourism destination is an adaptive socio-economic system.

Table 3.9: Examples of recent studies on the **finance and organisational** application of SNA

Summary of related work in finance and organisational applications		
Author	Title	Research theme
Goz <i>et al.</i> , (2021)	Effect of centrality measures for keyword extraction from Turkish documents	The performance of graph-centrality measures as initial node weights in graph-based keyword extraction of Turkish banking documents was investigated. The results indicated that eigenvector centrality was the most important measure. The focus was on the degree, eigenvector, betweenness and closeness centrality measures.
Bouchet <i>et al.</i> , (2020)	CEO network centrality and corporate social responsibility	The impact of CEO social network centrality on corporate social responsibility was studied.
Das <i>et al.</i> , (2020)	Influential nodes in social networks: centrality measures	This work explores how people are connected via social networks, including a discussion of a variety of popular applications for social network analysis.
Llantos <i>et al.</i> , (2019)	Characterizing instructional leader interactions in a social learning management system using social network analysis	By applying social network analysis to the interaction logs of an LMS system, the researchers aimed to answer the question as to whether instructional leadership can be manifested in social system interactions.

Table 3.10: Examples of recent **survey studies** of SNA

Summary of related survey studies		
Author	Title	Research theme
Popereshnyaki and Yurchuk (2021)	Social networks: analysis, algorithms and their implementation	A survey on the applications of social network analysis techniques was conducted, to establish the relationship between graph theory and social network analysis.
Saqr <i>et al.</i> , (2020)	Capturing the participation and social dimensions of computer-supported collaborative learning through social network analysis: which method and measures matter?	This research presents findings on the study of various centrality measures frequently used in social network analysis.
Will <i>et al.</i> , (2020)	Combining social network analysis and agent-based modelling to explore dynamics of human interaction: a review	In this paper, an investigation as to whether agent-based modelling and social network analysis can be coupled, is conducted. The results showed that the consideration of the concept of social integration through networks was an important aspect of the study.
Can and Alatas (2019)	A new direction in social network analysis: online social network analysis problems and applications	A review of popular social network analysis, including applications and references for researchers focused on analysing social network data.

Table 3.11: Examples of recent studies on the **networking** application of SNA

Summary of related work in networking applications		
Author	Title	Research theme
Ahmad <i>et al.</i> , (2020)	Social network analysis based localization technique with clustered closeness centrality for 3D wireless sensor networks	The authors propose a novel wireless localisation technique based on social network analysis, to study the different properties of networks as a graph.
Chen and Huang (2020)	A novel centrality for finding key persons in a social network by the bi-directional influence map	A novel centrality measure to find key nodes within a social network by a proposed synthesised index which accounts for both the inflow and outflow matrices of a vertex is presented.
Colladon and Naldi (2020)	Distinctiveness centrality in social networks	Five new measures of distinctiveness centrality are proposed. The new metrics attribute a higher score to nodes keeping a connection with the periphery of the network.
Dias <i>et al.</i> , (2020)	Network centrality analysis in management and accounting sciences	A social network was developed based on the co-authorship of academic papers published in leading Brazilian journals. Network metrics were calculated to determine the probability of authors occupying prominent positions in the network.

Table 3.12: Examples of recent studies on the **computing** application of SNA

Summary of related work in computing applications		
Author	Title	Research theme
Behera <i>et al.</i> , (2020)	MR-IBC: MapReduce-based incremental betweenness centrality in large-scale complex networks	In this paper, a MapReduce-based incremental parallel algorithm for exploring the influential nodes based on the betweenness centrality is proposed.
Takan (2020)	Creation of mutants by using centrality criteria in social network analysis	Centrality metrics from social network analysis were used in a new mutant reduction method, in order to improve mutation testing in evaluating the effectiveness of hardware and software tests. The results showed that mutants selected via the proposed technique revealed improved performance.
Jamali and Mitchell (2018)	Centrality-based improvements to CDCL heuristics	Modifications to modern decision and clause-deletion heuristics that exploit formula structure by using various centrality measures were explored. Specifically, centrality metrics were used in clause deletion, and led to solving nine more formulas for the 2017 main track.

Tables 3.6 to 3.12 portray the various domains in which SNA is applied. These summaries all highlight how SNA can be used to determine the influence that nodes have within a network, which is a beneficial property when attempting to detect suspicious activity.

3.4 Chapter conclusion

This chapter is the second and concluding chapter of Part I (Literature review) of the study. The chapter provided an overview of recent literature on self-organising maps and social network analysis. This began with discussions related to building and training self-organising, as well as using a trained self-organising map model for visualisation purposes. Finally, discussions related to social network analysis and how metrics can be calculated from a graph were presented.

In the next chapter, which is the first chapter in Part II of the study, the creation of an effective method for anomaly detection in a transactional setting to ultimately apply the approach in a real-world setting will be addressed.

Part II

Development and illustrative applications

Chapter 4

Autoencoders for anomaly detection in a transaction-based system

4.1 Chapter introduction

In this chapter, which is the only chapter in Part II of the study, the aim is to report on the creation of an effective method for anomaly detection that can be applied in a real-world context and incorporated into a machine learning-based framework for anomaly detection. The purpose of Part II is to present a discussion on the development and illustrative applications of autoencoders using a public data set. The chapter serves as a motivational argument for the autoencoder-based anomaly detection model's modelling prerequisites that will be applied to a real-world case study (see Chapter 5). The principles and techniques described and validated in this chapter will also be included in the final proposed machine learning-based framework for anomaly detection (see Chapter 8).

Building a neural network-based anomaly detection capability requires effective implementation of a chosen algorithm and should also include processes relating to the usability of the capability within an operational environment. Unsupervised learning techniques have proven to be effective for detecting anomalies (Phua *et al.*, 2010) although their usage contains some uncertainties regarding the choice of an optimal number of clusters in the case of clustering (Sugar and James, 2003) or the choice of a suitable threshold in the case of an autoencoder (Khan and Taati, 2017). An approach is required that supports the experimental nature of unsupervised learning and caters for an organisation's structured needs.

A proposed unified approach is the core of this chapter. The approach begins with the development of a sparse autoencoder for detecting anomalies. The choice of structure for the sparse autoencoder architecture is justified using an implementation of a neural architecture search; that is, multiple candidate architectures are simulated, with the simulated architecture that produces the best evaluation metric being selected. The determination of a suitable threshold for classifying anomalies is generally subjective, and this chapter also proposes an approach to finding the optimal threshold. Following the application of the optimal threshold, the sparse autoencoder produces an anomaly score per instance. These are determined by the mean square error (MSE) and are used to determine how anomalous an instance might be. The larger the anomaly score, the more likely it will be that the instance is anomalous.

However, a challenge with these anomaly scores is that they are difficult to interpret, beyond the fact that higher means more anomalous (Kriegel *et al.*, 2011). A Gaussian scaling function is proposed to transform the anomaly scores into a distribution ranging from zero to one, a form that is comparable across different models and understood universally. These transformed scores can therefore be considered the probability that the instance is anomalous.

The proposed unified approach is thus comprised of three steps *i.e.*, neural architecture search, threshold optimisation, and Gaussian scaling steps. These three steps are presented in three different sections, namely Sections 4.3, 4.4, and 4.5.

According to Sommer and Paxson (2010), anomaly detection projects often fail to be deployed in an operational setting due to issues around a lack of interpretability and effective evaluation. The proposed unified approach will provide practitioners with a means of constructing a performant architecture, as evaluated against multiple preferred performance metrics for highly imbalanced classes, and producing an output in the form of a probability score.

To ascertain whether the aforementioned unified approach is a viable option for anomaly detection, the approach should be validated on a relevant data set. Considering that the approach is designed for use in an anomaly detection setting, any data used to illustrate the concept needs to contain a severe class imbalance, with one majority and one minority class. The intention of using data for illustrative purposes is to demonstrate that the unified approach is effective in detecting anomalies and that the approach could be extended to real-world data and ultimately built into a machine learning-based framework for anomaly detection.

The remainder of this chapter is organised as follows: in Section 4.2, the data set used for experimental research is presented. The implementation of the neural architecture search, threshold optimisation, and Gaussian scaling are then illustrated

in Sections 4.3, 4.4 and 4.5, respectively, followed by a performance evaluation of the optimal architecture in Section 4.6. The results of the approach are then discussed in Section 4.7, while Section 4.8 provides an extension of the proposed approach. Finally, conclusions about the unified approach are drawn in Section 4.9.

4.2 Description of illustration data

Data related to anomaly detection is notoriously difficult to obtain according to Phua *et al.*, (2010). Organisations are traditionally somewhat conservative when it comes to publishing data related to cases of in-house anomalies. The reasons for this could be that exposing their internal anomalies might pose a reputational risk, or the data may present an opportunity for prospective bad actors to capitalise on potential perceived gaps in the system.

The unified approach experiment was conducted using a publicly available anomaly data set that has been cited in the literature (Dal Pozzolo, 2015). This particular data set’s use is intended to illustrate the concepts for this experiment, but in follow-up applications in Chapters 6 and 8, the intention is to use real-world applications. The public data set was sourced from Kaggle, a website dedicated to hosting machine learning competitions (Kaggle, 2017), and was collected by Worldline S.A.¹, a European payments provider.

The data contains transactions from September 2013 entered into over two days. It is a European user credit card data set and contains 284 807 transactions of which only 492 (or 0.172%) are confirmed as fraudulent. The data set is labelled and can therefore be used for either classification or anomaly detection purposes. Principal Component Analysis (PCA) has been performed on most of the features, with the resultant data set containing 28 unknown features. There is therefore no way in which to infer identities or additional user information from them.

Two additional features have not been processed via PCA, namely the time and amount of the transaction. The labels consist of 1s and 0s, with 1 indicating a confirmed case of fraud and 0 indicating a genuine transaction. Although the data used for this study is transactional, the proposed unified approach can be extended to any anomaly detection application.

4.2.1 Latent space representation using t-SNE

To get a sense of the structure of the data set, one would want to try and visualise the data to ascertain whether a separation between the classes is possible. This presents a problem when the dimensionality of the data is high, as humans are generally

¹<https://worldline.com/>

unable to visualise data beyond three dimensions. One method for visualising the data would be to employ a technique that compresses data from a high-dimensional space down into a lower-dimensional space.

According to Maaten and Hinton (2008), t-Distributed Stochastic Neighbor Embedding (t-SNE) achieves this by converting similarities between data points to joint probabilities and attempts to minimise the KL divergence (see Section 2.11.2 Chapter 2) between the joint probabilities and the low-dimensional embedding and the high-dimensional data. The algorithm for a standard implementation of t-SNE as devised by Maaten and Hinton (2008) is presented in Algorithm 4.1.

By compressing the credit card data down into two dimensions using t-SNE, and mapping the respective class labels onto the compressed data, it would become apparent whether the classes are indeed separable.

Algorithm 4.1: Simple version of t-Distributed Stochastic Neighbor Embedding (Maaten and Hinton, 2008)

input : dataset $X = \{x_1, \dots, x_n\}$, cost function parameters: perplexity $Perp$, optimisation parameters: number of iterations T , learning rate η , momentum $\alpha(t)$
output: low-dimensional data representation $Y^{(T)} = \{y_1, \dots, y_n\}$
begin
 compute pairwise affinities $p_{j|i}$ with perplexity $Perp$;
 set $p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n}$;
 sample initial solution $Y^{(0)} = \{y_1, \dots, y_n\}$ from $N(0, 10^{-4}I)$;
 for $t = 1$ **to** T **do**
 compute low-dimensional affinities q_{ij} ;
 compute gradient $\frac{\delta C}{\delta Y}$;
 set $Y^{(t)} = Y^{(t-1)} + \eta \frac{\delta C}{\delta Y} + \alpha(t)(Y^{(t-1)} - Y^{(t-2)})$;
 end
end

In Figure 4.1, the implementation of t-SNE has reduced the feature space into several clusters, with the normal data points coloured in green and the fraudulent data points coloured in red. Although not completely separated, most of the fraudulent transactions appear within two distinct clusters.

The t-SNE results further justify dedicating additional effort towards using this data set in which a separation between its classes is possible. Due to the imbalanced nature of the data set, the normal transactions severely outnumber the fraudulent transactions, therefore yielding a far greater number of normal clusters.

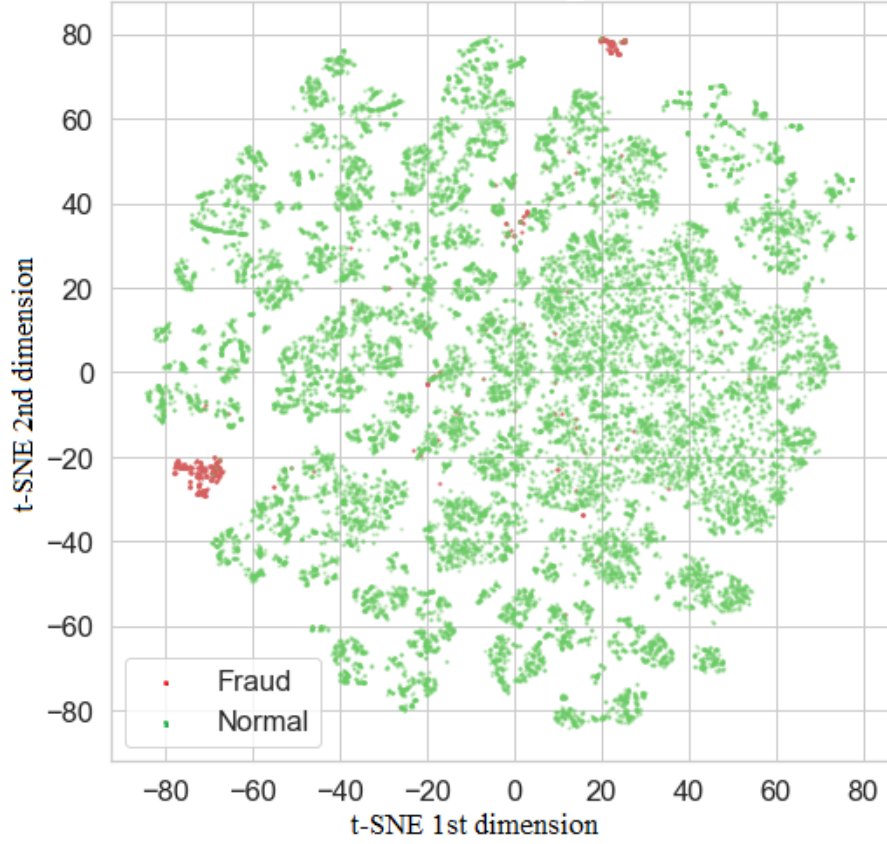


Figure 4.1: Visualising latent space of credit card data using t-SNE

The representation in Figure 4.1 was achieved by using a perplexity $Perp$ of 30, and a learning rate η of 200. According to Maaten and Hinton (2008), typical perplexity values lie within the 5 through 50 range. On the credit card data set, a perplexity value of 30 shows a desirable level of separation.

4.2.2 Sampling approach

In preventing sampling bias, a robust sampling approach needs to be applied. Such an approach needs to cater for the limited number of fraudulent instances in the data set. The proposed sampling approach starts by taking the 284 807 transactions and shuffling them so that they are in random order.

The set is then split into a training set (90%) and a test set (10%). This splitting operation includes stratified sampling on the target variable to ensure that both sets have normal and fraudulent transactions and that they are representative of the data. The test set is kept aside, while the training set is used for further processing.

The newly created training set is then shuffled again and is split into two new sets, a training set made from 90% of the previously created training set, and a validation set made from the remaining 10% of the previously created training set. Again these are created by using stratified sampling on the target variable, which evens out the split between fraudulent and normal transactions about their proportion.

The fraudulent transactions from the second newly created training set are then removed so that the training set contains only normal transactions, whereas the validation set keeps both its normal and fraudulent transactions. The fraudulent transactions are not required in the training set as the learning process is unsupervised, and the purpose of the autoencoder's learning process is to learn normal behaviour only. The best architectures can then be determined by using both of these training and validation sets. Once the best architecture has been determined, the optimal architecture is evaluated against the test set that was held out previously to determine how well the model generalises on unseen data.

Those transactions that were originally removed from the training set are then added back to the validation set so that there are more fraudulent records in the validation set. This also prevents any wastage of fraudulent transactions. The final result of the sampling process is that the validation set ends up with 443 fraudulent transactions, whereas the test set ends up with 49 fraudulent transactions. Together these make up the initial 492 fraudulent transactions. The sampling process will be revisited and illustrated in more detail when the proposed approach is applied in subsequent chapters.

Since the validation set has almost 10 times the number of fraudulent transactions, it is expected that the performance metrics that are more sensitive to imbalanced classes would yield significantly different results when measuring the performance on the validation set versus that of the testing set. Flach (2019) suggests that using a single measure to evaluate the performance of a machine learning model is inadequate, and therefore multiple measures should be considered. As determined in Chapter 2, there is a variety of performance metrics that can be used in an anomaly detection context. While the proposed balanced score approach will be used for both validation and testing, directly comparing the validation balanced scores with the testing validation scores is not beneficial, considering the proportional difference in the class imbalance between the validation and test sets.

4.3 Neural architecture search

The autoencoder model discussed in this section follows from the discussion in Chapter 2. In this section, Neural Architecture Search (NAS) is applied to the credit card fraud detection data set presented in Section 4.2.

A common method of optimising a model’s hyperparameters is through a grid search, where an exhaustive search through a specified subset of an algorithm’s learning space is conducted. A randomised approach to hyperparameter optimisation is more efficient when compared to a grid search and manual search approaches (Bergstra and Bengio, 2012). These approaches all relate to the typical hyperparameters available for tuning during training time, including hyperparameters, such as the learning rate, dropout rate, and batch size. Automating the hyperparameter optimisation process can be considered a form of automated machine learning (AutoML), where AutoML is the process of automating the application of machine learning for real-world applications (Feurer and Hutter, 2019).

NAS is the process of automatically engineering neural network architectures. It is a subset of the field of AutoML, and assists with determining the wide range of hyperparameter choices about a deep neural network’s architecture, regularisation, and optimisation (Feurer and Hutter, 2019). According to Elsken (2019), there are three dimensions to NAS. They are defining the search space, the search strategy, and the performance estimation strategy. The search space relates to the neural architectures that will be simulated; the search strategy relates to how the search space is intended to be explored, while the performance estimation strategy relates to how the performance of each architecture is measured.

Within the proposed unified approach, the NAS implementation simulates a population of candidate architectures, from which the optimal architecture is determined. In general, researchers typically use a single metric to evaluate the performance of an architecture. Within the unified approach, it is proposed that an average of the Area Under the Curve (AUC) (Hancock and Khoshgoftaar, 2021; Chawla, 2009), the Average Precision (AP) (Mekterovic *et al.*, 2021; Fan and Zhu, 2011), and a normalised Matthews Correlation Coefficient (MCC) (Chicco and Jurman, 2020; Raghavan and El Gayar, 2019) are used to derive a balanced score that can be used to select the best performing architecture.

Each candidate architecture is created, by assigning it a random depth and a random number of nodes, with an upper bound set to the size of the input layer, for each layer of the corresponding depth. For the experiment, each architecture has the same regularisation constraint placed on it. This implementation is presented as Algorithm 4.2.

Algorithm 4.2: Neural Architecture Search algorithm

input : number of n training epochs, intended population size m of
generated architectures, size of the input layer l
output: an array $S = \{s_1, \dots, s_m\}$ containing the balanced scores of the m
simulated architectures

for $i = 1$ **to** m **do**
 generate random depth d ;
 generate random architecture A , with a node upper bound l and length
 d ;
 for $j = 1$ **to** d **do**
 if $A[j] < min$ **then**
 $min \leftarrow A[j]$;
 end
 end
 remove min from A ;
 encArchitecture $\leftarrow A$ sorted in descending order;
 decArchitecture $\leftarrow A$ sorted in ascending order;
 build input layer of size l ;
 for $k = 1$ **to** $d - 1$ **do**
 build encoder layer with encArchitecture[k] nodes;
 end
 build bottleneck with min nodes;
 for $n = 1$ **to** $d - 1$ **do**
 build decoder layer with decArchitecture[n] nodes;
 end
 build output layer of size l ;
 train the autoencoder;
 calculate the balanced score s_i using validation data;
 $S \leftarrow s_i$;
end
output S ;

The search space for this experiment is chain-structured densely connected neural network architectures, created through the assignment of a randomly generated length and the number of nodes per architectural element. The search strategy is deduced by simulating the search for m iterations. Finally, the performance estimation strategy is defined by the balanced score mentioned above.

4.4 Threshold optimisation

When computing the balanced score in Algorithm 4.2, the AUC and AP metrics can be calculated directly from the raw anomaly score as produced by the autoencoder. For the MCC, however, a binary outcome is needed. That one may determine these binary outcomes, a threshold for the raw anomaly scores is required to split the scores into a binary classification outcome.

Determining the optimal threshold has traditionally been a subjective and manual process. The process of determining an optimal threshold is a vital step in the modelling process, where a poorly chosen threshold would lead to unfavourable performance (Choi *et al.*, 2019). The proportion of anomalies in their training set was used as the threshold for classification purposes. The authors also noted that additional research needs to be done to determine systematic approaches for investigating threshold values, of which one approach is presented below.

Autoencoder-based anomaly detection is a deviation-based anomaly detection method that makes use of semi-supervised learning (An and Cho, 2015). A few different approaches for determining the optimal threshold for an autoencoder have been proposed in the literature. Cao *et al.*, (2016) used a kernel density estimation approach on the compressed data within the hidden layer. Dau *et al.*, (2014) used a different approach. Assuming that the model was trained on normal data only, the threshold was selected by using the largest reconstruction error for the normal data reconstructions. A stochastic anomaly threshold that uses accuracy as the performance metric for optimisation purposes has also been used (Aygun and Yavuz, 2017), although as discussed in Chapter 2, accuracy is not an effective metric for evaluating the performance of a model with a very high-class imbalance.

The process of determining the optimal threshold can be determined through the simulation of a range of possible threshold (MSE) values to determine each threshold's respective impact on the performance of effectively classifying anomalies. The method starts with a linearly spaced vector of possible threshold values, starting at zero, and extending to a value of n , containing m samples. It then iterates through the m thresholds and calculates the binary classification outcome for each threshold. The binary classification is then used to calculate the MCC, AUC and AP scores, which are averaged into a balanced score. Finally, the threshold that corresponds to the highest balanced score is returned. This method for determining the optimal threshold for the balanced score is presented as Algorithm 4.3.

Algorithm 4.3: Anomaly score threshold optimisation algorithm

```
input :  $Y = \{y_1, \dots, y_m\}$  is a linearly spaced vector, starting at zero, and  
        extending to a value of  $n$ , containing  $m$  samples  
output: an optimal threshold value  $Y[index]$   
for  $i = 1$  to  $m$  do  
    read  $y_i$  from  $Y$ ;  
    calculate the binary classification outcomes for  $y_i$ ;  
    calculate the MCC, AUC and AP scores from the binary classification  
        outcomes;  
    calculate the balanced score  $r_i$  by averaging the MCC, AUC and AP  
        scores;  
     $R \leftarrow r_i$            // the balanced score computed  
end  
 $max \leftarrow R[0]$ ;  
for  $i = 1$  to  $m$  do  
    if  $R[i] > max$  then  
         $max \leftarrow R[i]$ ;  
         $index \leftarrow i$ ;  
    end  
end  
output  $Y[index]$ ;
```

The threshold optimisation process for the test set is illustrated in Figure 4.2, where it is apparent that the highest balanced score is achieved at a threshold of 5.5. The value of n was 20, the maximum possible threshold value. The total number of threshold samples m was set to 100.

The illustration in Figure 4.2 depicts the threshold optimisation process that takes place on the optimal architecture following the completion of the NAS process. Therefore, the threshold optimisation algorithm is used twice, namely to determine the threshold for each architecture during the NAS process, and then to determine the test set threshold for the optimal architecture following the completion of the NAS process.

The threshold is represented by the dark blue line in Figure 4.3 which attempts to best split the two classes, with the non-anomalous classes mainly occurring below the threshold due to their low reconstruction error. The converse is true for the anomalous class, where most have a high reconstruction error and are therefore above the threshold. Algorithm 4.3 effectively finds the best fit for the threshold, as determined by the balanced score.

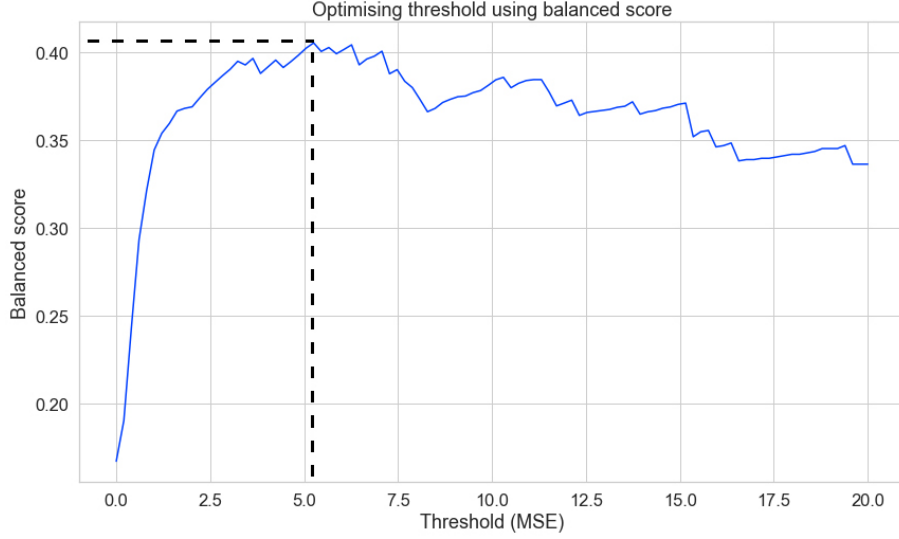


Figure 4.2: Optimising threshold selection using the balanced score

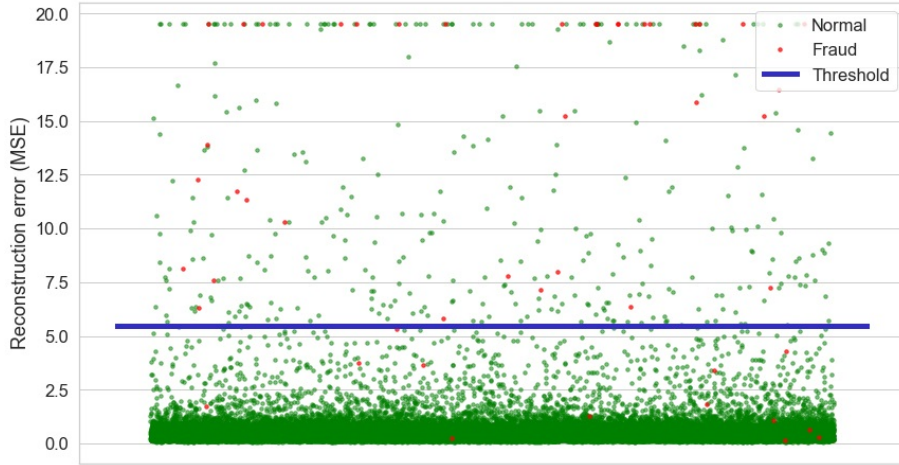


Figure 4.3: Reconstruction errors (MSE)

Following the implementation of Algorithms 4.2 and 4.3, a winning architecture is returned. An example of 50 simulated architectures with their corresponding balanced scores is presented in Figure 4.4. The best performing architecture is the 14th architecture, as illustrated by the red bar in Figure 4.4, with a balanced score of 73.2%. This architecture has 11 hidden layers and 5 361 learnable parameters, as per Table 4.1. The architecture column in Table 4.1 contains a value of [4, 8, 19, 22,

23, 28], which represents the number of nodes in each hidden layer of the optimal architecture. This is the structure used to create the encoder and decoder portion of the model. The bottleneck layer contains four nodes, and the values of 8 through 28 are the number of nodes in the remaining hidden layers for each of the encoder and decoder.

Each autoencoder’s encoder component is a mirror image of the decoder component in terms of structure, with a single bottleneck layer between them. For the sake of clarity, the full list of the 11 hidden layers for the optimal architecture would be [28, 23, 22, 19, 8, 4, 8, 19, 22, 23, 28]. The distribution of the 50 balanced scores exhibits a reasonable amount of variance, which is expected, considering the random nature of the architecture construction process. Due to the upper bound and depth constraints placed on the construction process, there are no exceedingly bad or failed architectures.

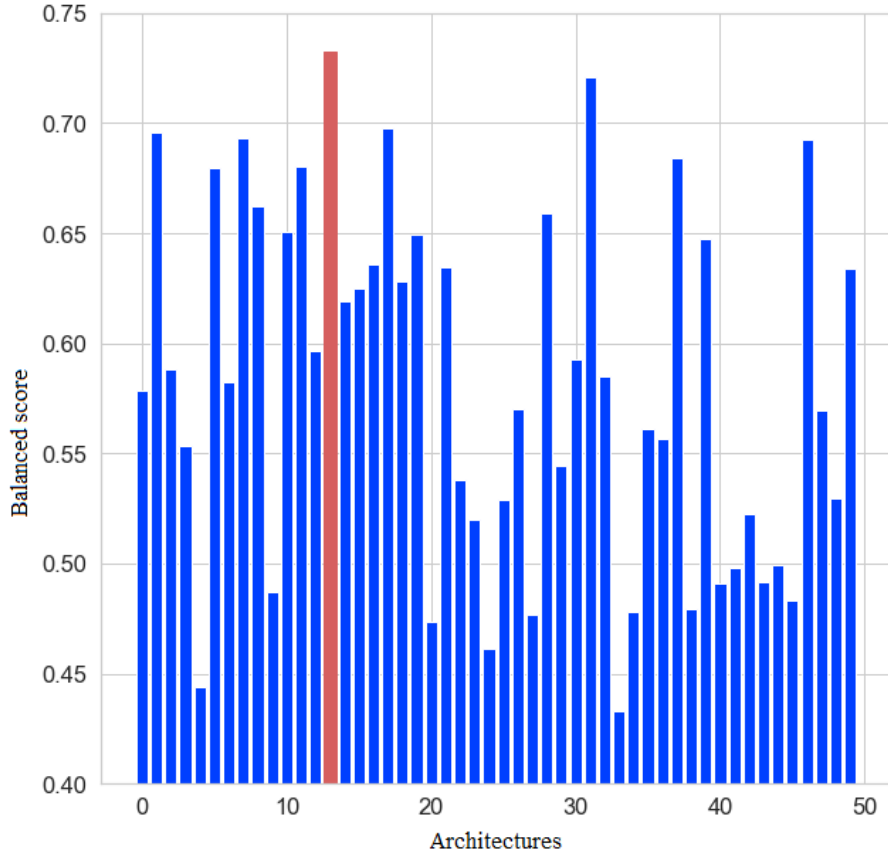


Figure 4.4: Simulating 50 architectures with NAS

Combining the metrics into an average or balanced score helps to normalise the differing magnitudes of the performance metrics. For example, the optimal architecture has an AUC score of 89%, an MCC score of 74.4%, and an AP score of only 56.4%. This would still be considered a good model (the best of 50 simulations), even though a metric score of 56.4% might not necessarily suggest so. According to Saito and Rehmsmeier (2015), the baseline for average precision is the fraction of positive classes, which in this case would be less than 1%, whereas the baseline for an AUC classifier is 50%. Therefore, an AP score of 56.4% is an acceptable score considering that the baseline would be less than 1% for the positive class.

Table 4.1: Details of top 20 architectures

AP	AUC	MCC	Balanced score	Architecture	Params	Threshold
0.563	0.890	0.743	0.732	[4,8,19,22,23,28]	5361	3.030
0.541	0.891	0.728	0.720	[5,8,9,10,14,20,23]	3700	3.232
0.498	0.896	0.698	0.697	[3,4,14,15,22,23,25]	5066	3.232
0.503	0.882	0.701	0.695	[7,14,24,25]	2647	3.434
0.503	0.874	0.701	0.693	[3,7,10,12,13,17,20,24]	4446	3.232
0.492	0.890	0.693	0.692	[3,9,11,24]	1900	3.434
0.484	0.880	0.687	0.684	[3,8,10,11,14,24,26]	4382	3.232
0.470	0.892	0.677	0.679	[3,6,8,18,21]	2532	3.030
0.478	0.878	0.682	0.679	[3,7,10,19,25]	3116	3.232
0.464	0.848	0.672	0.661	[3,9,12,13,19,23]	3468	2.626
0.450	0.865	0.661	0.659	[3,9,13,14,18,22,24,26,27]	7560	2.828
0.451	0.838	0.661	0.650	[3,8,9,12,28]	2850	3.030
0.432	0.868	0.647	0.649	[3,6,10,13,15,27]	3356	3.030
0.443	0.844	0.655	0.647	[15,16,22,23,26,27]	6634	1.616
0.407	0.871	0.627	0.635	[3,4,10,11,14,15,25]	3442	3.030
0.439	0.811	0.652	0.634	[7,8,14,17,21,27]	4436	3.030
0.397	0.884	0.619	0.634	[3,4,6,9,16,20,28]	4050	3.838
0.417	0.830	0.635	0.627	[3,7,9,18,27]	3184	3.636
0.385	0.880	0.609	0.625	[9,12,15,19]	2158	1.616
0.388	0.855	0.612	0.618	[3,5,6,12,22]	2160	3.232

While AP is generally smaller in absolute value when compared to AUC in cases where there is a severe class imbalance, AP considers how the positive classes are handled, and is, therefore, an important addition to the balanced score. This difference in magnitude between metrics is applied equally to each candidate architecture in the simulation; therefore, it may be considered a fair approach to combining performance metrics, as each candidate architecture is treated equally.

4.4.1 Model loss

Figure 4.5 illustrates the model loss during training. The training process converges after just over 20 epochs, although to demonstrate how long to train each model, the training process was extended to 100 epochs.

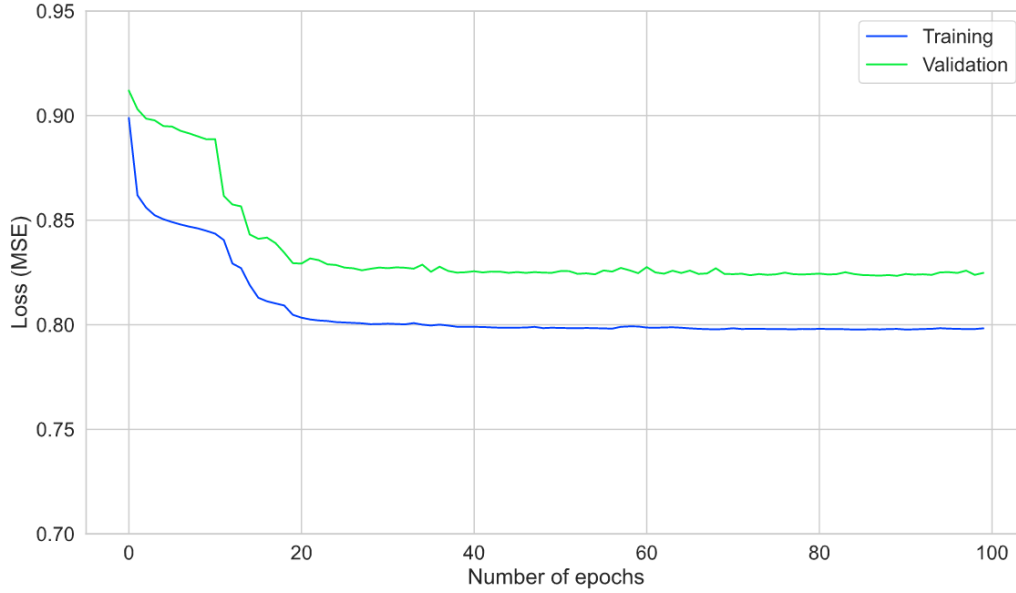


Figure 4.5: Illustrating model loss

The learning process made use of the *Adam* optimiser, which according to Kingma and Ba (2015), is an efficient method for performing stochastic gradient descent. It is also well suited to larger data sets such as the credit card data set used in this chapter for illustrative purposes.

4.4.2 Latent feature representation

To confirm that the model effectively discriminates between the two classes, the model was reshaped to include a bottleneck of two nodes, as presented in Figure 4.6. By reducing the bottleneck to a size of two nodes only, the data in that low-dimensional representation can be visually examined. The low-dimensional representation of the data at the bottleneck portion of an autoencoder is known as the latent space, where it is expected that all the anomalous instances found in the high-dimensional data lie in proximity to one another.

As was discussed in Chapter 2, the method whereby an autoencoder reduces the dimensionality of input data into a low-dimensional space is similar to that of a non-linear form of PCA. The bottleneck portion of an autoencoder represents the output of the compression performed by the encoder and is subsequently used as input into the decoder to reconstruct the data. Autoencoders are versatile in the sense that the decoder can be removed after the model is trained, leaving the encoder and bottleneck portions as a means of compressing data. This compressed data can then be visually represented, or the latent space can be used as features to train a secondary machine learning model.

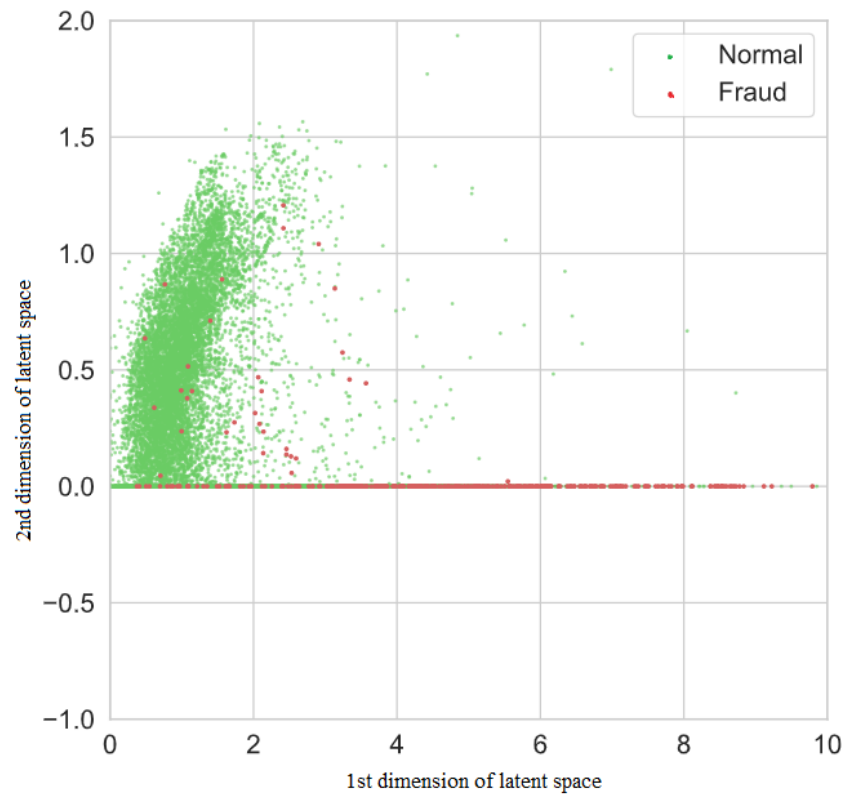


Figure 4.6: Visualising latent representation in two dimensions

There is some degree of separation between the classes in the two-dimensional latent space. This approach is comparable to what was achieved by applying t-SNE to the input data, although this approach illustrates how effectively the encoder compresses the data down into a meaningful lower-dimensional representation.

4.5 Gaussian scaling

The output of the winning architecture is an array of reconstruction scores, one for each transaction. These reconstruction scores, as calculated by the MSE, are not entirely interpretable in their current form, as illustrated in Figure 4.7. To determine the probability that the transaction is anomalous, a normalisation approach is proposed to scale the data to a range of $[0;1]$. One approach is to apply linear scaling to the scores, although due to the nature of the distribution of anomaly scores, this approach does not effectively capture the separation between the classes as presented in Figure 4.7.

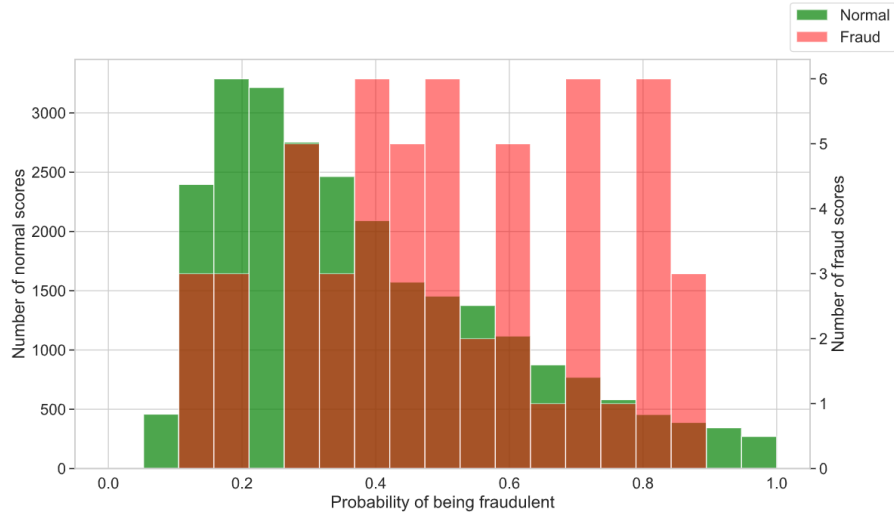


Figure 4.7: Anomaly scores with linear scaling

An alternative approach, applying a Gaussian scaling function to the scores, is proposed. Kriegel *et al.*, (2011) propose that Gaussian scaling is a good general choice function for transforming raw anomaly scores. As is illustrated in Figure 4.8, the resulting distribution of probability scores shows a much better separation between the classes. With the non-anomalous transactions in green, using the left y-axis as its reference, and the anomalous transactions in red, using the right y-axis as its reference, a superior split and one which captures the distribution of the underlying anomaly scores is presented.

According to Figure 4.8, the majority of the normal instances fall within the lowest score tier, whereas the majority of the anomalies fall within the 90%+ tier. This is a favourable outcome for an operational function that makes use of probability scores such as these for their decision-making processes.

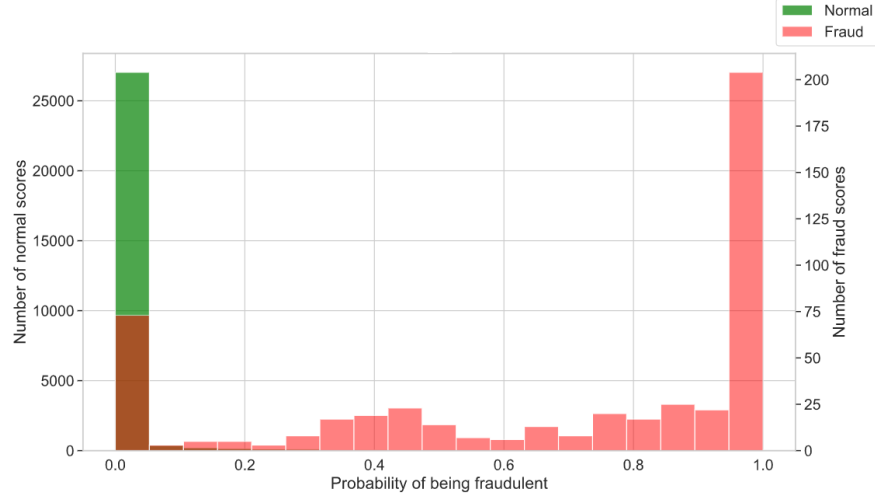


Figure 4.8: Anomaly scores with Gaussian scaling

The Gaussian scaling approach makes one pass of the raw anomaly scores, calculating the mean and standard deviation for the scores. The mean and standard deviation are then used to calculate the cumulative distribution function, which itself is transformed via the Gaussian error function. Finally, the output is transformed to the range $[0;1]$. The algorithm for applying Gaussian scaling is presented as Algorithm 4.4.

Algorithm 4.4: Gaussian scaling of anomaly score algorithm

input : $S = \{s_1, \dots, s_m\}$ is an array of anomaly scores, containing m samples

output: an array $U = \{u_1, \dots, u_m\}$ containing the associated unified scores for m samples

for *one pass of array S* **do**

calculate μ_s // mean

calculate σ_s // standard deviation

$cdf = \frac{S(o) - \mu_s}{\sigma_s * \sqrt{2}};$

$erf(cdf)$ // transform the cdf with the Gaussian error function

normalise the transformation to the range $[0;1]$;

end

output U ;

Considering that the procedure for the unified approach has now been explained, along with a presentation of all of the algorithms, the next section determines the performance of the proposed techniques.

4.6 Performance evaluation of optimal architecture

An approach for determining the performance of the optimal architecture would be to construct a confusion matrix, which compares the true labels from the test set with the classifications made by the trained model. A confusion matrix illustrates the nature of any misclassifications that are made and allows for the determination of additional performance metrics, such as precision and recall. The confusion matrix was previously presented in Chapter 2, Section 2.3.5.

In Table 4.2 the confusion matrix for the optimal architecture on the test set is presented. Of the 49 fraudulent transactions in the test set, 41 were correctly classified as fraudulent, which results in a recall score of 84%. As was discussed in Chapter 2, Section 2.3.5, recall is defined as the number of correctly classified anomalies divided by the total number of anomalies in the test set. The high recall comes at the expense of a much lower precision score (the number of correctly classified anomalies divided by the total number of records classified as anomalies) of 5%, where 776 non-fraudulent transactions were classified as fraudulent. Classification performance metrics, and their relevancy for anomaly detection, will be discussed in more detail later on in the study.

Table 4.2: Confusion matrix of optimal architecture

		Predicted class		Total
		Fraud	Normal	
True class	Fraud	41	8	49
	Normal	776	27656	28432
Total		817	27664	28481

Due to the highly imbalanced nature of the data set, accuracy is neglected as a performance metric, as it does not necessarily account for the imbalance. A model that simply predicts the majority class would end up with a very high accuracy of over 99% even though all the fraudulent transactions would be incorrectly classified. This would not be an effective model and would serve no purpose in an operational context. A more detailed discussion on the performance of the proposed unified approach, as well as the challenges of assessing performance in imbalanced classification problems, will be conducted in Chapter 5, Section 5.6.1.

4.7 Results and discussion

Combining the NAS and threshold optimisation processes provides a formal method for deciding on the appropriate architecture. The approach is straightforward to implement and understand, and the results show an increase in balanced score of around 15% when compared with a trial and error approach. The initial trial and error experiments require a significant amount of time to iterate through possible architecture configurations. These configurations are manually developed, and their design is a subjective process where it is uncertain whether an optimal architecture has been achieved. The subjectivity comes from not knowing whether to increase or decrease the number of layers, whether to increase or decrease the number of nodes in each layer, and so on. The increase in the balanced score is attributed to the number of logical architectures that are simulated in a reasonable amount of time when compared to the same amount of time spent finding an optimal architecture via the trial and error approach. Trial and error open up the possibility of designing a flawed architecture, whereas the NAS approach sets certain conditions that limit the possibility of simulating flawed architectures. Ultimately, the NAS approach automates the simulation process, which allows for a reduced time to achieving an optimal architecture, while allowing for the exploration of architectures that may not occur to the practitioner.

The determination of an optimised threshold is advantageous over the visual inspection approach often used in practice. While it may be possible to visually discriminate effectively, it is unlikely that this will yield a solution that is optimal according to a performance metric well suited to the distribution of the classes. Without the optimised threshold, it is a subjective decision to decide which transactions are fraudulent. The optimised threshold approach, therefore, adds structure and consistency to the anomaly detection problem.

The Gaussian scaling function effectively transforms the raw anomaly scores into a distribution that is indicative of an anomaly detection problem. According to Figure 4.7, using linear scaling yields a distribution that contains some level of discrimination between the classes, but as is shown by Figure 4.8, Gaussian scaling helps to represent the scores in a manner that represents the discrimination between the classes evident in Figure 4.6.

In this chapter, the aim was to propose a method to detect anomalies. This method was illustrated on a Kaggle data set, to validate the method to ultimately apply it to real-world data and build it into an anomaly detection framework.

4.8 Extension of proposed approach

The social network analysis concepts introduced in Chapter 3 (Section 3.3) have not been included in this chapter. While the transactional features in this study capture the various transactional behaviours that users within the system exhibit, these features do not capture the relationships between users. Variables that explain relationships are essential in the context of fraud detection and often reveal additional information that may lead to the identification of anomalies. Social network analysis provides an opportunity to transform data that captures the linked nature of relationships between users, effectively representing these relationships as a mathematical graph. The implications for this representation are twofold. Firstly, graph data allows for visualising the relationships between users in graphical form and secondly, social network analysis algorithms can be applied to the linked data to determine metrics that describe the connectedness of the users within the entire graph of users. Neither of these is possible without data that captures these relationships between users, therefore specific networked data will be included in the unified approach to expand the set of features beyond the transactional level of aggregation. This network data extension of the unified approach will be presented in Chapter 6.

4.9 Chapter conclusion

The purpose of this chapter, the only chapter in Part II, was to report on the creation of a unified approach for detecting anomalies in a transactional setting, which can be built into a machine learning-based framework that can be used in a real-world setting. An example credit card fraud data set was used throughout the study to evaluate the unified approach.

A population of candidate models was simulated using a NAS, with the optimal model being selected via the determination of a balanced score, namely an average of three effective evaluation metrics for anomaly detection problems. A threshold optimisation algorithm was used to determine the binary classification outcomes within the NAS process. Finally, a Gaussian scaling function was applied to the resulting anomaly scores to transform them into a probability score. The resultant scores are easier to interpret than anomaly scores and provide a more accurate representation of the distribution when compared with a linear scaling approach. In addition to the computational success, the accurate representation, and easily interpretable results, the proposed unified approach also provides a structured methodology for applying anomaly detection in a transactional setting.

In the next chapter, which is the first chapter in Part III of the study, the proposed unified approach presented in this chapter will be applied to a real-world

data set. The intention is to prove that the proposed approach can be successfully applied to detect anomalies in a transactional setting.

Part III

Model extension and real-world validation

Chapter 5

Practical real-world application of autoencoders for transactional anomaly detection

5.1 Chapter introduction

Chapter 5 is the first chapter of Part III of the study, which consists of three chapters. The purpose of Part III is to present details on the application of the proposed unified approach that was presented in Part II of the study. The real-world applications and the proposed model extension, therefore, form the focus of Part III. In this chapter, real-world data sourced from an industrial organisation is used to validate the proposed unified approach illustrated in Chapter 4. The proposed unified approach is applied to a data set taken from a transactional setting to assess whether the approach can be extended to a real-world anomaly detection context. This involves performing the unified approach to what is considered standard transactional features.

The aim is to validate the approach to construct a trained autoencoder, which will ultimately form part of an anomaly detection framework to be used alongside a self-organising map (SOM) for detecting transactional anomalies in the industry. The SOM and anomaly detection framework will be introduced in Chapter 7 and Chapter 8, respectively. The unified approach, which is in the same form as the unified approach proposed in Chapter 4, includes using a neural architecture search approach to simulate several candidate autoencoder architectures to design an optimal architecture that can be used to classify anomalies. A threshold optimisation algorithm is applied to determine the optimal threshold on the validation and test sets, while Gaussian scaling is used to represent the anomaly scores in an interpretable form.

Anomaly detection decisions often require knowledge of features that are important in the modelling process. To determine the contribution and importance of each feature in the models used, a well-known technique called SHAP (Lundberg and Lee, 2017) is also introduced to further enhance the results of the anomaly detection process.

The remainder of this chapter is organised as follows: in Section 5.2 the standard industry data set is introduced and details on the application of the proposed unified approach to this data set are included. This is followed by the neural architecture search and threshold optimisation processes in Sections 5.3 and 5.4, respectively. In Section 5.5, Gaussian scaling is applied to the outputs derived from the optimal model, while a performance metric evaluation of the optimal architecture is provided in Section 5.6. This includes an introduction to SHAP, which is used to determine which features contributed to the detection of anomalies within the industry data. In Section 5.7, a discussion of the results is provided, while Section 5.8 finalises the chapter with conclusions about the results.

5.2 Description of the industry data

The real-world data set used in this chapter is sourced from an organisation in the financial services industry. While the organisation is not actively involved in this research study, they have provided access to their data set for research purposes. This industry organisation provides financial services to 10 million customers, and the business ecosystem involves buying, selling, making and receiving deposits, as well as the use of credit cards to fund user accounts. Considering that each of these aforementioned activities is recorded in the organisation’s production databases, the data set is therefore considered transactional. More than 50 000 unique payments and 400 000 unique trading transactions are facilitated by the organisation per day.

For this research study, the data set used is related to credit card fraud and contains a class imbalance. A class imbalance exists because the data set contains a very large number of records, and the number of fraudulent transactions is, in proportion to a large number of normal transactions, small. The original state of the data set, that is the raw data set before any preprocessing, consists of aggregates and Boolean variables that attempt to capture the transactional behaviour of the users within the industry organisation’s operational setting.

The data set has been anonymised and therefore the unique identifier has been removed so that the underlying users can in no way be traced. This anonymisation process was undertaken at the point where the data set was transferred across from the industrial organisation, before being used for this study. A naming convention is used where the names of the features have been changed from their descriptive names

to v1 through v47. This is to reduce the possibility of external bad actors reverse-engineering the fraud prevention practices at the industry organisation.

Transactionally aggregated and user-aggregated features are two distinct methods of expressing data in fraud detection systems, both of which prove to be effective in detecting anomalies (Dal Pozzolo, 2015). The industry data is aggregated at the user level. In practice, this means that there will only ever be a single row of data for each user. In contrast, data aggregated at the transactional level would mean that many rows could exist for a single user, depending on how many transactions each respective user makes. These aggregates are calculated periodically, aggregating data as new transactions are created. Some features are time-dependent and calculate the aggregated value within a trailing window period. An example of this would be the number of deposits made over the past seven days.

Other features look at transactions for all periods, such as the number of unique credit cards that a user has attempted using on the industry organisation’s platform. The user-level aggregated format is to be used for both training the model, as well as for making predictions against the trained model. Once the model has been trained, the aggregated features will continue to be generated and served to the model for fraud detection purposes. A sample of the first rows of the data is illustrated in Table 5.1. This table is a transposed representation of the original data set, and therefore the original features are illustrated as rows in Table 5.1. Transposition has been performed to facilitate a clear representation of the 47 features.

The original data taken from the industrial organisation is comprised of 120 676 transactions, of which 439 records are confirmed as fraudulent. This equates to a class imbalance where only 0.36% of the classes are attributed to the minority class. There are 49 features in the data set, of which one is the unique identifier of each user, and another is the class label or target variable indicating a confirmed credit card fraud case. Instances of confirmed fraud are assigned a target variable of 1, while instances of no credit card fraud are assigned a target variable of 0. For model training, the identifier is excluded, but can be used to map the vectors and their respective prediction output back to each user’s unique identifier. The class label is kept and will be used for classification purposes. Therefore, the 49 features are reduced to 47 features to train each model. No missing values are found in the data set, as the aggregation process used returns a 0 rather than a null value in instances where no values are counted for the respective feature.

Due to the data being sourced from an industrial organisation, that requires that the data be kept confidential, and considering the sensitivity of financial data, the remaining 47 features will not be described in detail. These features are typical features found in a transactional database and relate to financial and behavioural attributes associated with each transaction.

Table 5.1: First 10 samples of industry data

	0	1	2	3	4	5	6	7	8	9
fraud	0	0	0	0	0	0	0	0	0	0
v1	1.068	-0.935	-0.935	1.068	-0.935	-0.935	1.068	-0.935	-0.935	1.068
v2	0.965	0.965	-1.036	0.965	0.965	0.965	0.965	-1.036	0.965	-1.036
v3	-0.882	-0.882	1.133	-0.882	1.133	-0.882	-0.882	1.133	-0.882	1.133
v4	1.154	-0.866	-0.866	-0.866	-0.866	1.154	1.154	-0.866	1.154	-0.866
v5	-0.391	2.552	-0.391	2.552	-0.391	-0.391	-0.391	-0.391	-0.391	-0.391
v6	-0.055	-0.028	-0.278	-0.277	-0.278	0.013	-0.276	-0.278	-0.277	-0.278
v7	-0.185	-0.185	-0.185	-0.185	-0.185	-0.185	-0.185	-0.185	-0.185	-0.185
v8	2.079	-0.480	-0.480	-0.480	-0.480	2.079	-0.480	-0.480	2.079	-0.480
v9	-0.546	-0.546	-0.546	1.831	-0.546	-0.546	-0.546	-0.546	-0.546	1.831
v10	0.312	0.312	0.312	0.312	0.312	0.312	-0.350	-0.350	0.312	-0.350
v11	-0.292	0.499	-0.292	19.490	-0.292	-0.292	5.247	-0.292	0.499	-0.292
v12	-0.308	0.804	-0.308	3.029	-0.308	-0.308	0.804	-0.308	1.360	-0.308
v13	-0.177	-0.177	-0.177	-0.177	-0.177	-0.177	-0.177	-0.177	1.014	-0.177
v14	-0.093	-0.093	-0.093	-0.093	-0.093	-0.093	-0.093	-0.093	5.425	-0.093
v15	-0.136	-0.136	-0.136	-0.136	-0.136	-0.136	-0.136	-0.136	-0.136	-0.136
v16	-0.273	-0.273	-0.273	-0.273	-0.273	-0.273	-0.273	-0.273	1.023	-0.273
v17	-0.336	-0.336	-0.336	-0.336	-0.336	-0.336	-0.336	-0.336	1.114	-0.336
v18	-0.665	1.498	1.498	-0.665	1.498	1.498	-0.665	1.498	1.498	-0.665
v19	-0.398	-0.398	-0.398	-0.398	-0.398	-0.398	2.507	2.507	2.507	2.507
v20	-0.324	-0.324	-0.324	-0.324	-0.324	-0.324	-0.324	-0.324	-0.324	-0.324
v21	-0.069	-0.069	-0.069	-0.069	-0.069	-0.069	-0.069	-0.069	-0.069	-0.069
v22	-0.285	-0.285	-0.285	-0.285	-0.285	-0.285	-0.285	-0.285	-0.285	-0.285
v23	-0.021	-0.021	-0.021	-0.021	-0.021	-0.021	-0.021	-0.021	-0.021	-0.021
v24	-0.060	-0.060	-0.060	-0.060	-0.060	-0.060	-0.060	-0.060	-0.060	-0.060
v25	-0.123	8.120	-0.123	-0.123	-0.123	-0.123	-0.123	-0.123	-0.123	-0.123
v26	-0.137	7.252	-0.137	-0.137	-0.137	-0.137	-0.137	-0.137	-0.137	-0.137
v27	-0.044	-0.044	-0.044	-0.044	-0.044	-0.044	-0.044	-0.044	-0.044	-0.044
v28	-0.041	-0.041	-0.041	-0.041	-0.041	-0.041	-0.041	-0.041	-0.041	-0.041
v29	-0.008	-0.008	-0.008	-0.008	-0.008	-0.008	-0.008	-0.008	-0.008	-0.008
v30	-0.185	-0.185	-0.185	-0.185	-0.185	-0.185	-0.185	-0.185	-0.185	-0.185
v31	-0.213	4.677	-0.213	-0.213	-0.213	-0.213	-0.213	-0.213	-0.213	-0.213
v32	-0.221	4.521	-0.221	-0.221	-0.221	-0.221	-0.221	-0.221	-0.221	-0.221
v33	-0.039	-0.039	-0.039	-0.039	-0.039	-0.039	-0.039	-0.039	-0.039	-0.039
v34	-0.007	-0.007	-0.007	-0.007	-0.007	-0.007	-0.007	-0.007	-0.007	-0.007
v35	-0.032	-0.032	-0.032	-0.032	-0.032	-0.032	-0.032	-0.032	-0.032	-0.032
v36	-0.065	-0.065	-0.065	-0.065	-0.065	-0.065	-0.065	-0.065	-0.065	-0.065
v37	-0.073	-0.073	-0.073	-0.073	-0.073	-0.073	-0.073	-0.073	-0.073	-0.073
v38	-0.007	-0.007	-0.007	-0.007	-0.007	-0.007	-0.007	-0.007	-0.007	-0.007
v39	-0.170	-0.170	-0.170	-0.170	-0.170	-0.170	-0.170	-0.170	-0.170	-0.170
v40	-0.176	5.662	-0.176	-0.176	-0.176	-0.176	-0.176	-0.176	-0.176	-0.176
v41	-0.178	5.605	-0.178	-0.178	-0.178	-0.178	-0.178	-0.178	-0.178	-0.178
v42	-0.019	-0.019	-0.019	-0.019	-0.019	-0.019	-0.019	-0.019	-0.019	-0.019
v43	-0.016	-0.016	-0.016	-0.016	-0.016	-0.016	-0.016	-0.016	-0.016	-0.016
v44	-0.017	-0.017	-0.017	-0.017	-0.017	-0.017	-0.017	-0.017	-0.017	-0.017
v45	-0.076	13.100	-0.076	-0.076	-0.076	-0.076	-0.076	-0.076	-0.076	-0.076
v46	-0.086	11.582	-0.086	-0.086	-0.086	-0.086	-0.086	-0.086	-0.086	-0.086
v47	-0.108	-0.108	-0.108	-0.108	-0.108	-0.108	-0.108	-0.108	-0.108	-0.108

Feature examples include the IP address from where the transaction originates, how many cards the user has attempted to use, and whether there are differences between from which country the user is signed in and where his/her identity documents suggests he/she is located.

The real-world data features are transformed via standardisation, where each vector is centred around the mean with a unit standard deviation. This means that the mean of each feature becomes zero and the resultant distribution has a unit standard deviation, yielding a standard normal distribution with $N(0, 1)$. The formula for standardisation is as follows:

$$X' = \frac{X - \mu}{\sigma}, \quad (5.1)$$

where μ is the mean and σ is the standard deviation of the values in each feature. The values returned by the standardisation process are not defined to a particular range as is the case with normalisation, and therefore values greater than 1 are to be expected. Following the application of the standardisation process, as well as the removal of the unique identifier and keeping the target variable aside, the data is then considered ready for model training purposes.

5.2.1 Latent space representation using t-SNE

The t-SNE process defined in Algorithm 4.1 was applied to the data to reduce the dimensionality of the features into a lower-dimensional representation, namely two dimensions. The reasons for doing so are because high-dimensional data is difficult to visualise and analyse, and so by reducing the features to two dimensions it is possible to visualise the data across two axes, which can easily be analysed by the human eye. The two classes of the data set can then be mapped onto the resulting two-dimensional visualisation, to gain an understanding of whether a separation between the classes is feasible.

An alternative visualisation approach that could also be applied is Uniform Manifold Approximation and Projection (UMAP) (McInnes and Healy, 2018). UMAP assumes that data is uniformly distributed on a Riemannian manifold, where the Riemannian metric can be approximated and the manifold is locally connected. For larger data sets, UMAP provides performance improvements in terms of computation time, and it was designed specifically for visualising complex data in low dimensions. While UMAP is a promising technique in the literature, this study will focus on using t-SNE to visualise the input space.

The parameters used for fitting the t-SNE algorithm include a perplexity value (*Perp*) of 30 and a learning rate (η) of 200. The perplexity value was chosen by

reviewing the work of Maaten and Hinton (2008) who found that perplexity values within the range of 5 to 50 yield good results in several applications. Applying t-SNE to the real-world data yields a seemingly acceptable split between the classes, as is illustrated by Figure 5.1.

The majority of the fraudulent instances are clustered together in red, although a few fraudulent instances lie within the normal clusters too. These misplaced fraudulent instances are within the black circle in Figure 5.1. There appear to be a few, somewhat distinctive clusters in the real-world data. A possible explanation for the spiral clusters found in the data may be due to the sparsity of some of the feature vectors.

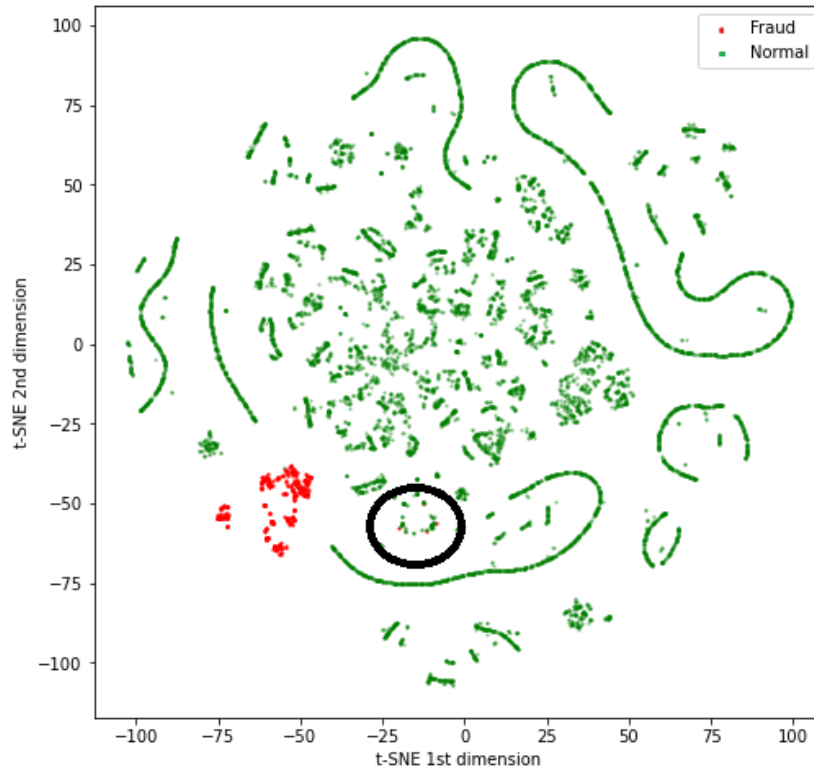


Figure 5.1: Visualising latent space using t-SNE on industry data

Considering that the data contains several aggregates over the transactional lifetime of a user, there would likely be a significant subset of users who never transact and therefore hold very little information in their feature vectors.

5.2.2 Sampling approach

The sampling approach explained in Chapter 4, was applied in order to prevent bias when sampling from the industry data set. The approach can be described as follows. Let $D = \{d_1, \dots, d_n\}$ represent the original data set consisting of n records. The set V , where $V = \{v_1, \dots, v_t\}$ and $V \subseteq D$, is the set with t non-anomalous records. Then set F , expressed as $F = \{f_1, \dots, f_s\}$ where $F \subseteq D$ and $F \cup V = D$, is the set with s fraudulent records. Therefore, $t + s = n$, which is the number of records in the original data set.

The approach, as applied to the real-world dataset, is depicted in Figure 5.2 and may be described as follows. The sampling process starts with $n = 120\,676$ records, which are randomised so that the records are sorted in a random order to avoid any potential biases. The resulting randomised feature set is split into a training set T_1 and a test set T_2 , with T_1 containing 90% and T_2 containing the remaining 10% of the data set, *i.e.*, $T_1 = 0.9D$. This splitting includes stratified sampling so that each of the resulting sets contains a proportional share of the fraudulent class. At this point, the test set T_2 becomes a holdout set, while the training set T_1 is used for further sampling.

The training set T_1 is randomised again and then split into a further two sets, namely a training set T_3 comprising 90% of the previous training set and a validation set T_4 made up from the remaining 10% of the previous training set T_1 . Therefore $T_3 = 0.9T_1$. Stratified sampling is applied again to partition the classes about their proportions.

Let $S_1 \subseteq F$ be the set of fraudulent instances from the new training set T_3 . The set (S_1) is removed from T_3 so that the final training set contains only non-fraudulent instances, while the validation set T_4 keeps the fraudulent instances originally assigned to it. This is to train each candidate autoencoder in an unsupervised learning manner (*i.e.* using only non-fraudulent instances), while evaluating their performance against the validation set T_4 that contains labels from both classes. Following the determination of the winning architecture, the holdout test set T_2 is used to make a final evaluation of the model's performance on unseen data. The objective is to check whether the trained model performs as well on unseen data as it does on data on which it was trained, a concept known as model generalisation (Mohri *et al.*, 2012).

The set of fraudulent instances (S_1) removed from the training set T_3 are added back to the validation set T_4 to augment the validation set and provide fraudulent samples from which to assess each candidate architecture's performance. Removing S_1 from the training set yields the final training set T_5 , while the updated validation set becomes T_6 . Ultimately, the updated validation set T_6 ends up with S_1 , which

contains 395 fraudulent instances, while the test set T_2 contains 44 fraudulent instances. Together these make up the set F , which includes all s fraudulent instances mentioned in Section 5.2.

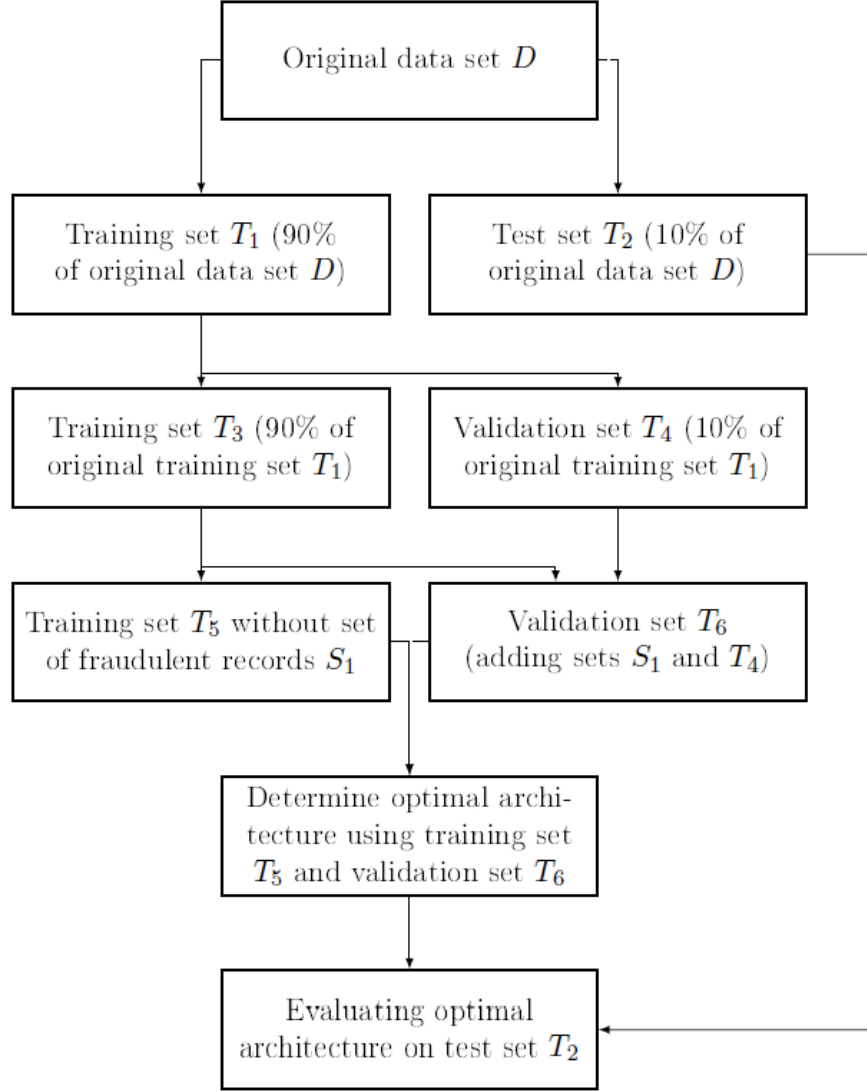


Figure 5.2: Sampling process

The sampling process concludes the discussion on the industrial data set used and the next section will describe the NAS procedure that was applied to the data set.

5.3 Neural architecture search using industry data

In Chapter 4, an NAS procedure was proposed and tested, with the results indicating that it was an effective way of performing NAS. The approach validated in Chapter 4 is applied to the industry data set. This involves applying Algorithm 4.2 to simulate 100 candidate architectures. While 50 architectures were simulated in Chapter 4, this was only for validation purposes, and 100 architectures will be simulated in this chapter to ensure improved results from the model search space. In this chapter real-world data is being used, thus the emphasis is placed on enhancing the NAS process to a greater extent to maximise performance results. The threshold optimisation algorithm presented as Algorithm 4.3 is applied to each architecture during the NAS process. The results of these simulations are presented in Figure 5.3.

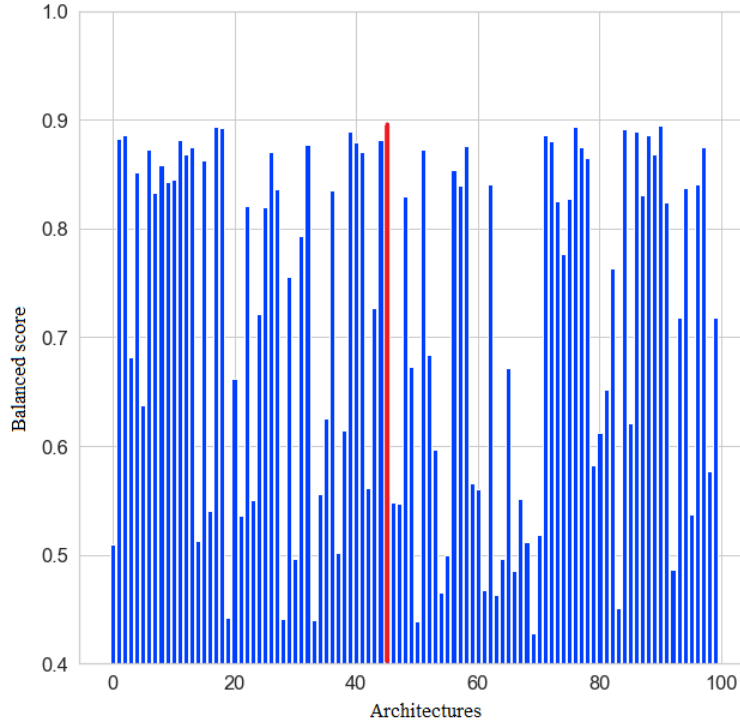


Figure 5.3: Simulating 100 architectures with NAS using industry data

Much variation exists in the distribution of the balanced scores derived from the NAS process. The best-performing architectures achieved a balanced score of just under 90%, with the highest score indicated by the red bar in Figure 5.3, whereas the worst-performing architectures have balanced scores below 45%. To reiterate, the balanced score is a performance metric comprised of the mean of three important measures related to anomaly detection, namely Average Precision (AP), Area Under the ROC curve (AUROC), and a normalised Matthews Correlation Coefficient (MCC). The balanced score was proposed in Chapter 4 and was found to be an effective measure for comparing autoencoder model performance.

The results of the NAS process on the industry data are displayed in Table 5.2. The best performing architecture achieved a balanced score of 89.7% on the validation set. The optimal threshold was determined to be 5.353. An interesting observation is that this architecture was one of the deepest architectures from the top 20 architectures, although slightly deeper architectures do not always yield better performance. This is confirmed by the architectures in positions 18 and 20 which have deeper architectures, but do not yield higher balanced scores.

Table 5.2: Details of top 20 architectures using industry data

AP	AUC	MCC	Balanced score	Architecture	Params	Threshold
0.804	0.995	0.892	0.897	[10,17,26,28,32,34,40,43]	17347	5.353
0.801	0.993	0.891	0.895	[7,17,19,22,24,46]	9618	5.050
0.798	0.994	0.889	0.894	[5,7,30,35,45]	10256	5.454
0.797	0.995	0.889	0.894	[8,21,25,27,30,32,39,44]	16831	5.656
0.794	0.995	0.887	0.892	[3,12,22,25,34,37]	9704	5.656
0.793	0.995	0.886	0.891	[4,19,28,32,37,40,46]	16795	5.353
0.789	0.993	0.883	0.889	[8,9,11,33,38,43,44]	15175	5.757
0.788	0.995	0.883	0.889	[8,13,23,35]	5903	5.252
0.783	0.994	0.880	0.886	[9,10,15,34,42]	8562	5.252
0.783	0.994	0.880	0.886	[7,25,26,27,28,29]	9240	5.151
0.783	0.994	0.880	0.886	[13,15,20,24,33,42]	10582	4.646
0.777	0.994	0.877	0.883	[9,14,21,23,28,36,37,44]	15628	5.353
0.774	0.994	0.875	0.881	[5,10,13,31,35,40,42]	13838	5.757
0.775	0.993	0.875	0.881	[9,12,14,27,29]	5820	5.555
0.773	0.993	0.874	0.880	[8,13,19,23,25,27,43,46]	15125	5.050
0.769	0.994	0.872	0.879	[9,10,11,13,25,29,36,42]	12234	5.454
0.766	0.994	0.870	0.877	[4,12,17,34,43]	8889	4.040
0.764	0.994	0.869	0.875	[5,6,25,27,29,30,36,37,46]	18092	5.454
0.762	0.994	0.868	0.875	[3,5,7,21,31,32,34,45]	13546	5.555
0.762	0.994	0.868	0.875	[8,18,22,23,25,28,34,37,45]	17141	5.252

The best-performing architecture on the validation set is then used to construct a model that is validated on the industry data test set T_2 , as illustrated in Figure 5.2. This model can be considered the final product of this approach, which could be used to detect anomalies on further unseen samples of data from the same industry context.

The training process for the final model involved constructing an autoencoder architecture with 15 hidden layers, using the Rectified Linear Unit (ReLU) activation function for the hidden layers and a sigmoid activation function for the output layer. According to Nair and Hinton (2010) the ReLU activation function is defined as:

$$f(x) = \max(0, \mathbf{x}), \quad (5.2)$$

which corresponds to the element-wise maximum of 0 and the input vector. The ReLU activation function exhibited improved performance as hidden layers when compared with standard binary hidden layers (Nair and Hinton, 2010). It is frequently used as a hidden activation function for building deep learning models due to its ability to manage sparse activations, its scale invariance, and the ease of its computation.

The sigmoid activation function used in the output layer of the model returns values close to 0 for very small input values, and values close to 1 for larger values. The sigmoid therefore always returns a value between 0 and 1, and is defined as follows:

$$s(x) = \frac{1}{1 + e^{-x}}. \quad (5.3)$$

The model is a sparse autoencoder that enforces a sparsity constraint on the network. L1 regularisation is applied with a regularisation factor of 0.0005. The training process involves training for 100 epochs, with a batch size of 256 samples. During model training, the *Adam* optimisation algorithm is applied with a learning rate of 0.0005.

The selected loss function is the mean square error (MSE), where the model seeks to minimise the mean of squares of the errors between the labels and the predictions. The MSE is formally defined in Section 5.4. Subsequent sections will further elaborate on the model and its results and performance.

5.4 Threshold optimisation using industry data

Since the best architecture has been established, the next step of the proposed unified approach is to perform threshold optimisation. A threshold is required to split the reconstruction errors produced by the model into normal and anomalous groups. The reason for the split is so that the model can produce an output that can inform decision making.

Within a fraud detection context, reconstruction errors above the threshold would be classified as anomalous or fraudulent, whereas reconstruction errors below the threshold would be deemed normal. The algorithm used for determining anomalies by using the reconstruction errors generated by an autoencoder is presented below as Algorithm 5.1.

Algorithm 5.1: Autoencoder based anomaly detection algorithm (An and Cho, 2015)

input : normal data set X , anomalous data set $\{x^i\}$ with $i = \{1, \dots, N\}$
threshold α
output: reconstruction error $\|x - \hat{x}\|$, list of anomalous transactions, list of non-anomalous transactions
 $\phi, \theta \leftarrow$ train an autoencoder using the normal dataset X ;
for $i = 1$ **to** N **do**
 reconstruction error $(i) = \|x^i - g_\theta(f_\phi(x^i))\|$;
 if *reconstruction error* $(i) > \alpha$ **then**
 | x^i is an anomaly;
 else
 | x^i is not an anomaly;
end

Araya *et al.*, (2016) refer to a threshold as being generally defined by the MSE. The formula for the MSE, which is the mean of the squares of the errors, is defined as follows:

$$MSE = \frac{1}{n} \sum_{i=1}^n (Y_i - \hat{Y}_i)^2. \quad (5.4)$$

Due to the squaring element of the MSE, outliers are heavily weighted, which is beneficial for anomaly detection where the emphasis is on explicitly surfacing outliers. This stands in contrast to the Mean Absolute Error (MAE), which is less sensitive to the impact of outliers.

In Table 5.2, the optimal threshold for each candidate architecture is shown. This threshold is set against each architecture’s performance on the industry data validation set. Figure 5.4 illustrates the results of the threshold optimisation process for the optimal architecture, where the optimal threshold is an MSE of 5.353. Finally, the industry data validation set is used to find the balanced score for each simulated architecture within a range of thresholds.

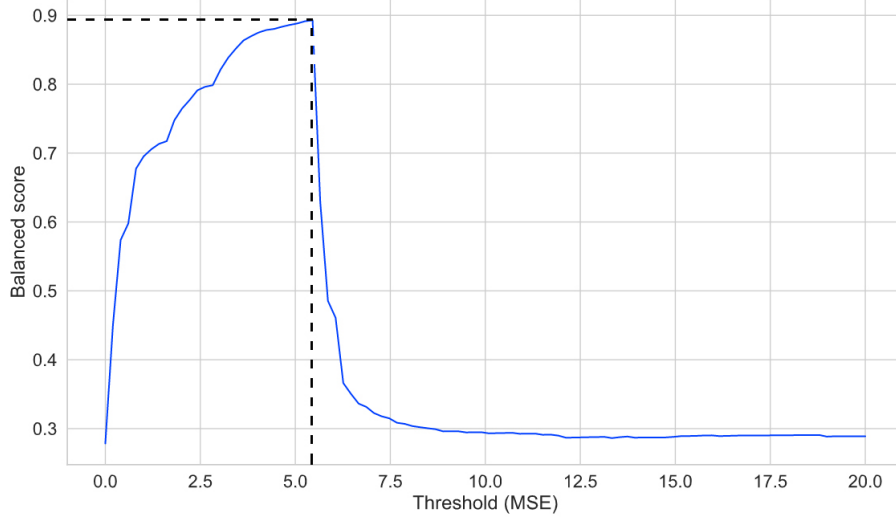


Figure 5.4: Optimising threshold selection using the balanced score

When iterating over the threshold search space of the optimal architecture, as per Algorithm 4.3, the balanced score rises sharply and peaks at 89.7% for an MSE of 5.353. This then drops significantly as the threshold is increased further. The threshold optimisation process does not continue for the test set, rather the threshold determined in Figure 5.4 is used to determine the optimal classification split between the values in the test set. Determining a secondary threshold on the test set data would introduce bias into the process, and would be considered overfitting.

This difference in the number of anomalies, for a similar overall sample size for each of the validation and test sets, increases the imbalance significantly for the test set, thus impacting the magnitude of the AP measure negatively. This is expected, as discussed by Saito and Rehmsmeier (2015), where the author claims that the baseline for AP scores is the fraction of the number of anomalies in the sample. There are 12 038 samples in the test set, including 44 anomalous labels. Therefore, according to Saito and Rehmsmeier (2015), the baseline AP measure would be 0.365%. In examining the threshold optimisation process, an AP score of 17.1% was recorded at the optimal threshold level, far exceeding the baseline and thus indicating that the

balanced score does indeed find the best threshold regardless of the magnitude of the balanced score.

Figure 5.5 is a graphical representation of the reconstruction error for the validation set. The MSE threshold of 5.353 is represented by the dark blue line in the figure. The upper limit on the reconstruction error values is capped at 20 to reduce the impact of extreme outliers. Should an MSE value greater than this capped value be calculated, then this value is replaced by the capped value. This approach is not deemed to hurt the classifications as the extreme outliers would still be considered anomalies if containing this arbitrarily selected capped MSE value. The approach will have a positive impact on the scaling of the anomaly scores, as will be discussed in Section 5.5.

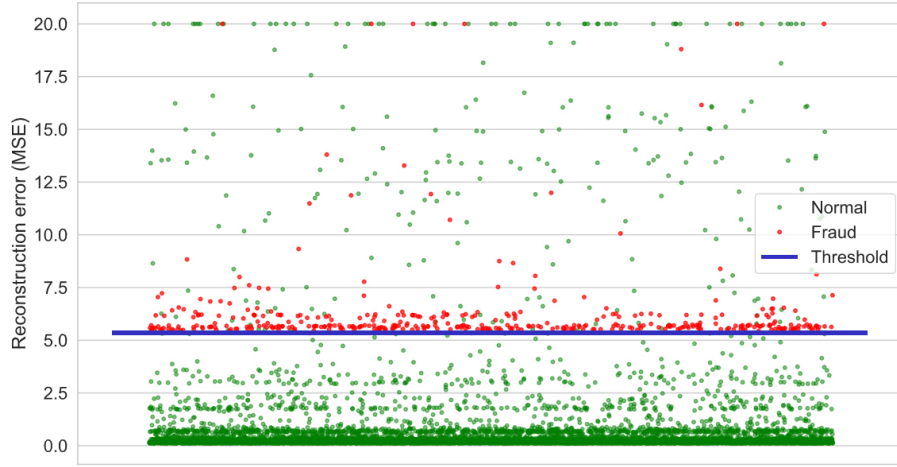


Figure 5.5: Reconstruction errors (MSE) using industry validation data

When the same threshold of 5.353 is applied to the industry data test set, a similar split between the normal and fraudulent classes is witnessed, as displayed in Figure 5.6.

The vast majority of the normal records sit below the threshold, while all the fraudulent records are above the threshold, thus indicating correct classifications for all fraudulent records in the test set. The model, therefore, generalises well to the unseen test data, including the use of the optimal threshold derived from the threshold optimisation process conducted on the validation set.

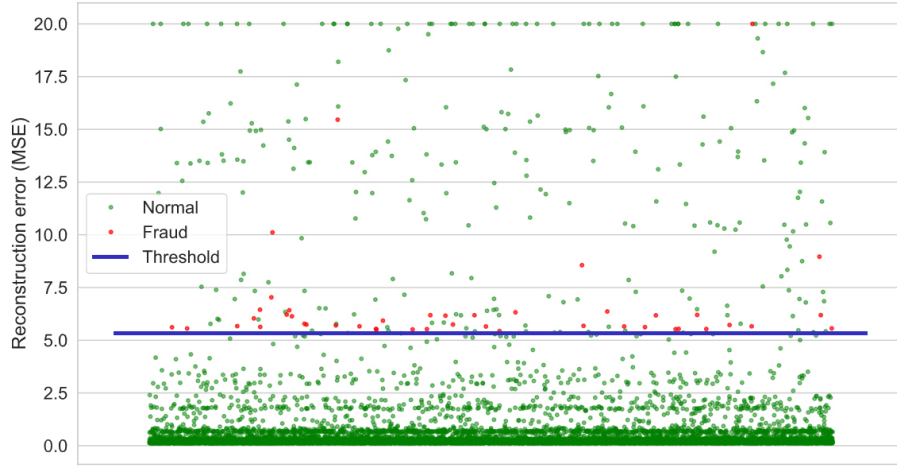


Figure 5.6: Reconstruction errors (MSE) using industry test data

Following the calculation of the optimal threshold, the latent space of the model bottleneck will be explored to determine whether a suitable split between the non-fraudulent and fraudulent classes is achieved.

5.4.1 Latent feature representation

The optimal model derived from the previous section is re-shaped into a form containing two nodes in the bottleneck layer. The reason for this is that output of the compression can be visualised via a two-dimensional representation, which can aid in inspecting the level of discrimination between the classes that the optimal model provides. This latent representation in two dimensions is presented in Figure 5.7.

Figure 5.7 displays a good degree of separation between the normal and fraudulent classes. The normal data appears to be in a representation consisting of two or three elongated clusters, with a further two isolated clusters positioned to the right of the figure. The fraudulent data appears to occupy one or two sparsely populated elongated clusters that occupy the same space as the two smaller, isolated clusters from the normal data set.

The fact that the fraudulent data forms groupings within the spaces where normal data is sparse indicates that the model has effectively learned to identify normal transactions, which in turn leads the model to effectively discriminate between normal and anomalous transactions.

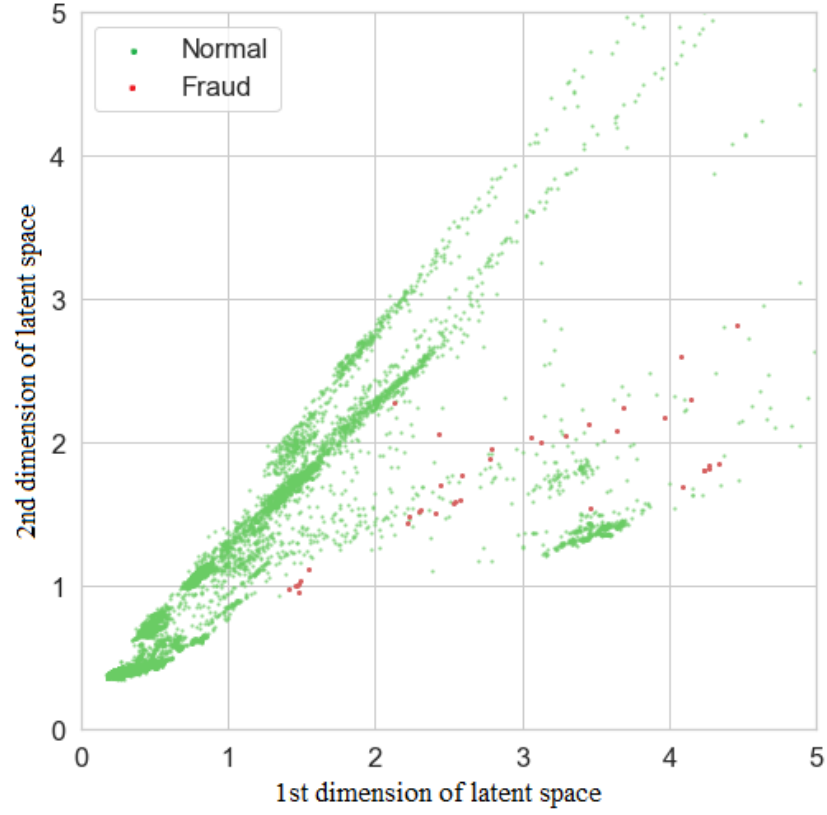


Figure 5.7: Visualising latent representation in two dimensions

The analysis of the latent feature representation is the final activity of the threshold optimisation process. The next step, which is detailed in the following section, would be to perform the Gaussian scaling of the obtained output.

5.5 Applying Gaussian scaling on the industry data

The last step of the proposed unified approach is the Gaussian scaling of the reconstruction errors. Applying linear scaling to the reconstruction scores yields the distributions shown in Figure 5.8. While a good amount of separation between the classes is visible, the bulk of the fraudulent scores clusters around the 0.30 or 30% probability mark. This is due to the scale differences between the bulk of the fraudulent instances and the extreme outliers to the right. These outliers occur even after the 20 MSE cap was placed on the extreme outliers in Figure 5.5.

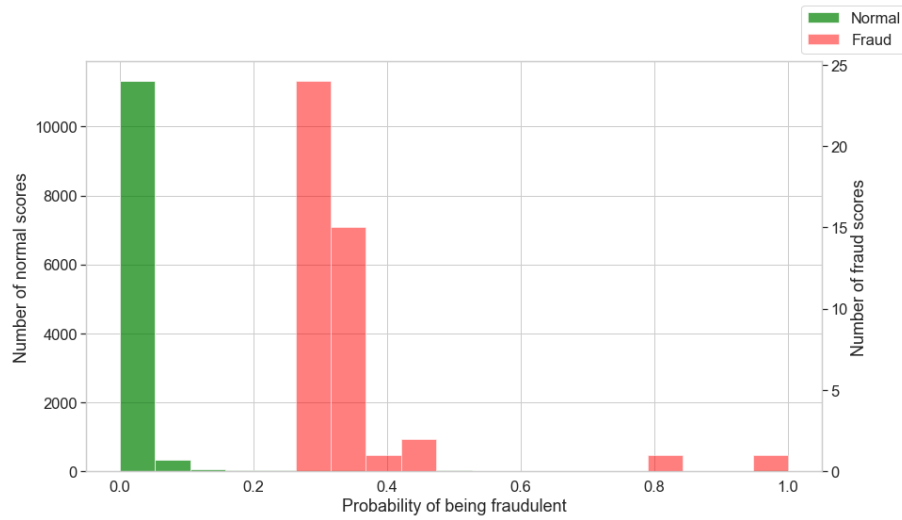


Figure 5.8: Anomaly scores with linear scaling for industry data set

The Gaussian scaling algorithm presented as Algorithm 4.4 assists with these scaling differences. The outcome of applying the Gaussian scaling function to the reconstruction scores is shown in Figure 5.9. Here, there is a far greater split between the classes, with essentially all the fraudulent instances in the test set residing in the plus 90% probability bracket.

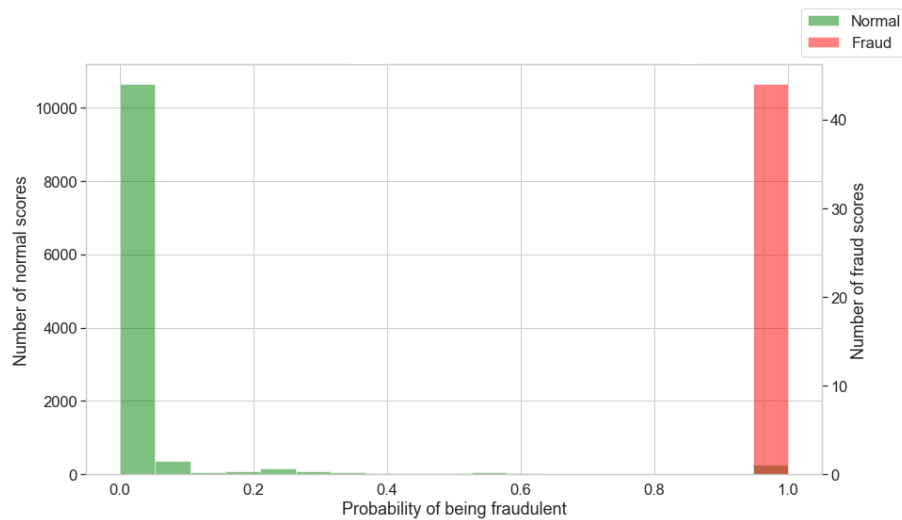


Figure 5.9: Anomaly scores with Gaussian scaling for industry data set

There are also a few misclassifications where normal transactions were classified as fraudulent due to a high score, although the number of misclassifications is minor in comparison with the overall number of correctly classified normal transactions. When making predictions based on the trained optimal architecture developed in Section 5.3, the outcome returned by the model is a reconstruction of the original input data from which the prediction was made. The MSE between these reconstructions and the original input data is then calculated, which is termed the reconstruction error. The errors are in their raw MSE form and should be converted into an interpretable form to be used for operational purposes.

5.6 Evaluating performance of optimal architecture

In this section, the starting point is an analysis of the performance metrics for the model developed in this chapter. Then the concept of SHAP (SHapley Additive exPlanations) is introduced, describing the general SHAP approach to determining feature contribution to model predictions. Feature contribution is important as it provides an element of interpretability to the model evaluation process. More specifically, SHAP can measure the contribution of the additional network metric features, indicating whether these features are significant contributors to detecting anomalies. Finally, an extension of SHAP and how it can be used to determine feature contributions in an anomaly detection context is presented.

5.6.1 Performance metrics

The confusion matrix presented in Chapter 2 (Table 2.1) is used to measure the performance of the model from a binary classification perspective. The holdout test set contains 12 038 records of which 44 are confirmed instances of fraud. This holdout test set is the set used to construct the confusion matrix. The optimal architecture was able to correctly classify all the fraudulent instances in the unseen test set, while misclassifying 227 of the confirmed normal instances. These result in a precision score of 16.2% and a recall score of 100%. An accuracy of 98.1% was achieved by the model, and the confusion matrix is presented in Table 5.3.

Table 5.3: Confusion matrix of optimal architecture with industry data

		Predicted class		Total
		Fraud	Normal	
True class	Fraud	44	0	44
	Normal	227	11767	11994
Total		271	11767	12038

As already explained in Chapter 2, the accuracy measure may be negatively influenced by imbalanced classes. Due to the highly imbalanced data used in this study, the accuracy of 98.1% should be treated with care and not just blindly accepted as a good performance measure. In addition, the F1 score is a popular measure to determine a model's accuracy; however, working with imbalanced classes may also affect the reliability of F1 scores, and therefore F1 scores as accuracy measures are not pursued in this study. Furthermore, the low precision score should also not simply be accepted.

It is important to investigate trade-offs when measuring the effectiveness of fraud detection systems. In this case, the precision-recall trade-off is desirable. Within this trade-off, recall is favoured in contexts where there is a high cost associated with false negatives. An organisation evaluating such a model would have to consider whether capturing all the fraudulent instances is worthwhile for such a low precision (Al-Shabi, 2019).

The phenomenon of achieving a very low precision score to maximise recall, as well as the overall effectiveness of the model, was also found in a study conducted by Al-Shabi (2019). They achieved precision scores between 1% and 9% for the chosen autoencoders, while achieving recall and accuracy scores of up to 91% and 98.7%, respectively. By cycling through various thresholds, they were able to achieve an increase in recall, for a comparatively far smaller increase in the false-positive rate. The process highlighted the goal of the fraud detection process to detect as many fraudulent records as possible while keeping the false positive rate low.

When calculating the balanced score metrics on the test set, the model achieves an Area Under the ROC Curve score of 98.9%, a Matthews Correlation Coefficient score of 39%, and an Average Precision score of 15%, while the overall balanced score was 60.4%. Examining these underlying components of the balanced score further, the AP score brings the total balanced score down significantly, especially when compared with the AP score of 80.4% achieved on the validation set. This is to be expected, as the severity of the imbalance is different between the validation and test sets.

As covered in Section 5.2, there are 395 anomalous instances in the validation set, while there are only 44 anomalous instances in the test set. The magnitude of this difference is significant enough to warrant a dramatic decrease in the performance metrics most sensitive to the balance between the classes, namely AP and MCC. Thus, making a comparison between the balanced scores of the validation and test sets is a trivial exercise. The imbalance and subsequent sampling approach are taken to influence the scores significantly, especially considering that the validation set was purposefully loaded with far more anomalous records when compared with the test set. Ultimately, the test set performance assessment should rather focus on whether

the model generalised well to the unseen data, and whether the business objectives of the model were satisfied.

5.6.2 Shapley values for feature contribution

While the most important factor of the machine learning process is achieving good performance results with a trained model, another traditionally underspecified factor, namely model interpretation, is vitally important in determining trust in the model (Lipton, 2018). SHapley Additive exPlanations (SHAP) is an approach for assigning feature importance for a particular prediction (Lundberg and Lee, 2017) and is based on the Shapley value concept from the field of cooperative game theory.

The idea behind SHAP is to explain the prediction of an instance by computing the contribution of each feature to the prediction (Molnar, 2019). Example uses of Shapley values in alternative studies can be found in Janzing *et al.*, (2020) and Sundararajan and Najmi (2020). Lundberg and Lee (2017) propose that Shapley value explanations are represented as an additive feature attribution method that has an explanation model that is a linear function of binary variables:

$$g(\mathbf{z}) = \phi_0 + \sum_{j=1}^M \phi_j \mathbf{z}_j, \quad (5.5)$$

where $\mathbf{z} \in \{0, 1\}^M$ and is the simplified input of the explanation model, M is the number of simplified input features, ϕ_0 is the model output without the simplified inputs, and $\phi_j \in \mathbb{R}$.

Calculating the Shapley values requires retraining a model on all feature subsets $S \subseteq F$ where F is the set of all features. An importance value is assigned to each feature which represents the impact that the feature has on the model's predictions. This involves training a model $f_{S \cup \{i\}}$ with the feature included in the training set, and then training another model f_S with the feature left out.

The resulting predictions from the two models are compared using the input $f_{S \cup \{i\}}(x_{S \cup \{i\}}) - f_S(x_S)$, where x_S represents the values of the input features in set S . Removing a feature from a model influences the remaining features' impact on a model, and so the preceding differences are calculated for all possible feature subsets $S \subseteq F \setminus \{i\}$. The formula for the Shapley values, which is the weighted average of all of the possible differences, is represented as:

$$\phi_i = \sum_{S \subseteq F \setminus \{i\}} \frac{|S|!(|F| - |S| - 1)!}{|F|!} [f_{S \cup \{i\}}(x_{S \cup \{i\}}) - f_S(x_S)]. \quad (5.6)$$

The output of anomaly detection models can be difficult to explain, especially as these models return an anomaly score as the output, which can be misinterpreted out of context. Manual verification is typically used to validate the scores, although, without an explanation of the contributing factors that contribute to the decision around whether to class a record as anomalous, this validation process can be challenging. Shapley values are then used in this study to overcome this problem and to assist with explaining the contributing factors of the underlying models.

Antwarg *et al.*, (2021) showed success in implementing SHAP on the output generated by autoencoder models used for anomaly detection, where the connection between the features with high reconstruction error and those that are most important in affecting the reconstruction error is explicitly determined.

The process for explaining the autoencoder anomalies begins with determining the Shapley values for the *topMfeatures*, namely the top values from an ordered list of reconstruction errors per feature, as represented by Algorithm 5.2. The algorithm starts by extracting the features with the highest reconstruction error from *ErrorList*, an ordered list of reconstruction errors per feature, and placing them in *topMfeatures*. Then each feature x'_i within *topMfeatures* uses Kernel SHAP to determine the Shapley values.

Algorithm 5.2: Calculate Shapley values for the top M features (Antwarg *et al.*, 2021)

```

input :  $X$ ;                // the instance to be explained
         $X_1...X_j$ ;          // instances that Kernel SHAP uses
        ErrorList;         // an ordered list of error per feature
         $f$ ;                 // autoencoder model
output: shaptopMfeatures // Shapley values for each feature
        within top  $M$  features
topMfeatures  $\leftarrow$  top values from ErrorList
for each  $i \in \text{topMfeatures}$  do
    | explainer  $\leftarrow$  shap.KernelExplainer( $f, X_1...X_j$ )
    | shaptopMfeatures[ $i$ ]  $\leftarrow$  explainer.shapvalues( $X, i$ )
end
output shaptopMfeatures

```

Kernel SHAP is an approximation approach for determining Shapley values which uses a model-agnostic method known as LIME for explaining predictions, and Shapley values to build a local explanation model (Ribeiro *et al.*, 2016). Kernel SHAP is used, as it provides accurate estimates using fewer evaluations when compared with other sampling-based estimates. Within Algorithm 5.2, Kernel SHAP takes the autoencoder f and the background set with j instances to build the local

explanation model and calculate the Shapley values. Following on from the highest reconstruction error collection step, Algorithm 5.3 continues with predicting the values for the i^{th} feature using autoencoder f , taking X and i as input and returning the predictions X' . The output of this algorithm is a list of *shapTopMfeatures* where each row corresponds to the Shapley values for one feature from *topMfeatures*. Algorithm 5.3 is presented below and is considered the prediction operation within Algorithm 5.2.

Algorithm 5.3: Prediction of the i^{th} feature (Antwarg *et al.*, 2021)

```

input :  $X$ ;           // the instance to be explained
         $i$ ;           // the feature to get the prediction for
         $f$ ;           // autoencoder model
output:  $x'_i$          // the value of the  $i$ 'th feature in  $X'$ 
 $x'_i \leftarrow f.predict(X)[i]$ 
output  $x'_i$ 

```

After the prediction algorithm, the Shapley values are divided into values that either contributed to the anomaly or offset the anomaly. This algorithm is presented in Algorithm 5.4, where the algorithm begins with checking whether the predicted feature value is greater than the actual input value. If the predicted value is greater, then contributing features have positive Shapley values, while offsetting features have negative Shapley values. The converse is true if the predicted value is smaller than the actual value, where contributing features would have negative Shapley values while offsetting features would have positive Shapley values.

Algorithm 5.4: Divide Shapley values into those contributing and offsetting an anomaly (Antwarg *et al.*, 2021)

```

input : shapTopMfeatures; // Shapley values for each feature
         $X$ ;           // the instance to be explained
         $X'$ ;          // the prediction for  $X$ 
output: shapContribute, shapOffset
for each  $i \in \text{shapTopMfeatures}$  do
    if  $x_i > x'_i$  then
         $\text{shapContribute}[i] \leftarrow \text{shapTopMfeatures}[i] < 0$ 
         $\text{shapOffset}[i] \leftarrow \text{shapTopMfeatures}[i] > 0$ 
    else
         $\text{shapContribute}[i] \leftarrow \text{shapTopMfeatures}[i] > 0$ 
         $\text{shapOffset}[i] \leftarrow \text{shapTopMfeatures}[i] < 0$ 
    end
output shapContribute, shapOffset

```

A feature that is considered a contributing feature pushes the value away from the true value of the feature that the process is trying to explain, hence contributing to the anomaly. Conversely, a feature that is considered an offsetting feature pushes the value towards the true value of the feature that the process is attempting to explain using SHAP. This approach of distinguishing between the contributing and offsetting features is useful, as it helps in providing a visual representation of which features contributed to an anomaly or not. It is also useful when used in an operational context, as a domain expert using the anomaly detection system would be informed about which features contributed to or offset any anomaly detected by the system.

Figure 5.10 describes the top contributing features for the autoencoder trained on the original industry data set. Implementing these approaches on the two autoencoder models used in this study yields the highest contributing and offsetting features for each model. Based on Figure 5.10, it is evident that features *v8* (whether the user used an ID card as proof of identification), *v5* (whether the user has the highest level of membership on the platform and can access all transactional functionality without limits), *v13* (the number of failed attempts to load a credit card) and *v6* (the total time that the user spent in the queue to get his/her identity verified) are the biggest contributors when detecting anomalies in the industry data set. The top contributing features in Figure 5.10 should therefore be considered important for detecting anomalies in a fraud detection context.

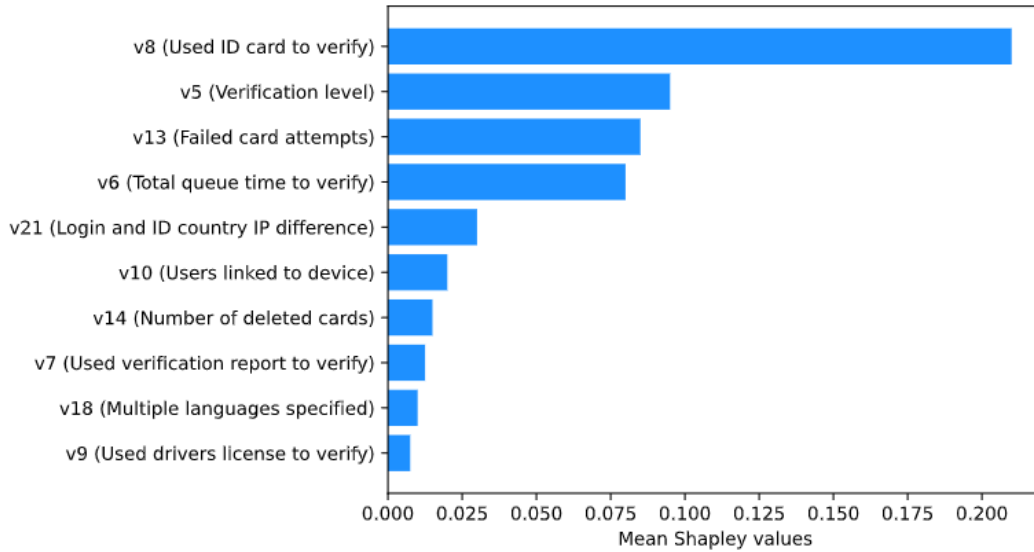


Figure 5.10: Shapley values for industry data

Figure 5.10 concludes the discussion and illustration of Shapley values in the real-world application of the proposed techniques. It is clear from the preceding discussion and results that the introduction of Shapley values was justified and greatly assist with the interpretability of the modelling process and provide insight into the importance of features used. Furthermore, the Shapley values established a certain degree of trust in the trained model and in the final framework proposed by the study. The next section will further elaborate on the results obtained with the real-world dataset.

5.7 Results and discussion

Applying the proposed unified approach to the industry data set provides a formal method for determining the optimal architecture for an autoencoder in a real-world setting. The best-performing architectures from Section 5.3 reported seemingly high balanced scores. The winning architecture achieved a balanced score of 89.7% and was comprised of 15 hidden layers with 17 347 tunable parameters.

The optimal threshold for this architecture on the validation set was an MSE of 5.353. When applied to the test set, the threshold generalised well, achieving recall and precision scores of 100% and 16.2%, respectively. Justification for the low precision in favour of a perfect recall score is presented in Section 5.6.1, with comparable studies yielding similar results. The threshold-moving approach applied to determine the optimal threshold is discussed by He and Ma (2013), although the authors merely used it on a single performance metric of choice, for example, the AUROC curve. The proposed balanced score approach provides a robust threshold-moving approach, using three critical performance metrics for imbalanced classes.

As discussed in Chapter 2, Section 2.3.6, accuracy is disregarded as an applicable performance metric, due to the accuracy paradox that arises with imbalanced classes. Other traditional classification metrics, such as precision, recall and F -score, are also disregarded in favour of the metrics that comprise the balanced score. This is specifically about the threshold optimisation process. Following the application of the threshold on the test set, the balanced score metrics can again be used to assess whether the model generalised to the test set, although as is highlighted in Section 5.6.1 the additional imbalance imposed by the sampling strategy would by definition favour the validation set balanced score due to the higher relative proportion of the minority class.

The results from evaluating the optimal architecture using industry data are good. The optimal architecture correctly classified all the fraudulent users in the test set, while only recording 227 false positives. In an operational setting, detecting so many anomalies at the expense of incorrectly classifying so few normal records is very

useful. Misclassification carries a cost, with misclassified anomalies carrying the cost of the anomalous event, *i.e.*, a fraudulent transaction is honoured and the repercussions are realised by the payment provider or the organisation itself. Misclassifying normal records means the time taken by a fraud analyst to review the classification is wasted. Achieving a good balance between these two misclassification rates is crucial in developing a functioning anomaly detection system.

Apart from the traditional performance metrics employed to evaluate the models formulated, several practical considerations need to be taken into account before implementing a specific instance of a model. Some of these practical considerations include ease of constructing the model, model runtime, model complexity, and expertise required to implement and interpret the model. These considerations will be revisited in Chapter 6.

The proposed unified approach was validated against a real-world data set, with the performance metrics and model interpretation via Shapley values confirming that the approach is indeed effective in detecting anomalies in a transactional setting. Four variables were identified as being significant in detecting anomalies, based on their Shapley values. These are:

- whether a user used an easy to forge identity document to sign up to the platform;
- whether he/she was verified to the highest possible level (and can therefore transact with larger amounts);
- whether he/she had been involved in card testing activity and therefore had many failed card attempts; and
- finally, whether the user progressed through the signup process swiftly (an indication that he/she signed up with multiple fictitious accounts before).

It is therefore recommended that when working with credit card transactions using an autoencoder that at least these features are included if feasible to do so.

5.8 Chapter conclusion

This chapter was the first chapter in Part III and the purpose was to apply the proposed unified approach proposed in Chapter 4 to a real-world data set taken from an industrial setting. The data set used was sourced from an industrial organisation and was used to train a sparse autoencoder to detect anomalies in the data.

The next chapter, which is the second chapter of Part III, will present an extension of the unified approach that was applied. This will illustrate how network

metrics can be calculated, as well as how they can be used as features for a machine learning model. The intention is to illustrate how network metric features can be used to improve the performance of the unified approach.

Chapter 6

Social network analysis extension of the unified approach for anomaly detection

6.1 Chapter introduction

Chapter 6 is the second chapter of Part III (Model extension and real-world validation) of the research study. The chapter proposes an extension to the unified approach that was presented in Part II (Chapter 4) and validated in Chapter 5. The proposed extension involves using network data sourced from the same industrial organisation referenced in Chapter 5, to understand whether the network data can enhance the unified approach's ability to detect anomalies in a transactional setting.

To begin with, an explanation of how social network analysis (SNA) can be used to model users in a transaction-based network is given. It involves the application of the various social network analysis metrics presented in Chapter 3, such as degree centrality, PageRank, *etc.*, on the real-world network, data set. These network metrics assist with determining the most important vertices within a graph, which ultimately helps to identify the impact that discovered anomalies might have on a network. The computed metrics are then used to augment the standard transactional features, to determine whether the network features contribute to improved model performance. The first pass of the proposed unified approach made in Chapter 5 is compared with a second pass made in this chapter, concluding with a comparative discussion of the results obtained from each pass of the standard and augmented features respectively.

The remainder of the chapter is structured as follows: in Section 6.2, the network data set is introduced and the calculation of various network metrics is included. In Section 6.3, the proposed unified approach is applied, including the addition of the network metrics determined in Section 6.2. Neural architecture search, threshold optimisation, and Gaussian scaling are applied to the optimal architecture. In Section 6.4 the performance of the optimal architecture is determined, using classification metrics, followed by the determination of the Shapley values to understand feature contribution. Section 6.5 then follows, in which the results of the chapter are presented, and a discussion of the practical considerations and recommendations, based on the findings are included. Lastly, the chapter is finalised with a conclusion of the results in Section 6.6.

6.2 Description of the network data

The data used to construct a graph representation of user activity has been sourced from an organisation operating in the financial services industry. The data contains all information that can be used to link users to each other within the industry partner’s transactional ecosystem. In constructing the graph, users are represented by the nodes in the graph while the edges represent the events that link the users together. Examples of edges include users sending payments to each other within the ecosystem, users with the same IP address, the same unique device, the same payment details, or the users who are linked via a promotional event, such as a coupon.

It should be noted that in this particular data set and how the network will be constructed, there is no guarantee that users within the original industry data set would be within the network data set. A user will only appear in the network data set if he/she has one or more of the links mentioned above to another user, therefore it is expected that the network data set will be somewhat smaller in terms of how many unique users are represented when compared with the original industry data set. Of the 120 676 users within the original industry data set (the full population of users within the particular segment of concern) only 27 361 users have links within the network data set. The seemingly low number of links is deemed to be acceptable, as the network data is meant to augment the main industry data. It is expected that most fraudulent users are likely to be connected in some way and that the majority of them would have links in the network data set from which potentially useful machine learning features can be inferred.

6.2.1 Introducing the network metrics

The twelve network metrics identified in Chapter 3 are listed below in Table 6.1. They are all conveniently available through the *NetworkX* library (Hagberg *et al.*, 2008). *NetworkX*, a Python package used for the creation, manipulation, and study of the structure, dynamics and functions of complex networks, was used to transform the tabular source data into a graph upon which network metrics may be determined.

Table 6.1: Description of the original network metrics

Feature	Description
Degree centrality	The fraction of nodes to which a node v is connected
Eigenvector centrality	The eigenvector centrality of a node, based on the centrality of its neighbours, and measures the influence of a node v
Closeness centrality	The reciprocal of the average shortest path distance from a node v to all other nodes
Betweenness centrality	The sum of the fraction of all pair shortest paths that pass through a node v
Load centrality	The fraction of all shortest paths that pass through a node v
Harmonic centrality	The sum of the reciprocal of the shortest paths from a node v to all other nodes
Average neighbour degree	The average number of connections of the nodes within a neighbourhood, relative to a node v
Clustering coefficient	The fraction of possible triangles through a node v
Node clique number	The size of the largest maximal clique containing a node v
Number of cliques	The number of maximal cliques for a node v
Square clustering coefficient	For each node v , the fraction of possible common neighbours
PageRank	The ranking of each node v , based on the structure of the incoming links

The metrics themselves cover a wide variety of SNA-related perspectives and considering that there is no literature suggesting that certain metrics are superior to others when it comes to detecting anomalies, the decision was made to include all twelve network metrics.

6.2.2 Calculating the network metrics

To calculate network metrics, the data needs to be represented as a graph G . Considering that the originating data is sourced from a relational database, this form of data needs to be reconfigured, using a graph database engine. Following this, existing instances of anomalies may be mapped onto the graph, so that the distances between non-anomalous and anomalous nodes can be determined. Mapping the anomalies involves traversing the entire graph structure to determine the shortest path between the relevant nodes of interest. Finally, network metrics may be calculated once the data is in the appropriate structure for performing said network metric computations.

The structure of the network data set is as follows. There are target, source, edge, and edge attribute fields in the data set, as depicted in Table 6.2.

Table 6.2: Source network data for first five edges

	Target	Source	Edge	Edge Attribute
1	81907060409976	4342919630908	dd04700628e1b4c454b4a	device
2	51012824933480	8583623832224	4555e3a4406a35b6493c5a	device
3	51012824933480	8003234156155	4555e3a4406a35b6493c5a	device
4	80032341561555	5101282493348	9495e0f1ea2ffd388bc445e	device
5	53853862801439	4342919630908	d5c1e1d28a1054fb3e6d14	device

The source denotes the user where the edge event originates, *i.e.*, the graph has a directional element that flows from source to target. The target is therefore where the edge flows to. Finally, the edge attribute merely describes the edge, *i.e.*, it would be called *IP address*, *device* or *credit card*.

Constructing the graph

The tabular data represented in Table 6.2 contains all the source data required to create a graph, namely the source and target nodes n , as well as the edges m that represent the connections between them. The network data was extracted from the industrial organisation's *Amazon Redshift* database, which required writing several SQL queries and appending them together so that their output matches the form that the *NetworkX* library expects. The process begins with initialising a graph object in *NetworkX*, followed by an operation whereby the tabular data is fed into the graph object, yielding a populated graph G . Once the graph object has been initialised, all the network metrics functions specified in Table 6.1 can be directly calculated.

Mapping bad actors on the graph

In addition to the network data, labels have been provided that describe whether each user in the network data set is either confirmed as fraudulent (the same features as the transactional industry data set), as well as whether that user’s profile is currently locked for whatever reason (generally a sign of some negative behaviour) within the partner’s ecosystem. These values can then be mapped onto the nodes in the network to calculate network metrics related specifically to a node’s position about bad actors in the network.

The first two network features listed in Table 6.3 are reliant on the industry organisation’s internal labelling of the users in the data set as being fraudulent or having an account that is in a locked state. A locked state is defined as the state when an account has been closed and all its associated functionality has been removed. These states can be temporary or permanent, depending on the severity of the suspicious activity associated with the account. A user that is marked as fraudulent would always have a locked account, but a user with a locked account does not always imply that the user is indeed fraudulent. There are several reasons why a user might be locked, most of which are an indication of a previous negative event linked to the user. The assumptions with these features are that users being closely linked to other users with a history of negative behaviour could be a potential indication of fraud for the user in question.

Table 6.3: Description of the manual network features

Feature	Description
Distance to closest fraudulent user	Hops to the nearest fraudulent node v in the graph G .
Distance to closest locked user account	Hops to the nearest locked node v in the graph G .

Shortest paths between nodes

To traverse the entire network, an efficient algorithm is needed. The requirement for an efficient algorithm is especially true when the network is larger than a few thousand nodes in scale. One common algorithm for determining the shortest path between nodes in a network is Dijkstra’s algorithm (Dijkstra, 1959) which is listed below as Algorithm 6.1.

Alternative algorithms for calculating shortest path distances exist, such as the minimal-geodesic distance algorithm (Balasubramanian *et al.*, 2009). While the geodesic algorithm may produce a more exact shortest-path distance calculation when compared with the approximation produced by Dijkstra’s algorithm, the longer time

it takes to compute the geodesic approach may not be feasible in the context of a real-world anomaly detection problem. Rather, it makes sense to use the faster, approximate solution provided by Dijkstra's algorithm, considering that there are 27 361 nodes in the graph.

Dijkstra's algorithm starts with a graph G and is run from the perspective of a source node s . The algorithm then loops through each node v in graph G , where it assigns an initial distance of infinity from the source node s and the vertex v in the variable $dist[v]$. $prev[v]$ represents the pointer to the previous node in the shortest path from source s to node v and is initialised to null. Where node v is not equivalent to the source node s , then v is added to a priority queue Q . Following this, the distance for the source node s is set to 0 so that it is picked first. A secondary loop is then initialised while the priority queue Q is not empty. The node v with the smallest distance is then extracted from Q and added to u . For each of the unvisited neighbours v within u , if the sum of the distance value from source s to u and the weight of edge (u, v) is less than the distance from the source s to v , then the distance value of v is updated.

Algorithm 6.1: Dijkstra's algorithm (Dijkstra, 1959)

```

input :  $G$ : graph,  $s$ : vertex
output:  $dist[ ]$ ,  $prev[ ]$ 
for each vertex  $v \in G$  do
     $dist[v] \leftarrow infinity$ ;
     $prev[v] \leftarrow NIL$ ;
    add  $v$  to  $Q$ ;
end
 $dist[s] \leftarrow 0$ ;
while  $Q \neq \emptyset$  do
     $u \leftarrow$  vertex in  $Q$  with minimum  $dist[u]$ ;
    remove  $u$  from  $Q$ ;
    for each neighbour  $v$  of  $u$  do
         $alt \leftarrow dist[u] + \text{length}(u, v)$ ;
        if  $alt < dist[v]$  then
             $dist[v] \leftarrow alt$ ;
             $prev[v] \leftarrow u$ ;
        end
    end
end
output  $dist[ ]$ ,  $prev[ ]$ ;

```

Visualising the network

In this section, a series of illustrations is presented to understand the structure of the network data. Due to the size of the clusters of network data, only certain visual examples will be given. Firstly, a smaller cluster with predominantly normal nodes is shown, highlighting the normal and locked (fraudulent) nodes, as well as the edge connections between them (Figure 6.1). Following this, another small cluster with predominantly locked nodes is depicted in Figure 6.2.

Another small cluster (Figure 6.3), with a mix of anomalous, locked and normal nodes is then presented. Finally, the series concludes with diagrams of the largest cluster in the overall network (Figure 6.4), as well as the entire network as a whole (Figure 6.5).

Figure 6.1 is a graphical representation of a small graph or cluster of linked accounts within the overall network of linked accounts. The green nodes are normal accounts, whereas the orange node is in a locked state, and this node represents a potentially anomalous account. The edges in this graph are comprised of IP addresses and device token links, where a device token indicates a unique token that is generated per unique device used on the industrial organisation's platform. If two accounts are linked via the device token, then that means that both the accounts were accessed via the same physical device which is a potential indicator of suspicious activity.

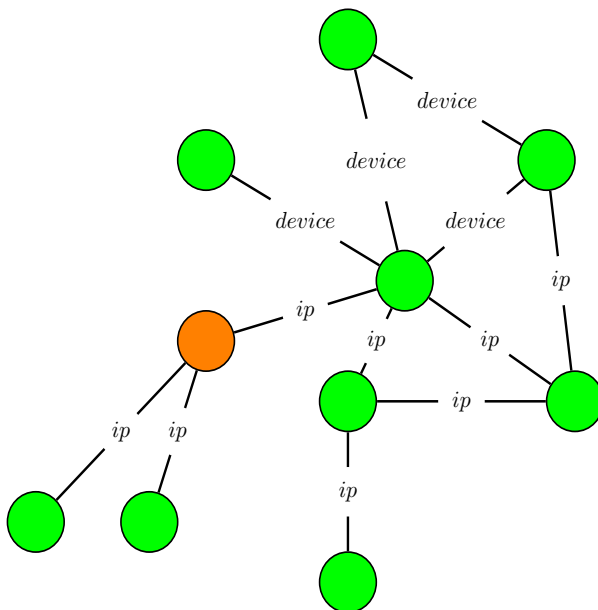


Figure 6.1: Small cluster with predominantly normal nodes

The diagram shows a sequence of nodes connected by lines. The nodes are colored orange and green. Labels 'ip' and 'device' are placed near the connections. The diagram illustrates a sequence of nodes connected by lines, with labels 'ip' and 'device' indicating the type of connection or node.

Figure 6.2: Small cluster with predominantly locked nodes

Another small cluster is presented in Figure 6.3, where the cluster is comprised of normal, locked, and confirmed anomalous accounts. The anomalous account is depicted by the red node in the cluster. Again, the links in the cluster are predominantly IP address- and device token-related. Other link types, such as shared credit card details and transactions between nodes occur less frequently. In practice, an analyst might investigate the otherwise normal accounts directly linked to the locked and anomalous nodes.

The largest cluster in the entire network is presented in Figure 6.4. The large cluster is a densely connected web of anomalous, locked, and normal accounts, and contains 1 515 distinct nodes, including eight instances of fraud, as indicated by the large red nodes. Some segments of the cluster splinter off into smaller, yet connected clusters, while the majority of the remaining nodes form a dense mesh of hundreds of accounts. While alarming, this cluster would be difficult to manually investigate.

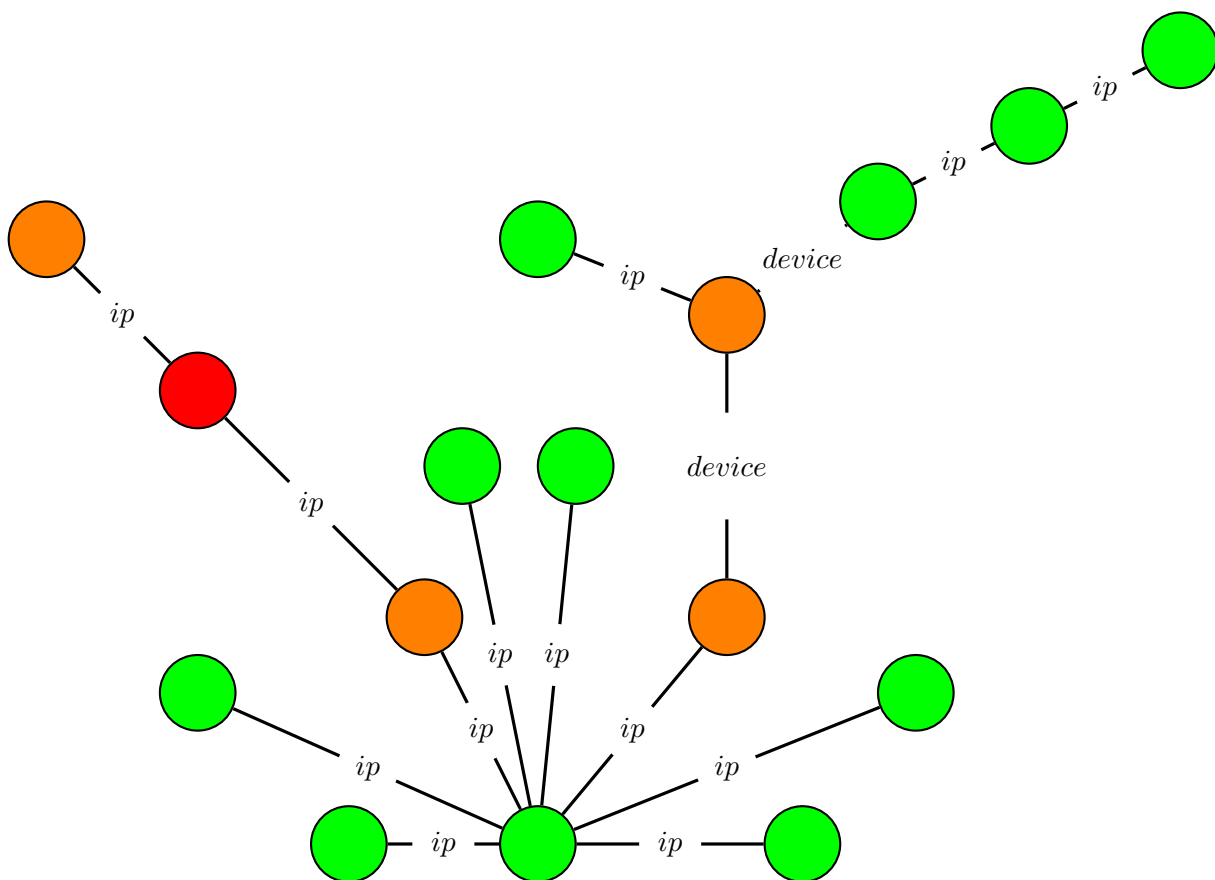


Figure 6.3: Small cluster with a mix of anomalous, locked and normal nodes

Generating network metrics that describe the positions of these nodes within the greater cluster would assist with surfacing suspicious links. These findings are confirmed by Sanni and Nurcan (2015) who found that network metrics save time during a manual investigation, which is a necessary step when validating any potential fraud cases uncovered by a tool. Sanni and Nurcan (2015) add that traditional fraud detection systems are transaction-oriented. These transactional approaches, therefore, output rows of the suspicious transactions without providing a view of the other transactions or events performed between multiple entities. With network analysis, the entities or users involved, as well as their overall activities, are available in a graph view. The graph view allows manual investigators to visually identify false positives, allowing them to move on to focus on more risky cases.

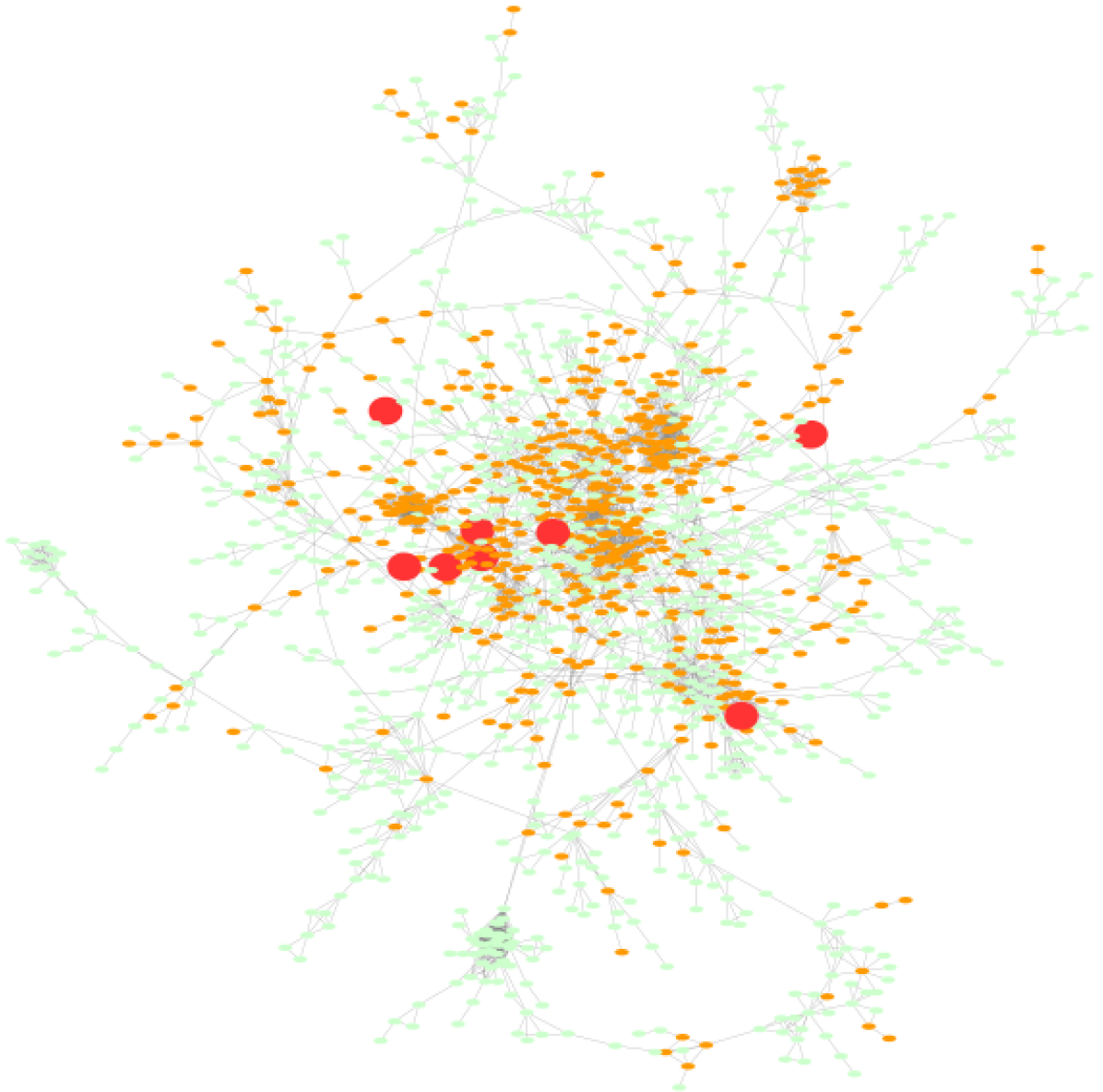


Figure 6.4: Largest cluster in the overall network. The red nodes indicate confirmed instances of fraud, while the orange and green nodes indicate locked accounts and non-fraudulent accounts respectively.

Figure 6.5 presents the overall network as a whole. In total, the network contains 27 361 nodes and 34 957 edges, where the edges include IP addresses, as well as shared credit cards and devices.

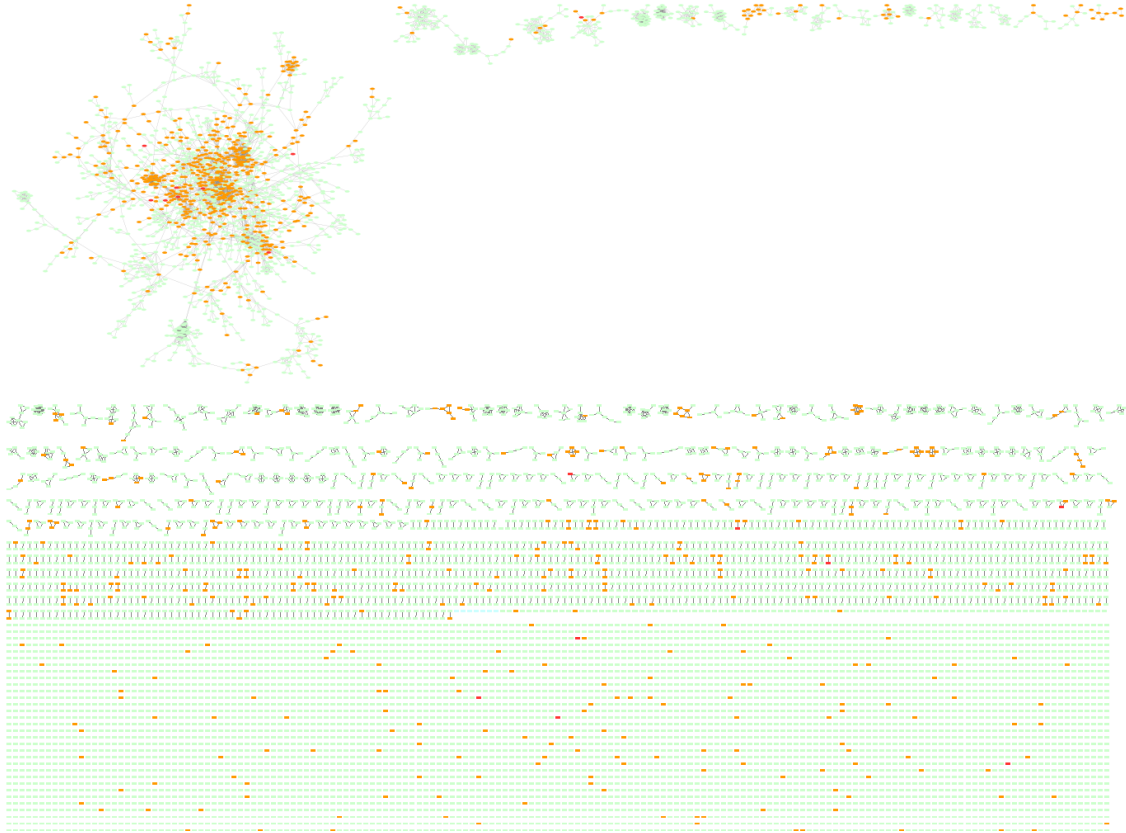


Figure 6.5: The entire network. The largest cluster in the network is found in the top left corner, with the clusters becoming smaller towards the bottom of the figure.

The large cluster (Figure 6.4) and some of the smaller clusters (Figures 6.1 through 6.3) are all positioned at the top of the network. The very small clusters, namely those with a few to only a single node within them, are positioned towards the bottom of the graph. The overall view sorts the network clusters from biggest to smallest, with the smallest clusters positioned at the bottom of Figure 6.5.

Description of final network metric features

Following on from the determination of the closest bad actors to each node within graph G , the network metrics are converted into features and used to augment the standard industry data to potentially improve the performance of the optimal autoencoder architecture selected via NAS. The final set of network features is presented in Table 6.4. The table includes both the feature names, as well as a description of each.

Table 6.4: Description of the final network features

Feature	Description
Distance to closest fraudulent user	Hops to the nearest fraudulent node v in the graph G
Distance to closest locked user account	Hops to the nearest locked node v in the graph G
Degree centrality	The fraction of nodes to which a node v is connected
Eigenvector centrality	The eigenvector centrality of a node, based on the centrality of its neighbours, and measures the influence of a node v
Closeness centrality	The reciprocal of the average shortest path distance from a node v to all other nodes
Betweenness centrality	The sum of the fraction of all pair shortest paths that pass through a node v
Load centrality	The fraction of all shortest paths that pass through a node v
Harmonic centrality	The sum of the reciprocal of the shortest paths from a node v to all other nodes
Average neighbour degree	The average number of connections of the nodes within a neighbourhood, relative to a node v
Clustering coefficient	The fraction of possible triangles through a node v
Node clique number	The size of the largest maximal clique containing a node v
Number of cliques	The number of maximal cliques for a node v
Square clustering coefficient	For each node v , the fraction of possible common neighbours
PageRank	The ranking of each node v , based on the structure of the incoming links

The features were standardised in the same fashion that the original industry features were standardised, to exclude any potential challenges that might arise through the differences in the scale between the new network features and the original industry data features. Should a network metric not have been available for any rows in the data, then the corresponding feature that represents the underlying network metric would contain a value of zero. There is no guarantee that network metrics would be available for all users in the industry data set, as they might simply not have behaved in a manner that warrants the collection of network data points.

The computed features for the first five nodes in the network are displayed in Table 6.5. There are 27 361 nodes in the network, therefore due to the volume, only the first five node values are illustrated. Table 6.5 demonstrates that the magnitude of the raw metrics is quite different between certain features. The table also contains several zero values, which describe the sparsity that exists within the feature set.

Table 6.5: Raw network metrics for first five nodes

	Node 1	Node 2	Node 3	Node 4	Node 5
Distance closest fraud user	0	3	3	1	0
Distance closest locked user	0	1	1	1	0
Degree centrality	0.073	0.183	0.146	0.256	0.146
Eigenvector centrality	1.235e-25	6.031e-07	6.018e-07	2.927e-06	2.590e-19
Closeness centrality	0.000	0.769	0.769	0.917	0.073
Betweenness centrality	0.000	0.004	0.000	0.001	0.000
Load centrality	0.000	0.004	0.000	0.010	0.000
Harmonic centrality	0.000	251.351	250.851	298.618	2.000
Average neighbour degree	1.000	5.400	6.000	9.571	3.000
Clustering coefficient	0.000	0.333	1.000	0.600	1.000
Node clique number	1	3	3	4	3
Number of cliques	1	2	1	2	1
Square clustering coefficient	0.000	0.636	6.000	0.074	0.667
PageRank	0.037	0.004	0.026	0.002	0.003

In Table 6.6, the network features are scaled and joined to the original industry data set to form the expanded industry data set including network metric features. During the feature engineering stage, care was taken to perform the sampling process before any feature scaling processes to prevent data leakage from occurring. Leakage is the use of information during model training that would not be expected to be available at prediction time (Kaufman *et al.*, 2011). By scaling the data before splitting the data into training, validation and test sets, the validation set would inherit some of the information from the training set. Data leakage is best avoided

by first splitting the data set into the required subsets, then performing feature scaling on the training set, and finally transforming the validation and test sets via the scaling function that was previously fit on the training set.

The features in Table 6.6 represent an abstraction of the data following the scaling process and are taken from the industry data set including the network metrics. Therefore, the smaller network set that was joined onto the original industry data set end up being sparsely populated.

Table 6.6: First 25 samples of industry data network metrics features

0	1	2	3	4	5	6	7	8	9	10	11	12	13
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.3	0.1	0.1	0.0	0.7	0.0	0.0	0.6	0.3	1.0	0.2	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1
0.2	0.1	0.1	0.1	0.7	0.0	0.0	0.6	0.2	1.0	0.1	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

The sparseness is due to the relatively smaller number of users in the network data set when compared with the full set of users in the original industry data set. There are fewer users in the network data set because not all users have a connection to another user. Due to the nature of the edges, being connected to another user is somewhat suspicious in itself, and therefore it is expected that a small proportion of the total user group is represented in the network.

Latent space representation using t-SNE

Applying t-SNE to the industry data following the inclusion of the network features yields a structure that is very similar to that presented in Figure 5.1 in Chapter 5. The output of the t-SNE application is presented in Figure 6.6. Upon careful inspection, all the instances of fraud are near the red cluster, whereas a few fraudulent instances were visible amongst the normal green clusters in Figure 5.1.



Figure 6.6: Visualising latent space using t-SNE on industry data with network features

These instances are evidence that the introduction of the network metric features has contributed to the level of discrimination between the fraud and normal classes in the industry data set, at least from the perspective of the t-SNE algorithm.

6.3 Unified approach including network metrics

In Section 6.3, the network modelling metrics developed in Section 6.2 are carried forward and the question is asked whether these metrics could be significant features and therefore improve the performance of the proposed unified approach applied in Chapter 5. The network metric features are to be included in the standard industry data feature set, and a second pass of the proposed unified approach is made to determine whether network metrics could be used as informative features for detecting anomalies in a transactional setting.

6.3.1 Neural architecture search including network features

Simulating 100 architectures, yields the following results, as per Figure 6.7. The best architecture generated, following the inclusion of the network metrics features, achieved a balanced score of 94.07% (indicated by the red bar) compared with the best architecture score of 89.76%, achieved by using only the standard industry data.

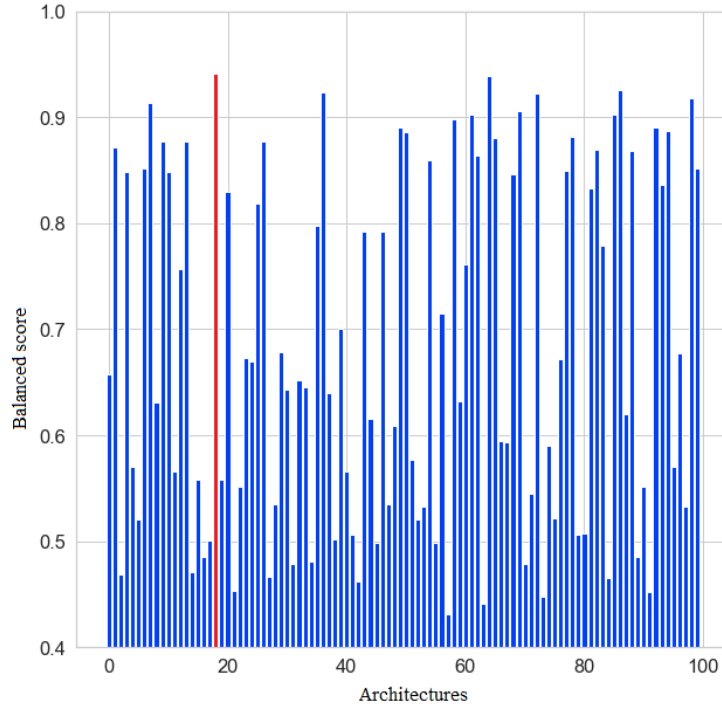


Figure 6.7: Simulating 100 architectures with NAS including network features

The average balanced score across the 100 simulated architectures, however, decreased from 70% when using the standard industry data to 68.1% after introducing the network metrics. More variability across the 100 simulations when including the network metric features is likely due to the same upper threshold placed on the first hidden layer generated during each simulation. To keep the experiment fair and comparable, the maximum size of the first hidden layer was bound to the same size as the input dimension of the standard industry data, *i.e.*, 47 neurons. The data set including the network metrics had an input dimension of 61, and this guaranteed drop-off from 61 down to 47 or below neurons in the first hidden layer is likely to have caused much of the variation in the balanced scores derived from the network metrics set. Details of the top 20 architectures using industry data with network metrics are presented in Table 6.7.

Table 6.7: Details of top 20 architectures using industry data with network metrics

AP	AUC	MCC	Balanced score	Architecture	Params	Threshold
0.885	0.997	0.938	0.940	[3,23,34,41,42]	13402	4.242
0.883	0.996	0.937	0.939	[18,19,22,26,28,38,43,46]	19607	3.636
0.858	0.995	0.923	0.925	[12,21,24,25,38,41]	13101	3.737
0.853	0.996	0.920	0.923	[5,7,12,17,19,26,39]	9372	4.242
0.850	0.995	0.919	0.921	[4,16,23,30,34,41,45]	16695	4.242
0.844	0.994	0.915	0.918	[3,4,27,31,35,40,43]	15994	4.343
0.835	0.996	0.910	0.914	[13,16,17,22,27,35]	9898	3.131
0.819	0.995	0.901	0.905	[8,10,14,19,22,27,35]	9479	4.343
0.814	0.995	0.898	0.902	[3,7,11,20,33,38,46]	13946	4.444
0.812	0.995	0.897	0.902	[4,12,15,22,25,26,34]	9765	4.242
0.805	0.994	0.893	0.897	[4,14,19,24,29]	6723	3.939
0.791	0.995	0.885	0.890	[8,13,15,17,20,29,35]	9575	3.939
0.790	0.995	0.884	0.889	[3,24,29,30,43]	11418	3.737
0.786	0.993	0.882	0.887	[3,8,18,23,26,29,32,42]	13956	4.343
0.782	0.994	0.879	0.885	[4,12,16,18,20,22,26,32]	9725	4.040
0.774	0.994	0.875	0.881	[3,8,18,22,25,29,31,39,45]	17392	4.242
0.775	0.989	0.876	0.880	[6,22,33,42]	9873	2.424
0.766	0.994	0.870	0.877	[6,9,15,20,30,35,36]	11547	3.535
0.765	0.994	0.870	0.876	[4,5,27,32,35,38,46]	16477	4.040
0.766	0.993	0.870	0.876	[5,7,10,11,12,19,22,38]	8598	4.444

Regardless of this variability, the best architectures derived from the inclusion of the network metrics NAS process comfortably surpass those from the standard industry data. It is possible that this variation also contributed to achieving the better-balanced scores in the network metrics simulation, in that the drop-off from

61 down to 47 or below sometimes contributed positively to the balanced score, depending on the design of the rest of the architecture.

The optimal architecture obtained during the NAS process can be visually displayed using *VisualKeras* (Figure 6.8), a tool for visualising Python-based neural network architectures (Gavrikov, 2020). Analysing the figure from left to right, the yellow bar on the far left indicates the input layer of the encoder. Following that, there are four hidden layers depicted in red. In the centre of the architecture, a small red block indicates the bottleneck, which is then followed by the decoder which is a completely symmetrical version of the encoder.

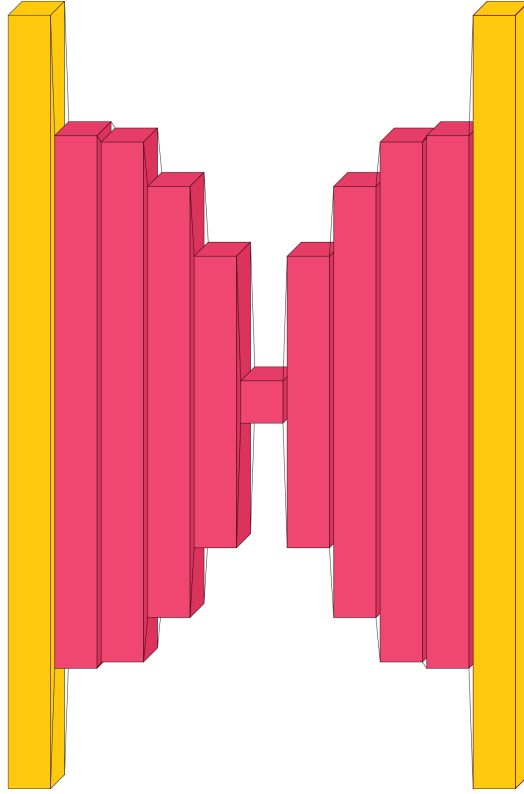


Figure 6.8: Visual depiction of the optimal architecture

With the neural architecture search completed, the next step in the proposed unified approach is the threshold optimisation step which is presented in the next section.

6.3.2 Threshold optimisation using industry data with network features

The optimal threshold value for the best performing architecture as determined by the NAS process is 4.545 when measured against the test set. The threshold is lower than the threshold of 5.85 determined when using the industry data before the introduction of the network metrics features. When compared with the best performing architecture in Chapter 5, Section 5.3, the optimal architecture achieved a balanced score of 94%, as opposed to a balanced score of 89.7%.

The determination of the optimal threshold on the test set exhibits a similar pattern to that shown in Chapter 5, Section 5.4, although the optimally balanced score is higher and the threshold lower after introducing the network metrics features, as illustrated in Figure 6.9.

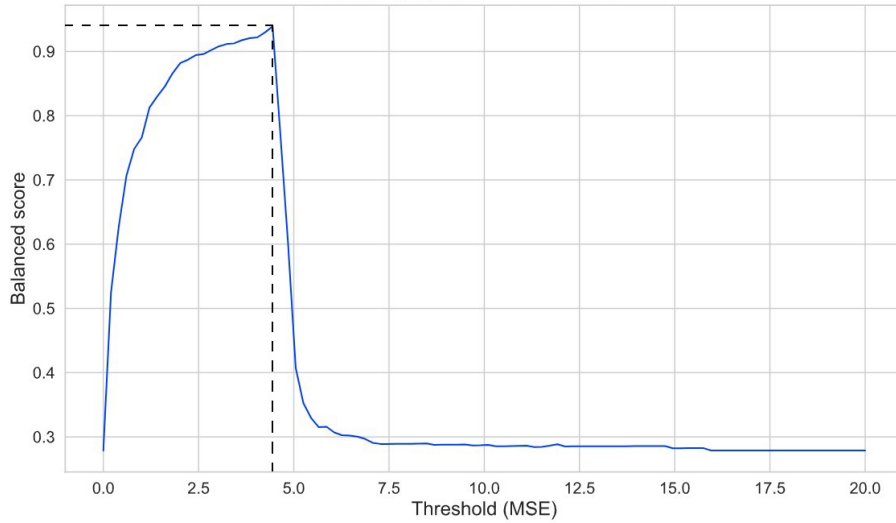


Figure 6.9: Optimising industry data with network features threshold selection using the balanced score

The optimal validation set threshold value of 4.545 is shown as the dark blue line in Figure 6.10, where extreme outliers are again capped at a maximum MSE of 20.

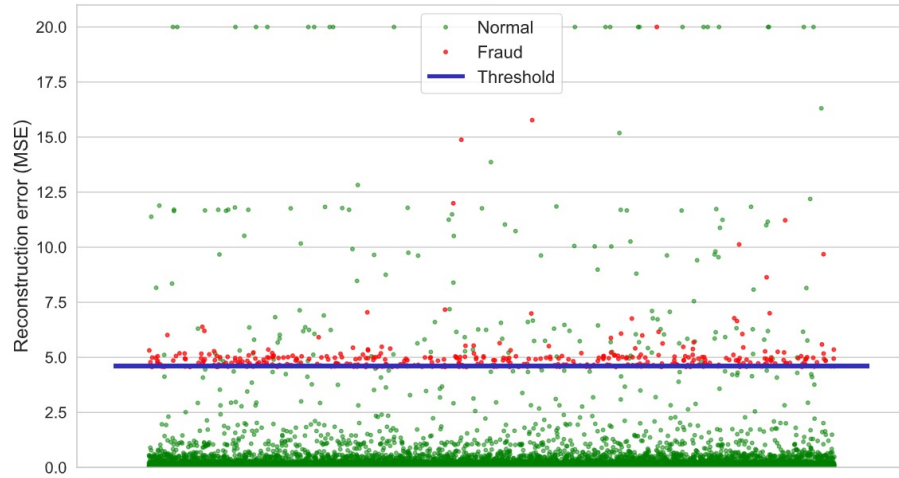


Figure 6.10: Reconstruction errors (MSE) using industry validation data with network features

When applying the optimal threshold on the unseen test set, the model captures all the fraudulent records, where all the red data points sit above the dark blue threshold line in Figure 6.11.

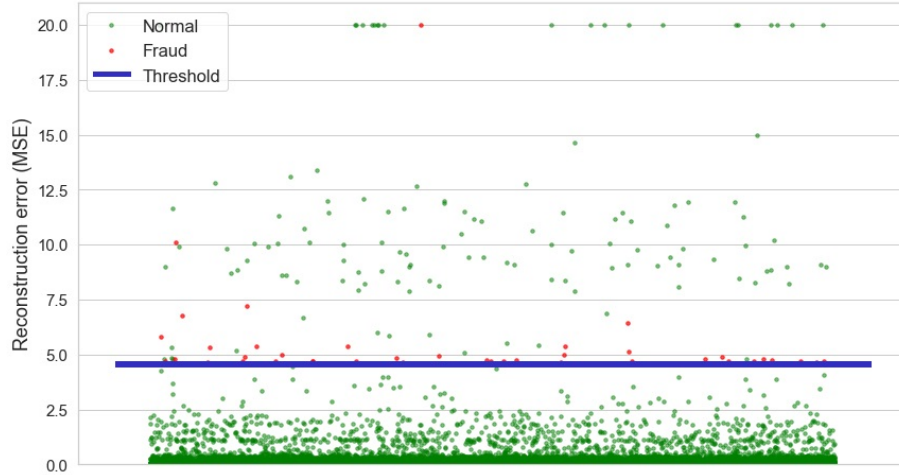


Figure 6.11: Reconstruction errors (MSE) using industry test data with network features

While the majority of the normal records, as indicated by the green data points, sits below the dark blue threshold line, some do sit above the line. With the threshold

optimisation step completed, the final step in the proposed unified approach is the Gaussian scaling step which is presented in the next section.

6.3.3 Applying Gaussian scaling on the industry data with network metrics scores

The predicted outputs from the optimal architecture are MSE values or reconstruction scores that indicate whether each user is likely to be a fraudster. These classifications are determined by the threshold set in Section 6.3 for the combined feature set. As discussed in Section 4.5 in Chapter 4, the reconstruction scores are not easily interpreted. Applying a scaling function can assist in scaling the scores into a range of $[0;1]$. The scaled scores can then be considered as the likelihood that a user is fraudulent.

Following the introduction of the network metric features, linear scaling was again applied to transform these scores into an interpretable format. The linearly scaled scores are presented in Figure 6.12.

While it is easy to discern between the two classes after linear scaling, the bulk of each class is closely related to one another. Using linear scaling would make it difficult to use operationally, where practitioners would expect to use realistic anomaly thresholds, such as an above 90% probability threshold to detect anomalies.

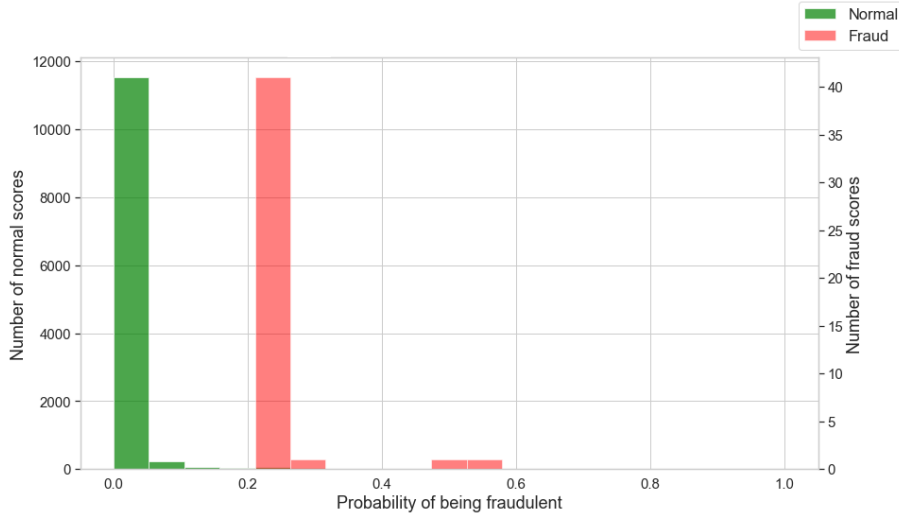


Figure 6.12: Anomaly scores with linear scaling for industry data set with network metrics

As presented in Algorithm 4.4 in Chapter 4, Gaussian scaling helps transform the raw scores into a more interpretable form. The application of Gaussian scaling is depicted in Figure 6.13. There is now very little overlap between the classes, except for some of the normal transactions being classified as fraudulent. There are also a few cases of normal scores hovering around the 40% to 50% mark, although the vast majority of the normal scores is located in the 0% to 10% score range.

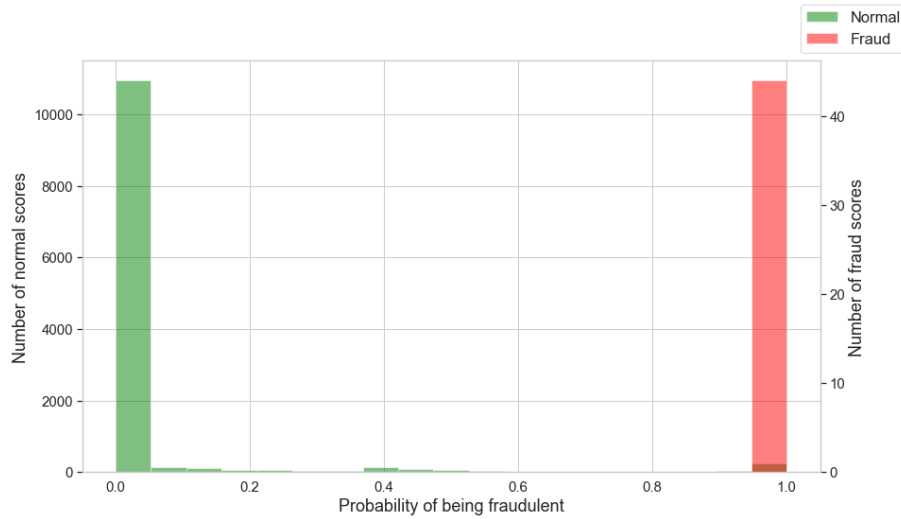


Figure 6.13: Anomaly scores with Gaussian scaling for industry data set with network metrics

The Gaussian scaled scores from both models, without network features (Figure 5.9) and with network features (Figure 6.13) are in the acceptable range for operational use, considering that most of the normal transactions are scored in the 10% or lower bucket, whereas all the fraudulent transactions are issued a score of 90% or higher. When comparing the two models directly, the second model, which includes the network metrics, records fewer false positives and would therefore be considered more effective for operational use.

This third step concludes the unified approach for the data set including the network metrics. Extending the unified approach through the inclusion of the network metrics resulted in an optimal autoencoder architecture with a higher balanced score than that obtained in Chapter 5. The resulting features are also more expressive of the underlying fraudulent behaviours, which extend beyond the transactional domain. The extended feature set includes network features that explain how anomalous users are linked together.

6.4 Evaluating performance of optimal architectures

In Section 6.4, the starting point is an analysis of the performance metrics for the model including network metrics developed in this chapter. The SHAP approach introduced in Chapter 5 is then applied to the model developed in this chapter to identify which features are significant contributors to detecting anomalies. Finally, a comparison between the SHAP values computed in Chapter 5 and this chapter is made to determine if any network metric features made a significant contribution in detecting anomalies.

6.4.1 Performance metrics

The confusion matrix presented in Chapter 2, Table 2.1, is used to measure the performance of the model from a binary classification perspective. Following the introduction of the network metrics features, the optimal architecture was able to correctly classify all the fraudulent users in the unseen test set, while misclassifying only 125 of the confirmed normal instances. Therefore, this results in a precision score of 26% and a recall score of 100%. The resulting confusion matrix for the optimal architecture using the industry data including network features is depicted in Table 6.8.

Table 6.8: Confusion matrix of optimal architecture including network features

		Predicted class		Total
		Fraud	Normal	
True class	Fraud	44	0	44
	Normal	125	11869	11994
Total		169	11869	12038

Making a direct comparison between the results of the confusion matrix for the transactional data (Table 5.3 in Chapter 5), and the confusion matrix that includes the network features (Table 6.8), it is clear that the outcomes of the classification process are very similar, with one key exception. That exception is that the number of false positives, namely the number of normal transactions incorrectly classified as fraudulent, drops from 227 for the model with the industry data to 125 for the model that includes the additional network features. Based purely on the outcome of each confusion matrix, including the network metrics features produces a more performant model that can reduce the number of false positives while still capturing the same number of true positives when compared with the standard industry data model.

6.4.2 Shapley values for feature contribution

Applying the SHAP approach to the autoencoder trained on the industry data set including the additional network metric features, yields interesting results, as presented in Figure 6.14. A number of the network metrics features appear in the top 10 contributing feature list. Features $v1$ through $v47$ indicate the transactional features (depicted as a blue bar in Figure 6.14), whereas features $v48$ through $v61$ indicate the network metrics features (depicted as a red bar in Figure 6.14).

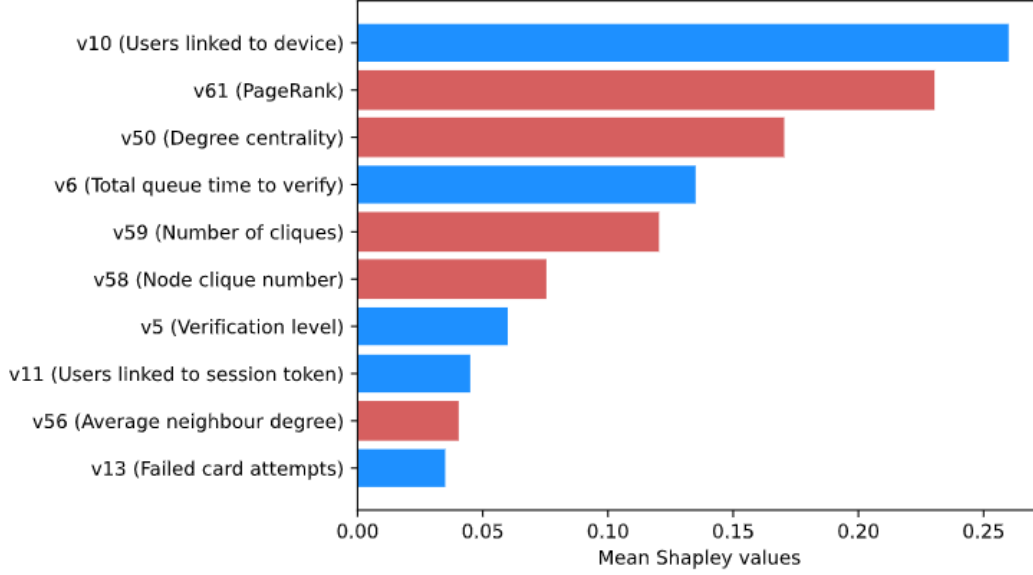


Figure 6.14: Shapley values for industry data with network metrics

The nature of each of the underlying autoencoder models is different as they are not trained on identical training sets, although these data sets are similar, and this similarity is expressed through some of the overlaps in the contributing features for each model. The most significant transactional features (indicated in blue in Figure 6.14) in the final model include the following:

Users linked to the device (v10): Feature $v10$ is the most significant of all and indicates whether other users are using the same registered mobile device. In the context of fraud detection, it is important to know whether users share a registered device because it is assumed that normal users only use their own devices and do not allow others to transact via their devices.

Total queue time to verify (v6): The second most significant transactional feature is $v6$, which is also ranked fourth among the top ten features. It shows whether a user

progressed through the sign-up process swiftly, which may indicate that the user has signed up with multiple fictitious accounts previously. The feature signifies that the user has gained a significant amount of experience in completing the sign-up process, which is not considered normal, where a normal user should only need to sign-up once. The feature may also detect an automated approach to completing the sign-up process, such as copying and pasting data from a template, which too is considered suspicious behaviour.

Verification level (v5): The third most significant transactional feature, verification level, indicates the level of transactional restrictions placed on an account. A higher verification level denotes a lower level of restriction, as more is known about the customer from a KYC (Know Your Customer) perspective. Both normal and fraudulent users would benefit from having a high verification level so that they can transact with relatively large financial amounts.

Users linked via a session token (v11): The fourth most significant transactional feature, users linked via a session token, measures the number of different users who transacted via a uniquely generated web token. As with users linked to the device, the assumption is that normal users would not transact via the same web token assigned to another user. Web tokens are generated by the organisation, whereas devices are registered using their physical MAC (media access control) address, of which the first is seen to be more accurate in detecting user devices.

Failed card attempts (v13): The fifth and final significant transactional feature, failed card attempts, indicates the number of times that a user attempted to load a card that was declined by the issuing bank. A high number of failed card attempts is suspicious behaviour, as it might suggest that a user is testing several stolen cards, an activity known as card testing.

The most important network metrics (indicated in red in Figure 6.14) that were identified by the SHAP analysis and which are proposed to augment the standard transactional variables are as follows:

PageRank (v61): The most significant network metrics feature is the PageRank metric, which determines the importance of nodes in a graph, based on the structure of the incoming edges. PageRank is a measure that captures the influence of a node within a graph, where influence is determined as the number of high-quality connections that a given node has to other nodes. High-quality connections are links to nodes that themselves are highly connected, which can be considered suspicious behaviour within a financial services ecosystem.

Degree centrality (v50): The second most important network metrics feature is degree centrality. The feature computes how many connections nodes have compared to all other nodes within a network. Therefore, nodes connected to more users than

other nodes are more likely to be anomalous. Conversely, it is assumed that normal users would not be strongly connected to other users in a financial ecosystem such as this.

Number of cliques (v59): Cliques or subgraphs are also strong contributing features, acting as another measure of how connected a node is to other large clusters within the graph. The number of cliques determines the number of possible subgraphs to which each node is connected, and is the third most important network metrics feature for detecting anomalies in the data. The higher the number of cliques, the more connected a user is within the network, which is deemed to be suspicious.

Node clique number (v58): The fourth most significant network metrics feature is the node clique number. A larger node clique number indicates that the user is strongly connected to other users and helps to surface anomalous behaviour within the network. The node clique number represents the largest clique containing each given node. The larger the node clique number per node, the more connected the user is within the network, and ultimately more likely to be acting as an anomalous user.

Average neighbour degree (v56): The fifth and final strongly contributing network metrics feature is the average neighbour degree which is simply the average of the degree (number of connections to other nodes) of the neighbours that surround each node. A node that is closely associated with other densely connected nodes would generate a high average neighbour degree, an indication of potentially suspicious behaviour.

The proposed approach to adding network metrics is validated by the Shapley value results, where the evidence presented can factor into decisions on whether to include network metric features in anomaly detection models. All the network metrics used in the study are related to the magnitude and nature of how users are connected, with these features acting as strong indicators of possible anomalous behaviour. The results in this chapter have proven that those network metrics that are related to the connectedness of nodes are important indicators of anomalous nodes and their behavioural characteristics. In the remainder of the section, formal accuracy measures normally used to evaluate model performance will be discussed.

6.5 Results and discussion

Evaluating the optimal architecture post the addition of the network metrics features shows an increase in optimal model performance when compared with the optimal model using only the standard industry data presented in Section 5.2 of Chapter 5. The optimal architecture determined through the NAS process achieved a balanced

score of 94.07%, which is an improvement on the 89.76% achieved in Table 6.7. The optimal model was constructed of nine hidden layers, compressing the data into a bottleneck of three neurons, and had 13 402 tunable parameters.

The optimal architecture for the data set without the network metrics in Chapter 5 and the data set including the network metrics in this current chapter both detected all the fraudulent instances in the test set, although the architecture in this chapter had 102 fewer false positives. The precision score was, therefore, higher for this model, which is favourable for an organisation looking to reduce the cost of misclassification.

The data used in this study were collected at a specific point in time, therefore, the data represents the state of the transactional records at that point. There is a possibility that fraudulent transactions reflected in the data had not yet been reported by the industry organisation or their associated payment providers at the time of the data extraction process. The delay between the fraudulent event taking place and the official confirmation of the validity of the event is a common challenge when conducting fraud detection in a real-world setting. Further details of this delay are discussed in Chapter 2. Ultimately, this means that there is a possibility that some of the false-positive classifications could indeed be true positives, as the official confirmation of fraud, as well as the associated labelling of the data, had not taken place at the time of the data extraction. The impact of these misclassifications is interesting, as it indicates that the anomaly detection approach appears to detect potentially anomalous transactional activity without being explicitly trained to do so using a label, as in a supervised learning context. Rather, the unsupervised or semi-supervised approach used in the proposed unified approach works well to detect still potentially unlabelled transactions.

The application of the proposed unified approach to anomaly detection in an industry setting provides a robust method for exploring several deep learning architectures, optimising their reconstruction error thresholds, and delivering an accurate probability score for each transaction case. It combines multiple aspects of the anomaly detection process into a logical pipeline that can be extended for future academic or industrial purposes.

Achieving an optimal model architecture through the proposed unified approach has several advantages. The random approach to the neural architecture search is a significant improvement on traditional grid search approaches, as it can find an optimal set of hyperparameters in a relatively short space of time. The random nature of the neural architecture search outperforms a grid search, especially in cases where the number of possible hyperparameters significantly influences the model's performance. A grid search often suffers from the curse of dimensionality where the number of possible parameters may grow exponentially as the number of hyperparam-

eters increases. These challenges in a traditional grid search, which are addressed with the proposed unified approach, are confirmed in the literature (Bergstra and Bengio, 2012), (Liashchynskyi and Liashchynskyi, 2019).

When analysing the industry data used, the addition of the network metrics added fourteen features to the data set. The addition of the new metrics has a knock-on effect on the design of the architectures, as each architecture needs to cater for the size of the input space when developing the input layer. The NAS process in Sections 5.3 and 6.3 made use of each data set input space as the input layer dimension during the NAS process, where the first simulated hidden layer in each process was constrained by a capped value, namely the input space of the smaller, pre-network metrics data set. Another observation regarding the NAS process is understanding the impact of the size of the decrease/increase in node count from each subsequent layer, as well as how performance is affected when combining certain node count progressions with architectures of varying depths.

As discussed in Section 6.4.2, introducing the network metrics into the anomaly detection system proved to be beneficial, with the study proving that the network metrics that are related to the connectedness of nodes are important indicators of anomalous nodes and their associated behavioural characteristics.

6.5.1 Practical considerations

In this section, a discussion on the practical considerations to be taken into account when implementing the models generated in Chapter 5 and this chapter is provided. These practical considerations are important when deciding on which model to implement in a production setting. In this discussion, model 1 will refer to the model generated in Chapter 5, which made use of the data set without the network metrics features. Model 2 will then refer to the model generated in this current chapter, which was trained using the data set that included the network metrics features.

Ease of constructing models

Both models, as well as their respective NAS processes, were generated using the TensorFlow framework in Python. The model including the network metrics (model 2) was comparatively far more complicated to set up, considering the introduction of the network metrics features. The set-up includes creating the model features via feature engineering. The steps involved in this process were discussed in Section 6.2. Considering the significant feature engineering required to construct these network metric features, this would be a significant deciding factor on whether to implement the second model in an operational environment.

Model run-times

The model training process was relatively similar for both model 1 and model 2, with model 2 taking slightly longer to train, considering its larger feature space. Due to the very small difference in training times between model 1 and model 2, training time would therefore not be considered a significant deciding factor when choosing between using either model in an operational setting.

Model complexity

Complexity relates to the number of hyperparameters of the models, including node count, number of hidden layers, and the total number of tunable parameters. These are dependent on the outcome of the NAS process, which determines the optimal architecture, based on a balanced performance score for the sake of considering the complexity. Model 1 has fifteen hidden layers while model 2 has nine hidden layers. Model 2 also has fewer overall trainable parameters and it can therefore be concluded that this model is less complex when compared with model 1. The impact of reduced complexity of the model architecture did not impact the ability to make efficient predictions with the model. Model prediction time was essentially the same for both models, and therefore it can be concluded that model complexity is not a significant factor in this context.

Knowledge required for implementation

There are different knowledge requirements for implementing both model 1 and model 2. While model 1 requires an understanding of how to construct and train autoencoders, as well as how to interpret the model results, model 2 requires an additional understanding of network analysis and how to use it to calculate network metrics. If the expertise is available to perform network analysis, it is better to experiment with network metrics as additional variables for the autoencoder, as this delivers an improved classification performance.

6.5.2 Recommendations

Considering the results presented in this section, the recommendation would be to include network metrics features in practice. The inclusion of network metrics is dependent on the ability to calculate the network metrics in a reasonable period and with reasonable computing resources. Including network metrics in the features reduced the number of false positives when compared with the initial industry data model, which would lead to a reduction in operational costs when responding to reported instances of fraud.

Comparing the Shapley values generated for model 1 and model 2 showed that some network metrics features contribute significantly to detecting anomalies. In particular, the PageRank and degree centrality network metrics had the greatest impact, mainly due to their ability to determine the importance of having many links within a network.

The application of the unified approach proved to be useful in detecting anomalies using real-world transactional data. The extension of the unified approach included the introduction of network metrics features, which is a novel approach for creating features in an anomaly detection context. There are also benefits to using SHAP in an anomaly detection system, including the ability to determine the strongest contributing or offsetting features for any anomaly detected.

The introduction of network metrics into the anomaly detection system also reduces the manual overhead required to investigate the relationships and interactions between users, an approach traditionally conducted by analysts. The raw network metrics used to develop the features for the anomaly detection system can be used in isolation to understand the connections between users, but without having to determine these connections manually.

If the expertise and resources are available to perform network analysis, it is better to experiment with network metrics as additional variables for the autoencoder as this delivers higher performance and brings an additional dimension to the transactional data mostly used in anomaly detection problems. Alternatively, the network features could be determined by computing approximating values, using graph neural networks.

In summary, when an anomaly detection model for a transactional setting is considered, the following is recommended based on the work presented in this chapter:

- Implement an autoencoder neural network model to identify anomalous transactions.
- Make use of the proposed unified approach to ensure that a “best” architecture for the autoencoder is selected, that an optimal error threshold is determined, and that a proper scaling of scores is implemented.
- Include a sufficient number of social network analysis metrics in the data set. Ensure that at least those network metrics related to the connectedness of nodes (users) are included to address the relationships and interactions between users. Examples include degree centrality, node clique number, number of cliques, average neighbour degree, and PageRank.

- Other non-network variables that proved to be important in financial transactions include verification level, total queue time to verify, users linked to a device, as well as the number of failed card attempts.
- Carefully select model performance metrics to ensure that the costs of misclassification are considered.
- Finally, take cognisance of the list of practical considerations listed in Section 6.5.1.

The list of recommendations together with the practical considerations concludes the results and discussion section. The focus in subsequent chapters will now shift to additional techniques that can be implemented.

6.6 Chapter conclusion

Chapter 6 was the second of three chapters in Part III of the study. The purpose of this chapter was to extend the proposed unified approach proposed in Chapter 5, by incorporating several network metrics features in the modelling process. The network metrics features were calculated using a secondary data set sourced from the same industrial organisation, where the data represented interactions between users within the same transactional system.

The proposed unified approach was first applied to an industry data set in Chapter 5. In this chapter, the approach was applied again to the industry data set, following the augmentation of the data set with network metrics features. A performance analysis comparison was then conducted to determine whether adding these features would have a significant impact on the performance of an autoencoder used to detect anomalies in a transactional setting.

From a performance perspective, a more desirable outcome was achieved after introducing the network metrics features, which effectively helped to reduce the number of false positives generated when using only the standard industry features. The network metrics features were considerably more complex to construct; however, they impact the ability to generate these features in a timeous manner in a real-world setting.

In the next chapter, which is the last chapter in Part III of the study, a discussion on self-organising maps, an unsupervised learning approach will be presented. It will be illustrated how self-organising maps can be useful in interpreting the structure of the data from an exploratory perspective. The intention is to illustrate how self-organising maps can be used to complement the anomaly detection approach presented in Chapter 6.

Chapter 7

Self-organising maps for anomaly detection

7.1 Chapter introduction

Chapter 7 is the final chapter of Part III which details the models and proposed techniques. The methods employed in Chapters 5 and 6, the other two chapters of Part III, successfully detect anomalies in transactional data using certain SNA metrics as additional features. The objective of this chapter is to further enhance and expand the analysis capabilities of the models already presented. To do this, an unsupervised learning approach called self-organising maps (SOM) is presented. The intention is to illustrate how an SOM analysis can complement the anomaly detection approach presented in the preceding chapters. The chosen SOM approach has useful interpretation features and will assist in describing the nature of anomalies, as well as highlight which features are most significant in the detection of anomalies.

Including the SOM in this study complements the classification of anomalies in Chapters 5 and 6, by providing decision makers with additional information, aiding them in understanding anomaly and feature behaviour over time. Therefore, the SOM will be used to augment the anomaly detection approach, but will not be involved in the actual classification of anomalies. To start with, a brief discussion on SOM model building is given in Section 7.2, followed by model training and model evaluation in Section 7.3 and Section 7.4 respectively. The use of a u-matrix as an interpretation tool is then discussed, as well as interpreting features via component planes in Section 7.5. The chapter findings is summarised in Section 7.6, whereafter the chapter is concluded in Section 7.7.

7.2 Self-organising map model building

In Chapter 7, the aim is to apply the various self-organising map steps described in Chapter 3, Section 3.2. This involves the building, training, evaluating, and interpretation of self-organising maps. The combined feature set developed in Chapter 6 is used to construct the self-organising map.

The transactional features computed in Chapter 5 were made up of 47 distinct features relating to the transactional behaviour of users within a financial system. There were 120 676 transactions in this feature set, each representing the aggregated behaviour of a unique user. An additional 14 features were then generated, where various social network analysis algorithms were applied to a set of network data representing the connections between the same 120 676 users within the system. The combined feature set, representing both the transactional and network features, therefore has 61 features and 120 676 transactions.

As discussed in Chapter 3.2, Section 3.2.1, self-organising maps produce a two-dimensional representation of higher-dimensional space. The two-dimensional space takes the form of a lattice, containing several predefined properties. To build an effective self-organising map, one needs to consider several critical lattice design choices. One of those design choices is whether to use a rectangular or hexagonal lattice structure. While the hexagonal structure has shown to be superior in terms of its clustering ability (Jiang and Harrie, 2004), the traditional rectangular structure is a less complicated design that can be used as an effective baseline solution for developing an unsupervised learning model that is meant to operate in a business context. The rectangular lattice structure is used in this study due to its suitability for creating a functioning baseline model.

Another design consideration is the dimension of the lattice. It has been shown that using $5\sqrt{n}$ nodes, where n is the number of samples or features in the high-dimensional space, is a good choice for the size of the $n \times n$ lattice (see Chapter 3, Section 3.2.2). Therefore, considering that the high-dimension space contains 61 features, $5\sqrt{n} = 39$, which rounded up to the nearest square number of 49, yields a 7×7 lattice.

7.3 Self-organising map model training

The model training process assumes the same training data that was used to train the autoencoder model in Chapter 6. This feature set contains 61 features, and 120 676 transactions, and is a combined feature set representing both the transactional and network features generated in this study. Normalisation has already taken place, therefore, no further pre-processing of the data is required. This normalisation en-

tailed a standardisation process where each vector is centred around the mean, with a unit standard deviation. The model is trained using *MiniSom*, a minimalistic and Python-based implementation of self-organising maps (Vettigli, 2018).

Before the commencement of the training process, the target variable of the training data set is set aside, as the SOM is trained in an unsupervised manner. The target variables (or labels) will come into play at a later stage when they will be mapped onto their respective nodes in the model's output lattice space. *MiniSom* requires that several parameters are set at training time. Firstly, the neighbourhood function for the training process needs to be specified. This function weights the neighbourhood of a position on the map. Possible values for this parameter are the Gaussian, Mexican hat, bubble, or triangle functions. In Chapter 3, Section 3.2.3, it was explained that the choice of the neighbourhood function parameter is largely irrelevant, and therefore the default Gaussian function is used. Secondly, Sigma, or the spread of the neighbourhood function, is also required. A sigma value of 1 is chosen, as the *MiniSom* tool recommends this as a proven, good default value.

The SOM model was trained for 200 epochs in order to evaluate the model performance over an adequate number of epochs. Figure 7.1 shows the results of the training process. Both the quantisation error E_q and topographic error E_t have decreased somewhat over the first 100 epochs; however, beyond that, E_t appears to start increasing again. E_q and E_t are the standard metrics for evaluating the quality of a SOM (see Chapter 3, Section 3.2.4).

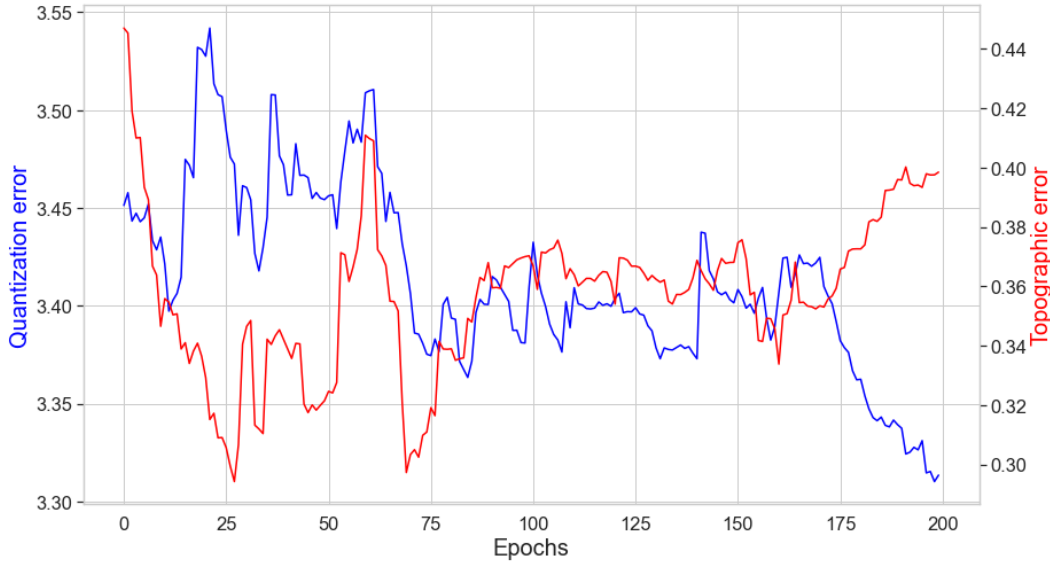


Figure 7.1: Training the SOM

E_q is the average distance of the input data points to the nearest node in the lattice, while E_t refers to the proportion of data points where the best matching unit (the node that is closest to the data point in the input space) and the second-best matching unit are neighbours within the lower dimensional lattice space. There is a balance between optimising for an improved E_q at the expense of E_t , and *vice versa*. This trade-off can be managed by iterating through various hyperparameter configurations.

Depending on the objectives of the problem, a balance between the two errors can be found by finding the optimal choice of hyperparameter combinations, a process known as hyperparameter optimisation. This process was not necessary for this study, as a suitably performant model was obtained through trial and error selection of a few hyperparameter configurations.

The phenomena whereby E_t increases, while a decrease in E_q is witnessed, can be experienced in cases where the dimension of the lattice is increased (Polzlbauer and Rauber, 2004). In Section 7.2, the optimal lattice structure was chosen following research on the best choice for lattice dimension selection, therefore the increasing E_t beyond epoch 150 can be disregarded. The final model was trained to 75 epochs, where both errors were relatively low and overfitting was not yet evident. This included training the model with the learning rate and sigma value set to 0.5, where sigma accounts for the spread of the Gaussian neighbourhood function. The SOM that was generated is of good quality according to the E_q and E_t metrics, therefore the model can be used with confidence moving forward.

7.4 Self-organising map model evaluation

There is a variety of techniques that can be used to analyse or evaluate the results generated by training an SOM. In this section, some of the more popular techniques will be presented, namely the node activation frequency, class distribution, class proportion, u-matrix, and clustering map visualisations.

7.4.1 Node activation frequencies

After the final model had been trained, the data was projected back onto the lattice space. Figure 7.2 showcases the node activation frequencies on the 7 x 7 SOM lattice.

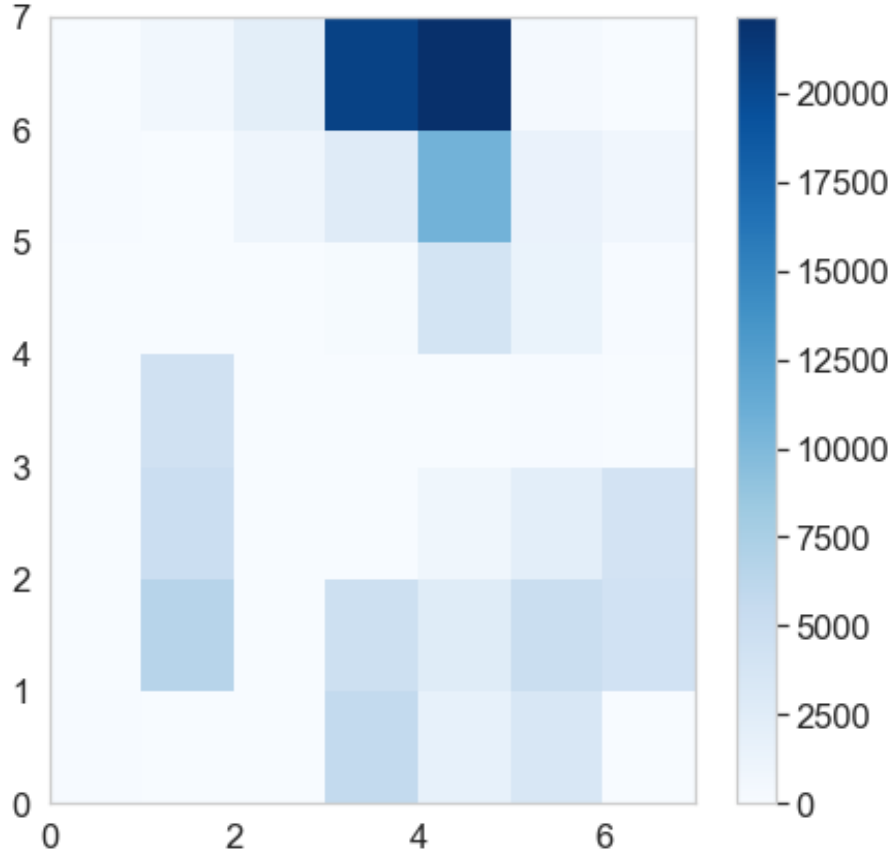


Figure 7.2: The node activation frequencies of the SOM

Node activation frequency is merely how often a node in the lattice is assigned a data point from the input space. The underlying classes of the data points are not considered in this visualisation, although the classes are depicted later in this section.

The legend indicates that darker shades of blue mean that an increasing number of data points are mapped to those clusters, while lighter nodes contain fewer data points. The most densely populated node is at coordinate (5,7) on the lattice. Considering the large class imbalance between the normal and anomalous records, the more densely populated nodes would naturally contain more of the majority class, although that is not necessarily evident in this visualisation.

7.4.2 Class distribution

Figure 7.3 provides an alternative view of the lattice, with the distribution of each of the classes being represented by different colours. The red points indicate anomalies, while the green points are the normal records. Shifting the focus to coordinate position (5,7), which was the darkest blue node in Figure 7.2, it is again clear that this node is densely populated. Interestingly, the node contains no anomalies, which in this case confirms that the SOM was able to effectively separate the majority of the normal data from the anomalies. Rather, the anomalies are mapped to five other nodes, with the majority mapped to coordinate position (6,7). The node at position (6,7) contains anomalies only, and therefore the model performs favourably in detecting anomalies in this node. Four other nodes contain a mix of both normal and anomalous records, and if the model was used for classification purposes, then these locations would be where the misclassifications occur.

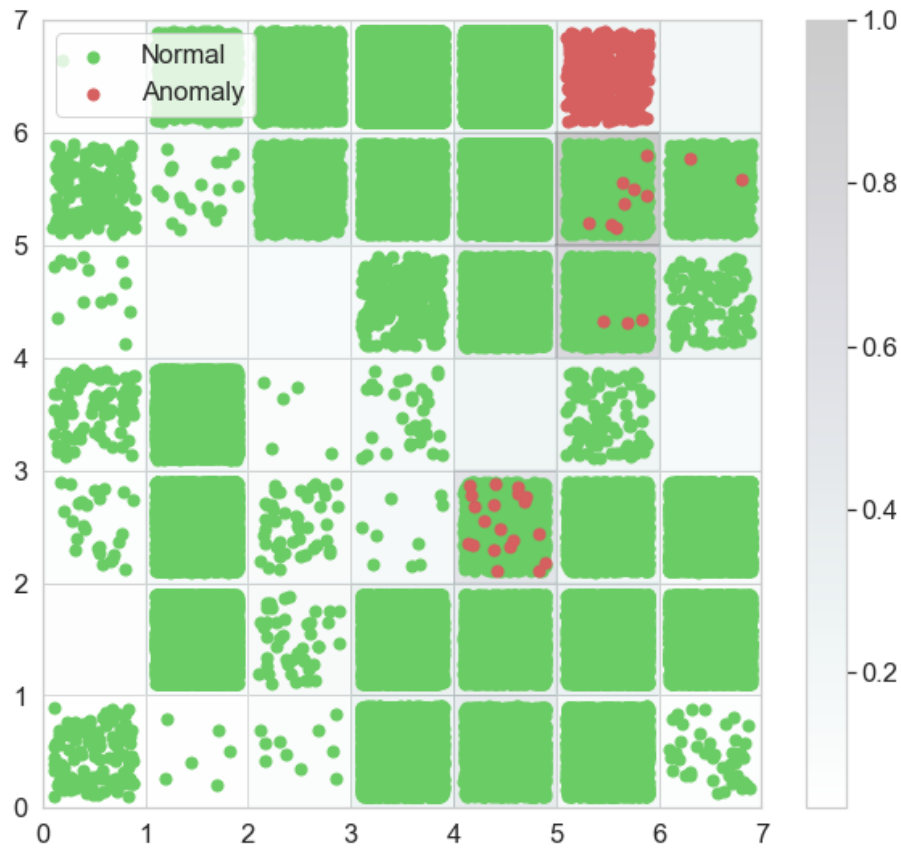


Figure 7.3: The distribution of the classes across the SOM

The other node positions that contain anomalies are (5,3), (6,6), (6,5) and (7,6). All four contain far fewer anomalies than the exclusively anomalous node at position (6,7). By the anomalous records being assigned to different nodes, it is clear that there are different patterns of anomalous behaviour at play, although these are not intuitively understood by merely looking at Figure 7.3. Rather, the results would need to be analysed from a different perspective to achieve an understanding of the underlying behavioural differences between the anomalous nodes.

7.4.3 Class proportion

Another perspective from which to view the class data on the lattice would be via a visualisation depicting the class proportions. The proportion of the classes across the SOM lattice is presented in Figure 7.4. Except for the red circle indicating the majority anomalous node, the balance of the node visualisations does not provide much insight. This is largely due to the class imbalance problem. Even the four nodes that contain a mix of normal and anomalous data are not easily visible, regardless of whether they contain anomalies. Therefore, proportion visualisations might not be the best approach in cases where a large class imbalance persists.

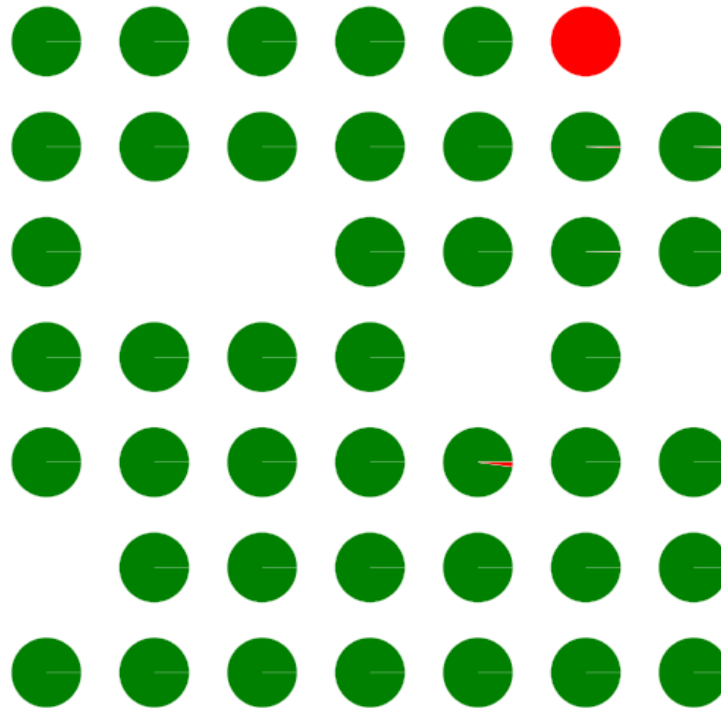


Figure 7.4: The proportion of the classes across the SOM

The visual analysis of the various lattice presentations confirms the results of the anomaly detection models presented in Chapters 5 and 6 and is a useful addition to the overall framework that is proposed by the study.

7.4.4 U-matrix

In the previous sections, the power of displaying the count and class distributions across the lattice nodes was illustrated. In this section, the u-matrix visualisation is applied. The u-matrix is a map produced by mapping the average Euclidean distance between each node and its nearest neighbours in a high-dimensional space onto a two-dimensional lattice. Figure 7.5 shows the application of a u-matrix visualisation to the trained model. The legend indicates the range of colours for the nodes, where a darker shade indicates nodes that are further away from their immediate neighbours in high-dimensional space. Conversely, nodes with a lighter colour are closer or similar to their neighbours in this space.

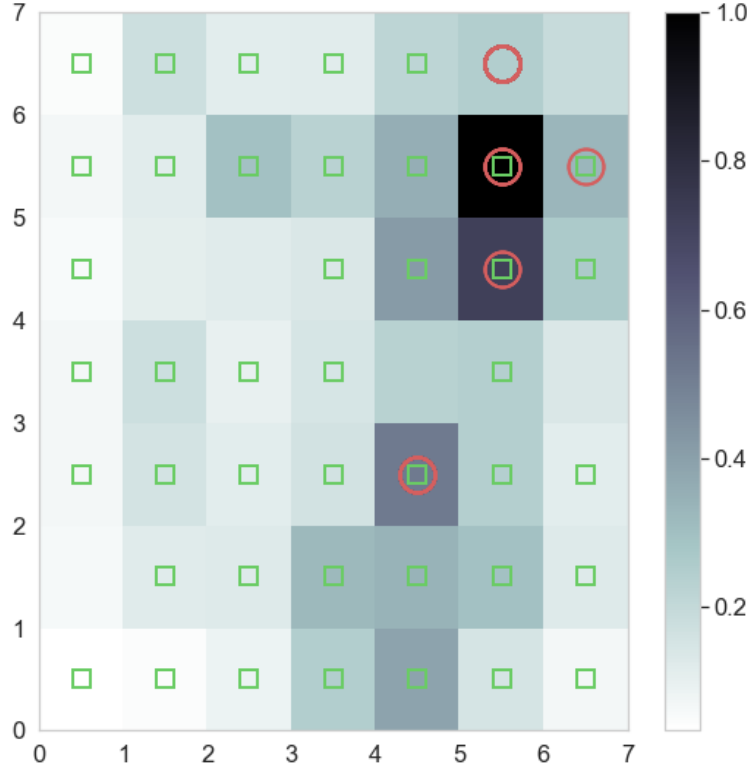


Figure 7.5: The u-matrix

The normal (green squares) and anomalous (red circles) classes have again been mapped to this visualisation. It is noticeable that the node at position (6,6) is the darkest shaded node on the lattice. Therefore, this node is furthest from its neighbours, which could be considered anomalous. Interestingly, this node contains a mix of both classes, and therefore this node detects something significantly different in these records.

The node containing anomalies only, at position (6,7), is surprisingly not darkly shaded and therefore it is not seen as too dissimilar from its neighbours. This could, however, be a direct result of the influence of one of its neighbours, the darkest shaded node at position (6,6). Two of the other three nodes containing anomalies, node positions (6,5) and (5,3), are the second and third darkest nodes behind position (6,7), thereby confirming that these nodes detect behaviour that differs from their immediate neighbours.

As was presented above, various visualisation approaches were applied to interpret the output of the final SOM model. The node activation frequencies indicated that the most active nodes were those where normal transactions were mainly detected, which is influenced by the class imbalance in the data. The class distribution visualisation provides additional insights into the anomaly detection problem by showing that the anomalies are not all clustering together in a single node. Rather, the anomalies are spread across five different nodes, indicating that there are likely to be different types of anomalies in the data. This level of detail was not present in the findings from Chapters 5 and 6, and therefore adds perspective to how anomalies can be detected.

The class proportions visualisation was not particularly insightful. This was mainly due to the impact of the class imbalance on the data, which made displaying the class split where a mix of anomalous and normal data occurred, difficult to achieve. Lastly, the u-matrix validated the findings in the class distribution visualisation, by depicting the distance between nodes in the lattice. This is useful for anomaly detection, as it indicates just how far away, or anomalous, a node is in relation to other normal or anomalous nodes.

7.4.5 Clustering maps

The strongest contributing features for each node in the lattice are depicted in the clustering map shown in Figure 7.6. A typical clustering map colours each node according to the feature. Considering that this study intends to detect anomalies, the largest nodes containing only normal transactions, as well as those five nodes that contain anomalies, are highlighted instead. In Figure 7.6, the nodes containing anomalies are coloured in red, while the dark green nodes are the largest five that

contain only normal transactions. The rest of the nodes are then given a pale green colour.

The clustering map indicates that $v3$ (whether or not the user has the lowest level of verification), $v10$ (whether other users in the system are using the same registered mobile device), $v15$ (the number of credit cards declined by the authorising bank), $v61$ (PageRank), and $v50$ (degree centrality) are the strongest features for the five nodes containing anomalous data. These results support the findings in Chapter 6 that showed that the number of users linked to each other via the same device, PageRank, and degree centrality were the most significant features for detecting anomalies.

Regarding the clusters with the largest number of normal transactions, $v4$ (whether the user had the middle level of verification), $v7$ (whether or not the user made use of a municipal verification report as proof of identification), $v31$ (whether there are differences in the IP country of the user's identity document and the country from which he/she originally signed up) and $v38$ (whether there are IP country differences between the user's passport country and his/her bank account country) are the strongest features reported.

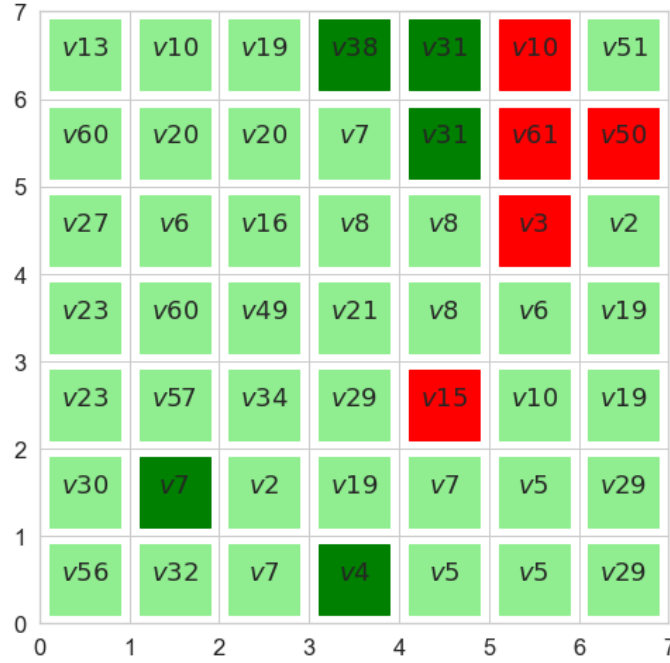


Figure 7.6: The clustering map of the most significant features per node

Clustering maps are useful in anomaly and fraud detection environments, as they provide an easy way of identifying the impact of specific variables in fraud detection exercises. Having knowledge of the variable impact is useful from an operational perspective, as it allows organisations to implement corrective processes to counter the behaviours detected by the contributing variables. These maps also help to create an audit trail of the changes in significant variables over time that in turn may assist with uncovering emergent fraudulent trends seen in the data.

7.5 Interpreting features via component planes

Another important trait of self-organising maps is the ability to visually inspect their component planes. Component planes are a visualisation technique that separately maps each feature of the node's weights onto a two-dimensional lattice. These are useful for understanding which features are highly correlated, as correlated features will have similar component planes. This can assist in identifying feature redundancies, an approach whereby some highly correlated features are removed from the training process. The component planes for all 61 features are displayed in Figures 7.7, 7.8 and 7.9. While some features appear to be correlated, such as *v27* and *v28* in Figure 7.8 where the node activations are very similar, the redundancies were not removed, as the focus was to illustrate the contribution of all features rather than to optimise the performance of the SOM.

Component planes can also be used for understanding feature contributions, by comparing the component plane for each feature to the node activation frequency and class distribution plots depicted in Figure 7.2 and Figure 7.3, respectively. In cases where a particular class dominates a node, any component plane that indicates a high weighting for that same node means that the underlying feature contributed significantly to the clustering decision. A red node indicates a higher weighting for that feature's contribution to the node clustering decision, while a blue node indicates a lower weighting and therefore a smaller contribution to the decision. Understanding these feature contributions is important for this study, as it allows practitioners to relate the output of the SOM to the feature contributions in Chapters 5 and 6, ultimately validating the contributions from both the autoencoder and SOM modelling approaches.

Shifting the focus to the primarily anomalous node, at the coordinate position (6,7) in Figure 7.3, it becomes evident that some features in the component planes (Figures 7.7 to 7.9) contribute more than others to its clustering decision. Features *v5* (whether the user has the highest level of membership on the platform and can access all transactional functionality without limits), *v6* (the total time that the user spent in the queue to get his/her identity verified), *v8* (whether the user used an ID

card as proof of identification) and *v10* (whether other users in the system are using the same registered mobile device) appear to be the most significant contributors to this node.

These results align with the key findings from applying SHAP to the autoencoder anomalies. More specifically, features *v5*, *v6*, and *v8* were three of the top four most significant features found after applying SHAP to the autoencoder trained using the transactional features only (see Chapter 5, Section 5.6.2), whereas features *v5*, *v6*, and *v10* were the most significant transactional features determined after applying SHAP to the autoencoder model trained using the combined feature set (see Chapter 6, Section 6.4.2). Although it is expected that the results obtained from an autoencoder and SOM would not be the same, it is reassuring that there is a significant overlap between the anomalous indicators determined by these two approaches.

The second node containing anomalies, at position (6,6) in Figure 7.3, appears to have significant contributions from a number of the network features, which range from *v48* through to *v61*. These findings indicate that network metrics features are significant contributors to detecting anomalies in a transactional system, therefore validating the extension of the unified approach through the inclusion of the network metrics. This correlates with the findings after applying SHAP to the autoencoder trained on the combined feature set (see Chapter 6, Section 6.4.2), where PageRank, degree centrality, and the number of cliques were found to be the most significant network metrics features.

The third node containing anomalies, at position (5,3) in Figure 7.3, has contributions from a number of the transactional features, namely features *v15* through *v19*. To reiterate, these are *v15* (the number of credit cards declined by the underlying bank), *v16* (the maximum number of attempts at loading a single credit card within an hour), *v17* (the maximum number of attempts at loading any number of credit cards within an hour), *v18* (the length of the language code used, indicating that the user may have used multiple language settings), and *v19* (whether the user was successfully referred to the platform by another user). The results are quite different from the features determined by the primary anomalous node at position (6,7), and likely point to a different type of anomaly. There is no expectation that only a single type of anomalous behaviour exists in a system, rather there may well be numerous typologies, each with its nuances as picked up through this feature contribution analysis.

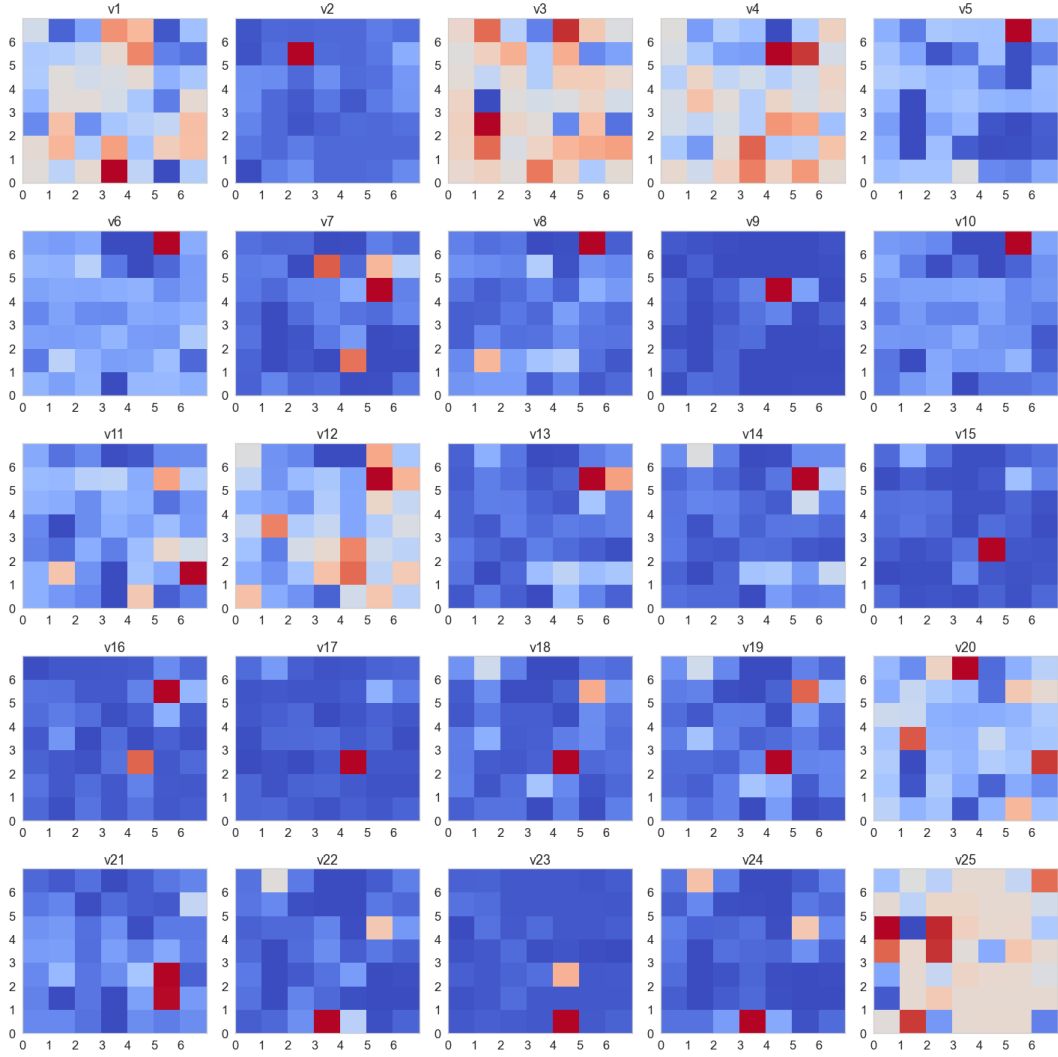


Figure 7.7: Component planes for features 1 to 25 (transactional features)

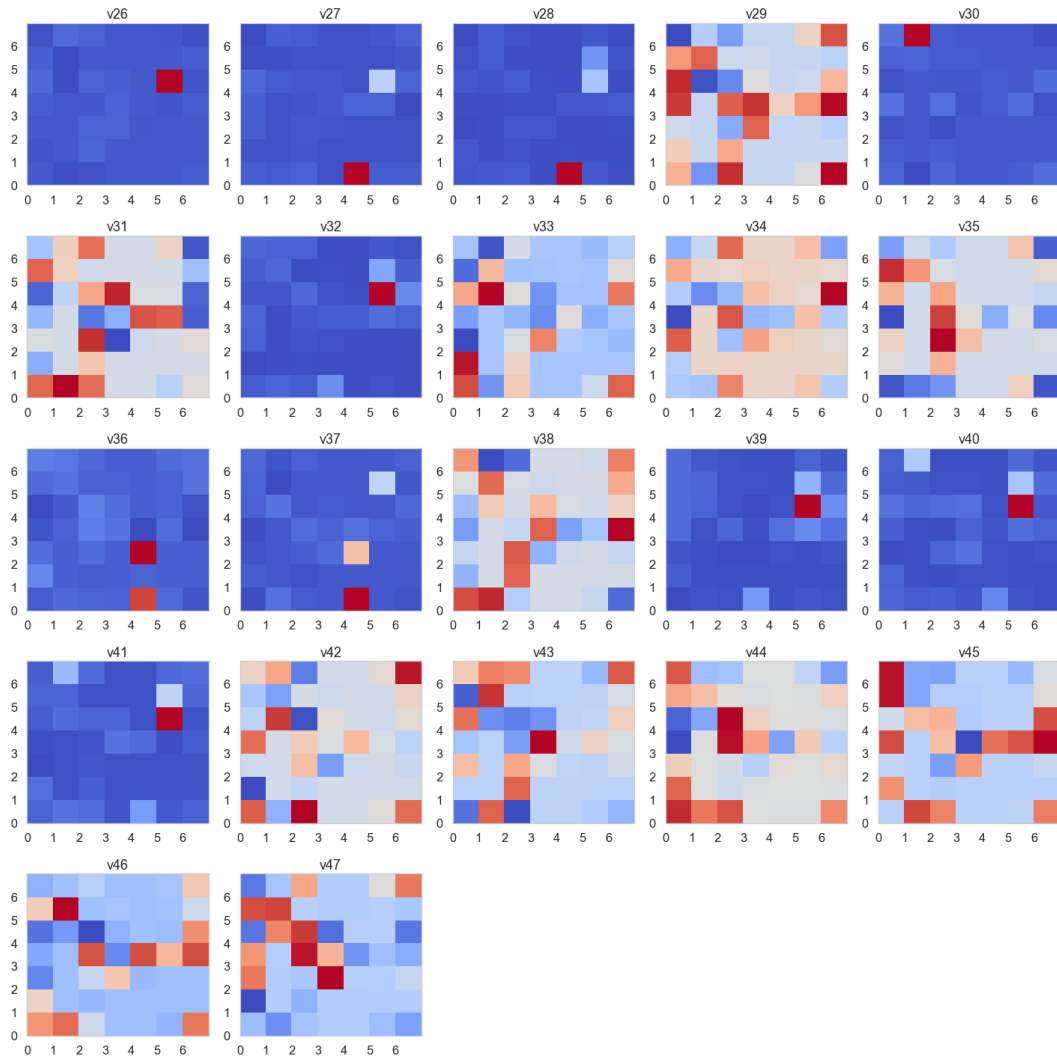


Figure 7.8: Component planes for features 26 to 47 (transactional features)

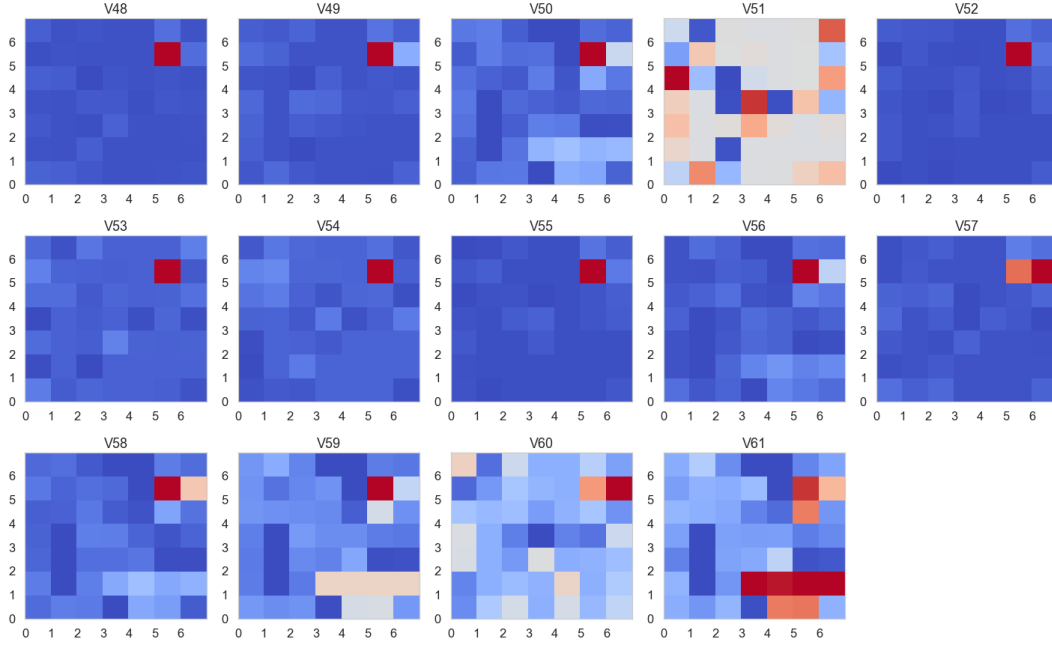


Figure 7.9: Component planes for features 48 to 61 (network metrics features)

Visual analysis of feature contributions using component planes is a powerful approach for understanding the clustering mechanism behind a self-organising map. Fraud analysts may use this technique to confirm the importance and relevancy of features used in anomaly detection models. Furthermore, component planes enable the study of variable behaviour over time. Such trend analyses may assist in the identification of other fraud contributing factors and will provide additional insight into anomaly detection modelling activities.

A disadvantage may be that visual inspection may not be beneficial in scenarios where fast analyses are required or a high number of potential anomalies exist. However, the technique also provides for node weightings that can be extracted programmatically, meaning that the top n features for anomaly detection could be retrieved and used for fast reporting purposes. This is an effective approach for fast analysis of feature contributions considering that an anomaly detection system is likely to detect hundreds or thousands of potential anomalies over time.

The approach of collecting the feature contributions from the node weightings will be used as part of the anomaly detection framework presented in Chapter 8. Table 7.1 displays an example of the node weightings for a single feature, $v5$.

Table 7.1: An example of the node weightings for feature $v5$

-0.164	-0.643	0.120	0.066	0.057	2.474	-0.011
0.074	-0.266	-0.962	-0.667	0.042	-1.001	-0.626
0.206	0.111	-0.039	-0.016	-0.637	-1.003	-0.022
-0.065	-1.022	-0.020	0.077	-0.090	-0.153	-0.040
-0.347	-1.034	-0.301	-0.074	-0.964	-1.030	-0.895
-0.175	-1.034	0.019	-0.900	-1.017	-1.006	-0.904
-0.009	0.080	0.076	0.704	-0.533	-0.590	-0.244

Table 7.1 is a 7 x 7 matrix representing the importance of $v5$ from the perspective of each node in the lattice. The value of 2.474 in the sixth position within the first row corresponds to the strongest weighting for $v5$, which is confirmed by the red node within the same position in Figure 7.7.

7.6 Results and discussion

The results obtained from training a self-organising map model on the combined transactional and network metrics features set have proven to be useful in several ways. The node activation frequencies derived from the trained model’s output allowed for an understanding of the distribution of all of the records across the lattice. Figure 7.2 showed that the vast majority of the data has been assigned to two nodes, depicted in bold blue in the image.

To gain a deeper understanding of how these distributions are split amongst the two classes, Figure 7.3 overlaid both the anomalous and normal classes onto the lattice. While some misclassifications occurred, the majority of the anomalies were grouped in a single node, at position (6,7) on the lattice. This node contains anomalies only, and thus the behaviours attributed to this node were considered the main type or typology of anomalous behaviour. The four other nodes that contain anomalies also contain normal data, and therefore these are the nodes where the misclassifications occur. Each of these four nodes could also be considered to exhibit its own unique anomalous behaviour typology, as there is no expectation that only a single type of anomalous behaviour should occur.

Figure 7.6 represents the lattice as a u-matrix, with the classes overlaid. The u-matrix shows how far each node is away from its neighbouring nodes in the high-dimensional space. Node (6,6) is the most prominent in this figure and is therefore very far from its neighbours, where the large degree of separation could be indicating that the node is anomalous from its neighbours. Interestingly, the node contains a mix of normal and anomalous data, and therefore this node may require further manual investigation. It might be the case that the normal data in that node are

exhibiting some abnormal behaviour that was not yet detected at the time of data collection, and which therefore would not reflect in the class labels. As a reminder, there is usually a delay in information feedback between the occurrence of an anomaly and the subsequent reporting by the financial authority involved, as was discussed in Chapter 2, Section 2.3.3.

Interpreting the feature contributions via component planes presented informative results. By making direct cross-comparisons between the anomalous nodes detected in Figure 7.3 and their underlying positions for each feature within Figures 7.7 through 7.9, it is visually possible to attain the features that contributed the most to each of the anomalous nodes. The exclusively anomalous node at position (6,7) was influenced by some of the same features found to influence the autoencoders after applying SHAP in Chapters 5 and 6. The most significant features were *v5* (whether the user has the highest level of membership on the platform and can access all transactional functionality without limits), *v6* (the total time that the user spent in the queue to get his/her identity verified), *v8* (whether the user used an ID card as proof of identification) and *v10* (whether other users in the system are using the same registered mobile device). These explain the behaviours behind the main anomalous typology discovered in the data.

The node at position (6,6), which contains a mix of normal and anomalous classifications, is influenced by a number of the network metrics features calculated in Chapter 6. This finding is meaningful, as it indicates that these features are important contributors to detecting anomalous data. While the visual inspection of the component planes works well in practice, the node weightings and hence the top contributing features for each node will be determined programmatically in the anomaly detection framework introduced in the next chapter.

Detecting anomalies in data requires the identification of the features relevant to the problem domain. In this study, anomaly detection models were trained to detect anomalous records within transactional features, following which social network analysis metrics were used to create network features that extended the standard transaction-level approach. Self-organising maps were then introduced into the study, to provide further confirmation of the effectiveness of the proposed framework. There was a variety of benefits to implementing a self-organising map in this study, some of which include the ability to describe the nature of the anomalies detected, highlight the significant features, and understand how features are clustered. Additionally, these benefits can be employed to study the changing anomalous behaviours over some time, as there is no assumption that these behaviours would remain static.

The results from Chapter 7 show that the benefits of self-organising maps described above helped to achieve an understanding of feature significance and their

associated behaviours. This was witnessed through the assessment of the component planes, which provided empirical evidence of feature contribution that aligned with the findings from Chapters 5 and 6, where SHAP was applied to the autoencoder models to determine the contributing transactional and network metrics features. Therefore, component planes expand upon the methodologies proposed in Chapters 5 and 6, thereby confirming self-organising maps as a necessary contribution in the context of the study.

7.7 Chapter conclusion

Chapter 7 is the final chapter in Part III of the study. The purpose of this chapter was to elaborate on the analysis capabilities of the previously presented models, through the introduction of self-organising maps. A SOM was used to complement the anomaly detection approach through its various functional benefits, namely feature interpretation, analysis of anomalies, and understanding of features significant to anomaly detection. The training of a SOM was also presented, followed by a process of model interpretation through the application of a variety of visualisation techniques. Examples included visualisations related to node activation frequencies, class distribution, and class proportion, as well as component planes and the u-matrix.

In the next chapter, which is the first chapter of Part IV of the study, there will be a discussion on the proposed anomaly detection framework. The combination of the various approaches developed in Chapters 5, 6 and 7 into a consolidated anomaly detection framework for detecting anomalies in a transactional setting will be addressed in detail. The intention is to illustrate a computer-based representation of the anomaly detection framework to the reader.

Part IV

Final framework and conclusion

Chapter 8

An anomaly detection framework

8.1 Chapter introduction

Chapter 8 is the first chapter of Part IV of the study. Part IV presents the proposed machine learning-based anomaly detection framework as well as the conclusion of the study. The ultimate goal of the study is to propose a new anomaly detection framework. In Chapters 5, 6 and 7, new approaches were proposed and discussed through appropriate literature reviews, and their application was validated through the use of real-world data. These approaches are now combined into a novel framework for anomaly detection. The purpose of this chapter is to provide a detailed description of the final proposed framework.

A high-level overview of the anomaly detection framework is set out in Section 8.2. More specifically, the framework includes five main components and a database, and the relationships and process flow between each of these components are discussed at a high level in this section. In Section 8.3, the framework's social network analysis (SNA) component is discussed in more detail. This component assists with the creation of network metrics features. The second component of the anomaly detection framework, the anomaly detection (AD) component, is presented in Section 8.4. Here, the unified approach is executed, and anomalous records are classified. This is then followed by Section 8.5, in which there is a discussion of the third component of the framework, namely the SOM component. In this section, the framework's second machine learning algorithm, a self-organising map, is trained to provide an additional approach to the anomaly detection problem. Two human-facing components, the visual analytics (VA) and what-if (WI) components are presented in Section 8.6 and

Section 8.7, respectively. Finally, the visualisation of the output results is summarised in Section 8.8.

8.2 The proposed framework

The proposed anomaly detection framework detects anomalies from raw transactional and network data, by training an autoencoder using the features derived from both data sources. The framework consists of five components, two sources of input data, and a database, all of which are displayed in Figure 8.1. The first component of the framework is the SNA component, which takes raw network data as input. This component follows several steps, where the output of the final step is used as input in the AD component. The resulting network metrics features are also stored in the database, to be used in conjunction with the transactional features for training the autoencoder and self-organising map models.

Secondly, the AD component of the framework is introduced, which contains the basis of the unified approach. The unified approach processes are displayed by the red rectangles in Figure 8.1. Raw transactional data is taken as input and combined with the network metrics features derived from the SNA component. These combined features are scaled and fed through the unified approach, with the resulting predictions being saved into the database. Various steps are followed in the unified approach, including neural architecture search (NAS), threshold optimisation, and Gaussian scaling processes.

Unsupervised learning is introduced through the third component, namely the SOM component. In this component, a secondary machine learning model is trained for locating the most prominent features for each of the anomalies detected. The SOM makes use of the transactional and network features stored in the database, and its results are reported to the user, allowing him/her to perform a visual inspection of the contributing features. This component complements the AD component, by providing additional information for decision makers to understand the nature of feature behaviour. Thus, anomaly classifications are still made exclusively by the AD component of the framework.

Visual analytics is included through the introduction of the VA component, which combines the outputs from the AD and SOM components by displaying their results. These results include the predicted anomalies retrieved from the anomaly detection model, the performance metrics for the model, and the top n features taken from the self-organising map.

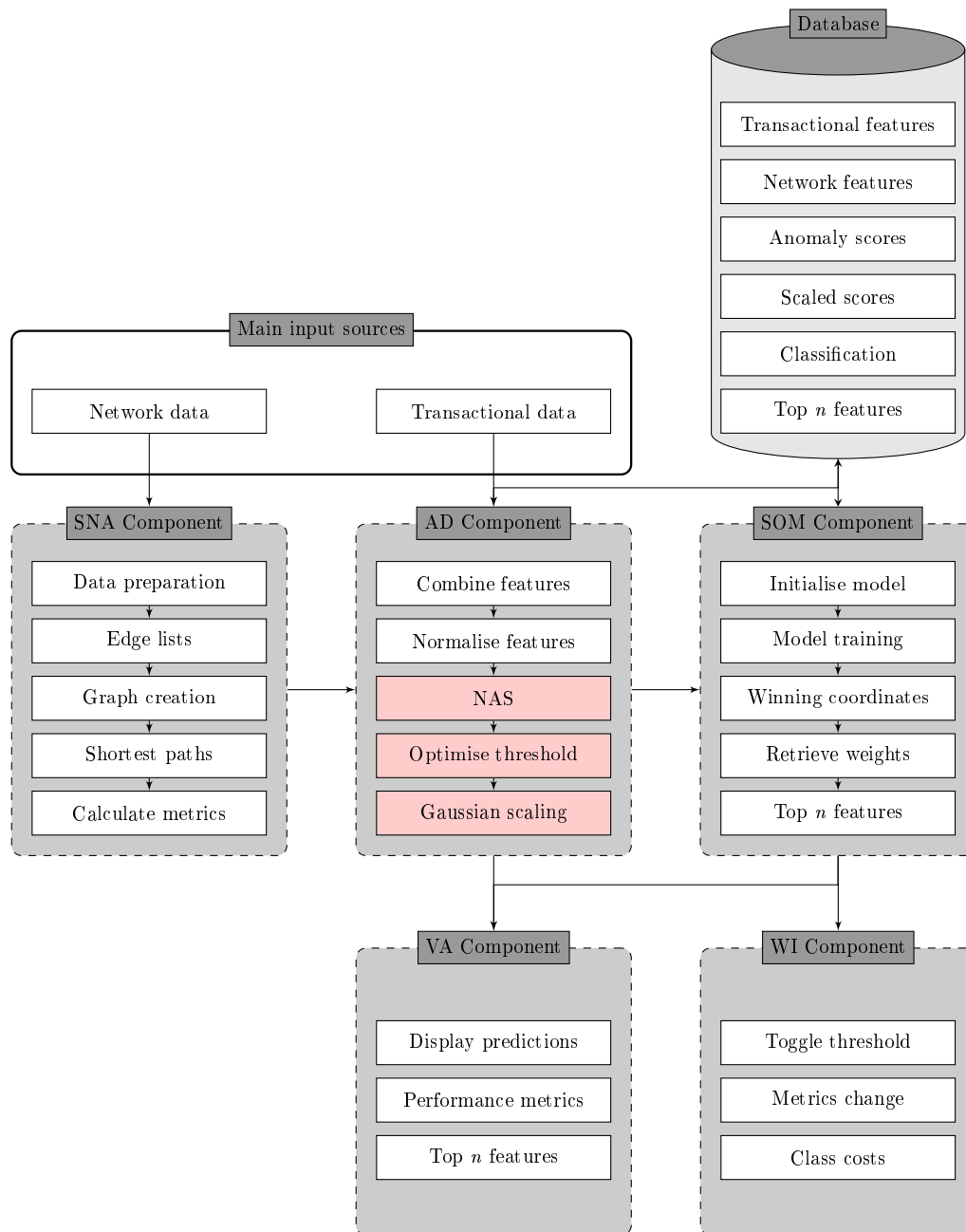


Figure 8.1: High-level overview of the anomaly detection framework

Lastly, what-if analysis is possible through the introduction of the WI component. Users can select a classification threshold for the anomaly detection problem, while simultaneously analysing the subsequent changes in the performance metrics. Misclassification costs can also be assigned, allowing the user to understand the total cost of fraud, depending on the threshold. The remainder of this chapter is structured to provide more detailed descriptions of each of the components in the anomaly detection framework, as well as how each component interacts with other components, the database, or user input.

In terms of how the framework is run in practice, both online and batch inference are possible. The AD component currently runs in batch mode, but could easily be substituted with an online version by hosting a model API (application programming interface). The network metrics in the SNA component are more expensive to compute when compared to the transactional features, although both could be calculated in real-time, depending on the hosting organisation's infrastructure. The more explorative aspect of the anomaly detection framework, the SOM component, is intended to be run offline and therefore there are no perceived benefits to running this in an online mode. In this study, a batch version of the framework is presented as an online version that requires significant infrastructure resources to keep the model API active.

8.3 The social network analysis component

The first component of the anomaly detection framework, the SNA component, has five underlying modules. Raw network data acts as the input data for this component, which would need to be received by the component in a predetermined format, namely as a comma-separated values (CSV) file. An overview of the SNA component is provided in Figure 8.2.

After the network data is received by the SNA component, a process of data preparation ensues, as per the data preparation module. This process involves reading the network data into the system as a data frame, which is a data structure consisting of rows and columns, and which allows for arithmetic operations to be performed on the rows and columns. *Pandas* (Python Data Analysis Library) (McKinney, 2010), a Python-based library for data analysis and manipulation is used for this module.

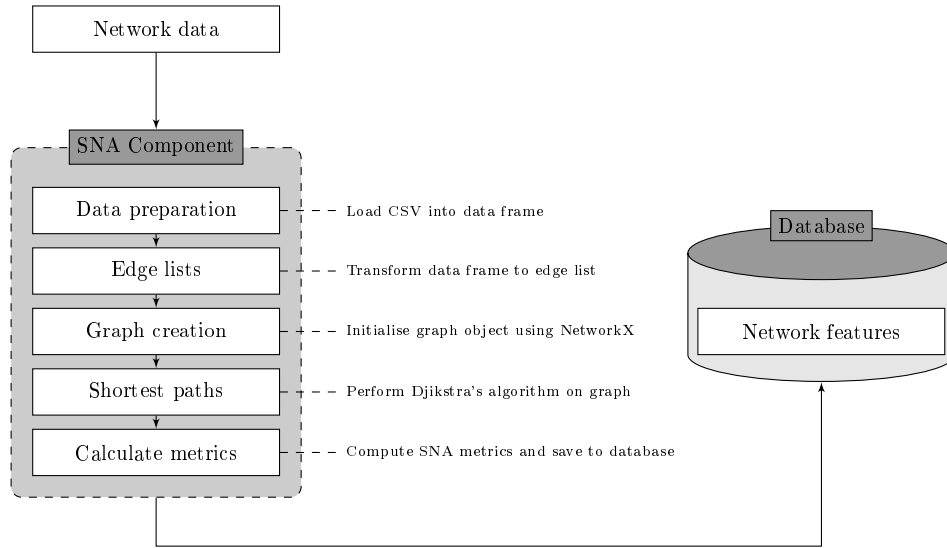


Figure 8.2: Overview of the SNA component within the anomaly detection framework

The edge list creation module follows, where the data in the data frame structure is manipulated to be used in the graph creation process in the next module. Edge lists in this context are data frames that contain variables related to the source, target, edge, and edge attribute data. The source is the node where the linked relationship starts, while the target is the node that receives the link from the source node. Edge data relates to the unique identifier that describes the link between the nodes, and the edge attribute describes the edge category, *i.e.*, is the edge related to a device token or an IP address.

NetworkX (Hagberg *et al.*, 2008), the Python library for working with complex network data, is used to create the graph, as part of module three of the SNA component. Once the graph is populated with the edge list data, the graph can be traversed to find the shortest path between certain nodes within the network. An implementation of Dijkstra's algorithm is applied, using *NetworkX* to achieve these shortest paths. Following this, the network metrics algorithms identified in Chapter 3 are applied. These include centrality, clique, clustering, link analysis and assortativity metrics. This step can take some time to execute, as the network metrics calculations are somewhat computationally expensive. Once these are complete, the outputs are stored in the database as network features.

8.4 The anomaly detection component

The second component within the anomaly detection framework is the Anomaly Detection (AD) component. This component is particularly important to this study, as it contains the proposed unified approach. An overview of the AD component is illustrated in Figure 8.3. The modules related to the unified approach are coloured in a light red colour. Transactional data acts as the direct input to this component, where the user provides a raw CSV file containing the transactional information, as extracted from the organisation's source systems.

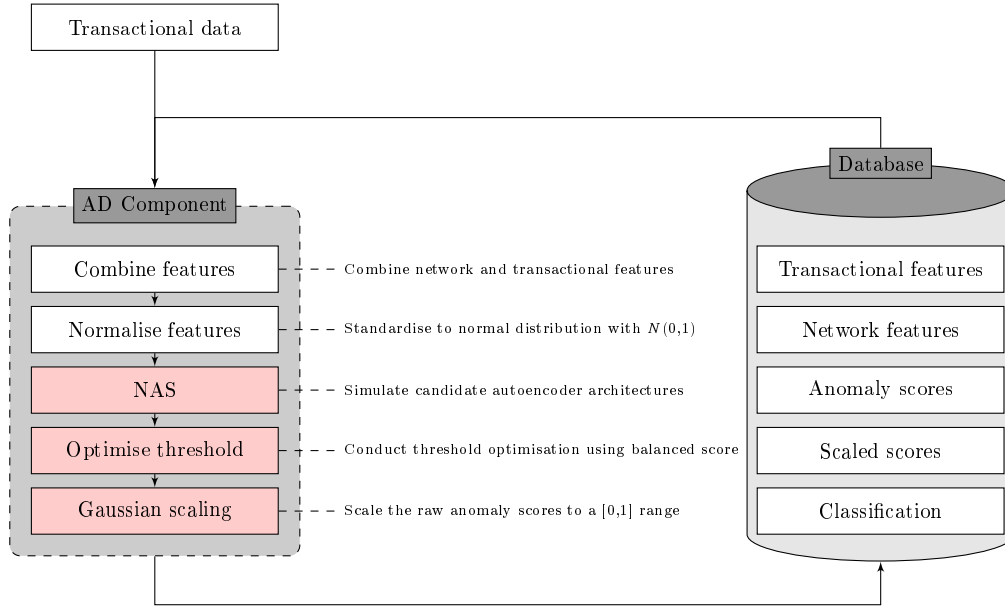


Figure 8.3: Overview of the AD component within the anomaly detection framework

Once the transactional input data is received, the first module combines the transactional data with the network features. These network features were saved to the database in the previous SNA component. The features are then combined, again using the Pandas library, and the combined feature set is normalised. The normalisation module makes use of the *scikit-learn* tool in Python, which is a library dedicated to machine learning and statistical modelling (Pedregosa *et al.*, 2011).

The normalised features are passed to the third module in the AD component, which is the starting point of the unified approach. A neural architecture search is conducted on the normalised feature set by making use of the *Keras* deep learning API (Chollet, n.d.). This process involves simulating a population of candidate autoencoder architectures, while simultaneously making optimal threshold choices, based on a balanced score. The third and fourth modules are therefore interrelated, as there is a back and forth relationship between them as each new candidate architecture is discovered and evaluated.

After the determination of the optimal architecture according to the balanced score, the fifth module commences. In this module, Gaussian scaling is applied to the raw anomaly scores returned by the model. The reasons for the Gaussian scaling process are twofold, namely to provide users with a more interpretable and universally understood range of probability scores and to provide better separation between the class predictions.

Lastly, the raw anomaly scores, as well as the Gaussian scaled version of the scores are written back to the database. The binary classification results, based on the optimal threshold determined during module four, are also written to the database. The classifications indicate a 1 if the record has been classified as an anomaly, or a 0 if the model classified the record as non-anomalous.

In summarising the events that impact the database's interaction with the AD component, two data objects are retrieved from the database, namely the network and transactional features. Three data objects are then also returned to the database, which is the trained model's prediction results.

8.5 The self-organising map component

Unsupervised learning is introduced into the anomaly detection framework via the SOM component. In this component, a self-organising map model is trained without using training data labels. The benefits of this approach are that the SOM model is easy to initialise and train, and provides much faster access to feature contributions when compared to an alternative approach to feature contributions, such as SHAP. The SOM component aims to provide decision makers with the feature interpretation and anomalous behaviour understanding capabilities that a trained SOM model exhibits. Figure 8.4 illustrates the SOM component.

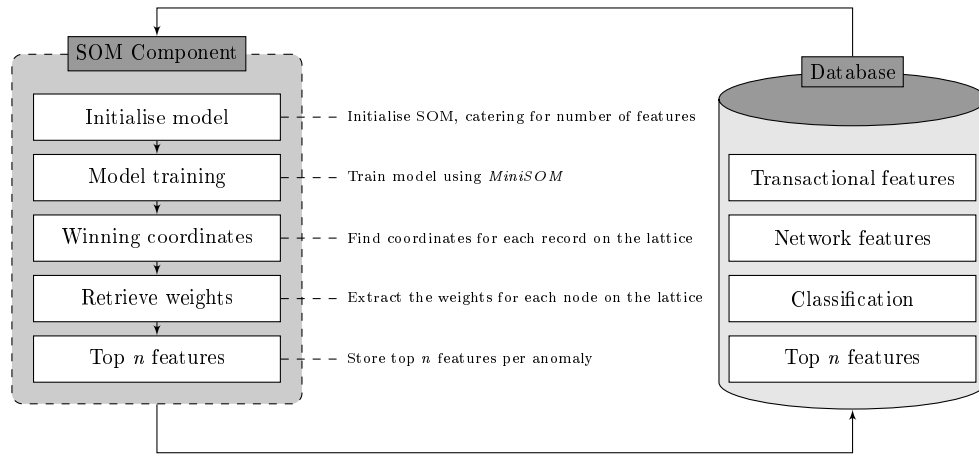


Figure 8.4: Overview of the SOM component within the anomaly detection framework

Firstly, the SOM model is initialised according to the design space requirements. This involves specifying the dimension of the lattice. Considering that the high-dimension space contains 61 features in the case study presented, a 7 x 7 lattice is used for model training purposes. The same combined and normalised features used in the AD component are again used to train the SOM model, as per module two.

One of the benefits of the SOM is that the winning coordinates for each clustering decision can be retrieved, with the weights of the underlying nodes indicating the importance of each feature in the decision. These are achieved by retrieving the AD component classifications from the database and overlaying them onto the SOM lattice to determine where the anomalies lie. Module four is the process of retrieving the weights, while module five selects the top n features for each record. Initially, retrieving the top three features for each record would be a satisfactory choice.

The involvement of the database in the SOM component entails the retrieval of the network and transactional features, the classifications from the AD component, as well as the top n features for each record being written back to the database for later visual analysis purposes.

8.6 The visual analysis component

Making the anomalies detected by the framework more accessible to the end user is vital. Without an informative visual analysis component, insights related to the anomalies detected might not be recognised by the decision makers using the framework. The Visual Analysis (VA) component provides an interface whereby practition-

ers can easily work through the anomalies that the framework detected. An example of the VA component is displayed in Figure 8.5.

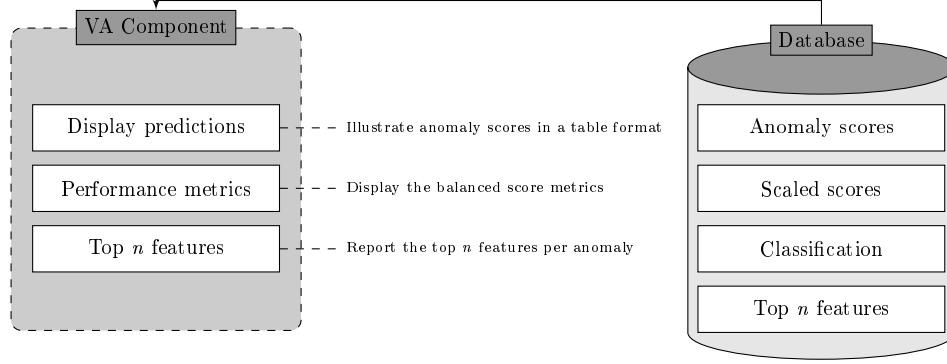


Figure 8.5: Overview of the VA component within the anomaly detection framework

A table that displays the predictions is generated by module one, while module two provides a summary of the performance metrics as they relate to the classification performance of the model. The top n features for each of these predictions are also displayed, so that the user can make the connection between recent high anomaly scores and an emerging feature contribution, among other possible uses.

The VA component does not store any results back into the database, but rather it visualises the data related to raw anomaly and scaled scores, classifications according to the threshold set, and the top n features per each of the anomalies detected by the framework.

8.7 The what-if component

While the VA component is crucial from a reporting perspective, it does not allow the user to simulate different problem scenarios. The What-if (WI) component is introduced to provide the what-if functionality that allows users to conduct these simulations. Saltelli *et al.*, (2019) explain that what-if or sensitivity analysis is the study of how the uncertainty in the output of a model can be apportioned to different sources of input in the model's input. The reader is referred to Figure 8.6 for an overview of the WI component within the anomaly detection framework.

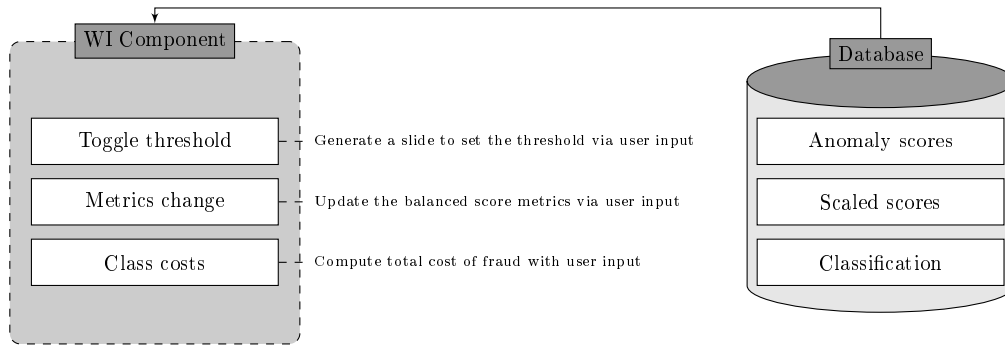


Figure 8.6: Overview of the WI component within the anomaly detection framework

The first module of the WI component is the interface that allows users to adjust the classification threshold. The performance metrics of interest in this study include the same classification metrics used in the balanced score in Chapter 4, namely the average precision, area under the curve, and normalised Matthews correlation coefficient scores. Changing the threshold impacts the performance metrics directly, and therefore these changes need to be understood.

Module two provides a view of the changing classification performance metrics, which includes a dynamic response from the database. An example of how a threshold change might impact model performance could be decreasing the anomaly detection threshold, which in turn leads to an increase in the number of incorrectly classified non-anomalous transactions (false positives) to potentially increase the number of correctly classified anomalies (true positives). The threshold is applied to the scores that are already saved in the database and therefore does not interact with the model directly. Thus it is expected that the response would be immediate. If a threshold change is made, then the performance metrics are recalculated, based on the new classifications made following the change.

Another important consideration in detecting anomalies is the cost of misclassifying anomalies, which is the basis of module three. As mentioned previously in Chapter 2, fraud and anomaly detection problems typically involve using a human analyst in the loop, to verify the anomalies detected. False positives and false negatives both incur different costs per misclassification, and it is useful information for a decision maker to understand these costs under the different possible classification thresholds. The costs also change over time, and therefore classification decisions made at training time do not always align perfectly with the most recent costs realised in the real world. Ultimately, module three allows a user to determine the total cost of fraud, by weighting the misclassifications according to their unit cost

per misclassification and summing the result. The total cost can then be inspected visually at different thresholds to inform further decision making.

The WI component, much like the VA component, does not store results back into the database. Rather, it acts as a presentation layer that displays data that is retrieved from the database. Both the interfaces for the VA and WI components are built using *Streamlit*, a Python-based tool for building data applications.

8.8 Visualisation of output results

In this section, the computer-based visual representation of the anomaly detection framework is presented. Various modules from the VA and WI components make up the structure of the visual representation. The toggle threshold module from the WI component is displayed in Figure 8.7. The threshold represents the mean square error (MSE) of the AD component predictions. An upper limit of 20 is placed on the toggle, as very large anomalies would make toggle selection difficult in practice. A default threshold of 5 is displayed for illustration purposes.



Figure 8.7: Example of the toggle at the default threshold

Figure 8.8 depicts an example of an updated threshold on the toggle. In this instance, a lower threshold of 4.35 is selected, which provides a superior split between the non-anomalous and anomalous classes when compared to the default threshold. Reducing the threshold further would hurt the framework’s ability to detect anomalies, as an increasing number of non-anomalous transactions would be incorrectly classified as anomalous.

Determining an optimal threshold is dependent on the problem that needs to be solved, as well as the choice of performance metrics for which the problem is optimised. While the default threshold can be inherited from the validation threshold obtained when the autoencoder was trained, there is no guarantee that this threshold will generalise well to all business environments. The cost of fraud presented later in this chapter can change rapidly, depending on the business environment, and therefore fraud detection practitioners may wish to adjust the threshold to optimise for the lowest possible total cost of fraud.

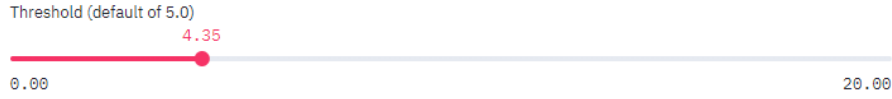


Figure 8.8: Example of the toggle at the updated threshold

Changing the threshold via the toggle updates a visualisation of the threshold, as presented in Figure 8.9. The default threshold is displayed on the left side of the chart, while the updated threshold is displayed on the right of Figure 8.9. The visualisation uses the same reconstruction error plot used in Chapters 4, 5, and 6, where the green points are the normal records and the red points are the anomalies. A blue line is used to represent the threshold, which attempts to split the data into non-anomalous and anomalous classes. Figure 8.9 is critical, as it allows practitioners to visually inspect the decision boundary for classifying anomalies. The higher baseline threshold on the left side of Figure 8.9 classifies fewer anomalous transactions correctly while decreasing the threshold on the right shows that all the anomalous transactions are detected.

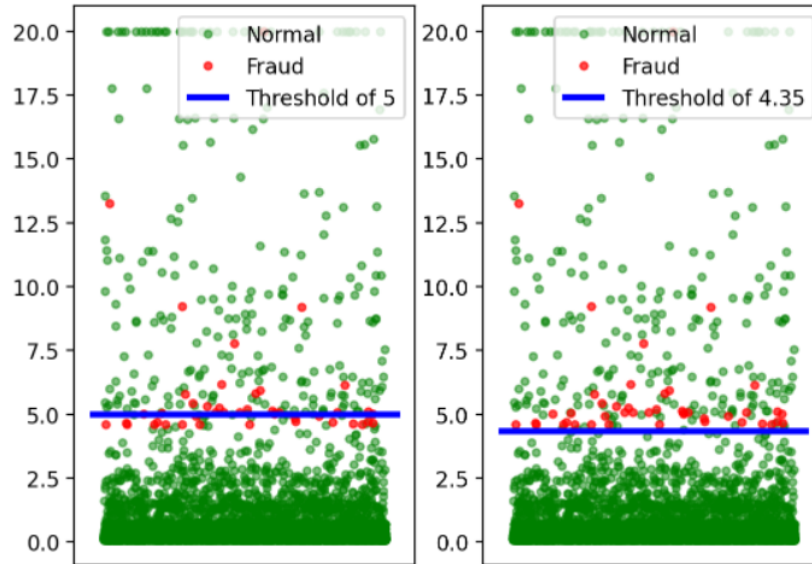


Figure 8.9: Threshold visualisation with baseline threshold of 5 (left subplot) and adjusted threshold of 4.35 (right subplot)

Classification performance metrics are then presented in Figures 8.10 and 8.11. According to the default threshold, displaying these metrics is an application of

the second module from the VA component, performance metrics. Adjusting the threshold triggers a change in the performance metrics, which represents the second module from the WI component.

Metric name	Score	Cut-off score
Area under the curve	0.7858	0.8
Average Precision	0.0610	0.01
Matthews Correlation Coefficient	0.2382	0.2
Balanced score	0.3617	
Threshold	5	
Number of fraudulent transactions detected	26	
Number of fraudulent transactions not detected	18	
Number of good transactions classified as fraudulent	232	
Number of good transactions classified as good	11762	
Total number of transactions assessed	12038	

Figure 8.10: Performance metrics for default threshold

The balanced score shown in Figure 8.10 is the average of the area under the curve (AUC), average precision (AP), and the Matthews correlation coefficient (MCC). Besides the scores, the metrics table also shows the threshold that was applied, the number of anomalous transactions detected by the underlying model, as well as the typical cut-off score for each of the performance metrics.

The cut-off scores are baselines for each performance metric and are expected to be set beforehand in consultation with organisational stakeholders. The cut-off acts as a minimum, therefore it is desirable for a score to be above the cut-off. An example might be to include a cut-off score of 0.8 for the area under the curve metric, based on industry standards or some other expert insight. The cut-offs intend to act as baselines for what might typically constitute good model performance. Two of the three cut-off scores are achieved at the default threshold setting (Figure 8.10), while all three are comfortably surpassed when the threshold is updated (Figure 8.11).

Figure 8.11 is an example of the performance metrics table after updating the threshold to 4.35. The newer, updated threshold improves all the performance metrics scores, which are all above their respective cut-off scores. The total number of anomalies detected by the system has now also increased from 26 in Figure 8.10 to 44 in Figure 8.11.

Metric name	Score	Cut-off score
Area under the curve	0.9895	0.8
Average Precision	0.1492	0.01
Matthews Correlation Coefficient	0.3821	0.2
Balanced score	0.5069	
Threshold	4.3500	
Number of fraudulent transactions detected	44	
Number of fraudulent transactions not detected	0	
Number of good transactions classified as fraudulent	251	
Number of good transactions classified as good	11743	
Total number of transactions assessed	12038	

Figure 8.11: Performance metrics for updated threshold

Figure 8.12 displays a sample of the predictions made by the AD component, representing the first module of the VA component. The unique identifier for each user being scored by the AD component is listed in the *user id* column. Following this, the raw anomaly score, the Gaussian scaled equivalent, and the applied threshold are depicted in the next three columns. If an anomaly score is higher than the threshold, then the transaction is classified as an anomaly and is ultimately reported in Figure 8.12. Once the anomalies are identified in Figure 8.12, the *user id* column can then be used to identify the transaction and investigate further.

User ID	Anomaly score	Gaussian score	Updated threshold
1962967514848127000	5.0553	0.9918	4.3500
916069761900426400	8.7343	1.0000	4.3500
8429518200734895000	7.5964	1.0000	4.3500
6939856710276230000	5.1395	0.9929	4.3500
5209478476781027000	5.7563	0.9977	4.3500
2580300194204458000	8.0862	1.0000	4.3500
4323896481535247400	5.4554	0.9960	4.3500
8452517242670209000	20	1	4.3500
4277235554241011000	20	1	4.3500
1295185135202508300	8.6570	1.0000	4.3500
4067848793363794000	6.3230	0.9993	4.3500
372290033801172100	5.0307	0.9914	4.3500

Figure 8.12: A sample of the anomalies detected

The third module of the VA component, the top n features, is also expressed as the last three columns of the predictions tables used to generate Figure 8.12, presented here as Figure 8.13. These columns, *Feature 1 name* through *Feature 3 name*, show

the three most significant contributing features, based on the SOM component. As an example, the first row in Figure 8.13 indicates that an anomalous record was detected where the corresponding contributing features are *clustering coefficient*, *device tokens per user*, and *average neighbour degree*. These are a mix of network metrics and transactional features, therefore illustrating the finding that network metrics are significant contributors to detecting anomalies in this study.

Feature 1 name	Feature 2 name	Feature 3 name
clustering	device_tokens_per_user	average_neighbor_degree
clustering	device_tokens_per_user	average_neighbor_degree
pagerank	number_of_cliques	degree_centrality
pagerank	number_of_cliques	degree_centrality
pagerank	number_of_cliques	degree_centrality
pagerank	number_of_cliques	degree_centrality
pagerank	number_of_cliques	degree_centrality
pagerank	number_of_cliques	degree_centrality
pagerank	number_of_cliques	degree_centrality
submitted_drivers_lice...	diff_id_country_first_...	degree_centrality
pagerank	number_of_cliques	degree_centrality
pagerank	number_of_cliques	degree_centrality
pagerank	number_of_cliques	degree_centrality

Figure 8.13: Displaying top n features for a sample of the anomalies detected

The total cost of fraud is calculated in Figure 8.14, which is an application of the third module of the WI component. The user of the anomaly detection framework is prompted to input costs for each classification decision outcome, which are combined to determine an overall cost to the organisation. Assuming that the different types of correct or misclassification would have a different cost assigned, the total cost would change, depending on the number of correct classifications produced by the threshold. The costs in Figure 8.14 are merely for illustration purposes, and the actual costs in the industry might be very different.

After updating the threshold, the total cost of fraud reported in Figure 8.14 dropped from 15 908 to 14 297, based on the inputs provided in the user interface. The highest cost per classification of 100 is assigned to *cost of not detecting anomalous transaction*, as this represents the average amount of a anomalous transaction in the system.

Cost of correctly detecting anomalies

1,00

-

+

Cost of incorrectly classifying normal transactions as anomalous

10,00

-

+

Cost of not detecting anomalous transaction

100,00

-

+

Cost of correctly detecting normal transactions

1,00

-

+

The default cost of fraud is 15908.0

The updated cost of fraud is 14297.0

Figure 8.14: Calculating the total cost of fraud

The second-highest cost of 10 is assigned to *cost of incorrectly classifying non-anomalous transactions as anomalous*, as this involves a potential loss of revenue. Finally, very small costs are assigned to the remaining two costs (both the cost of correct classifications), as the only costs involved in these are the system costs of running the framework.

The costs for all correct classifications and misclassification errors are calculated by multiplying each quadrant of the confusion matrix with its associated cost, where the costs are input by the user of the system. Table 8.1 is an example of a cost matrix, indicating a cost for each correct and incorrect classification. $cost_{TP}$ is the cost associated with correctly detecting anomalous transactions. In practice, the costs involved are likely administration related, including a small cost for dealing with the bank involved in any prospective transaction blocking process.

Table 8.1: Cost matrix

		Predicted class	
		Anomalous	Non-anomalous
True class	Anomalous	$cost_{TP}$	$cost_{FP}$
	Non-anomalous	$cost_{FN}$	$cost_{TN}$

Incorrectly classifying non-anomalous transactions as anomalous, $cost_{FP}$, leads to similar administration costs, as well as potential reputational costs incurred for

blocking a good customer from the service. $cost_{FN}$ is the cost of not detecting an anomalous transaction, where the cost is the size of the transaction. This is typically the largest per-unit cost, as indicated by the cost of 100 in Figure 8.14. Finally, $cost_{TN}$ is the cost of detecting non-anomalous transactions, which would typically be very small infrastructure costs associated with maintaining the system that makes the predictions.

8.9 Chapter conclusion

In this chapter, which was the first chapter in Part IV of the study, the reader was provided with a description of the various components that make up the proposed anomaly detection framework. Components related to social network analysis, anomaly detection, self-organising maps, visual analytics and what-if analysis were introduced and combined into a computer-based framework for anomaly detection in a transactional setting. In the next chapter, which forms the final chapter of Part IV of the study, a discussion on the synopsis and limitations of the overall study will be presented and the key contributions of the study to the literature are consolidated.

Chapter 9

Conclusions and recommendations

9.1 Chapter introduction

This chapter is the second and final chapter of Part IV (Final framework and conclusion), and concludes the study. In Chapter 9, the success of the research study is debated. A synopsis of the study, where each of the chapters within each of the four parts of the study is briefly discussed, is presented. The synopsis also includes a discussion on how the aims and objectives of the study, initially presented in Chapter 1, were achieved. This is then followed by the contributions of the study. Finally, the chapter concludes with a brief discussion on the limitations of the study, as well as directions for future research, based on the results of this study.

9.2 Synopsis of the study

The study contains nine chapters, comprising an introductory chapter followed by four distinct parts. In the introductory chapter, the problem statement and the research approach followed during the study were addressed. This was succeeded by Part I, containing two chapters related to literature reviews. Part II followed, in which model development and illustrative applications were conducted. Part III entailed the proposal of a model extension and real-world validation of the approaches. Lastly, Part IV dealt with the presentation of the final proposed framework and the conclusion of the study.

9.2.1 Introduction to the study

Chapter 1 serves as the introduction to the study and presented the background information, problem statement, aims and objectives, and research methodology applied in the study. A brief introduction to the different research topics, as well as a motivation for the study was highlighted. The main areas related to the study, such as anomaly and fraud detection, social network analysis and self-organising maps were introduced. Based on the contents presented in Chapter 1, a formal research question was then formulated as follows: *How can an anomaly detection framework, comprised of multiple machine learning techniques, improve anomaly detection in a transaction-based system?* The research question served as a guide to the research project and was translated into the primary objective of the study which was stated as the *design, development* and *proposal* of a machine learning-based anomaly detection framework that combines multiple approaches to inform whether a transactional behaviour is likely to be anomalous.

In answering the research question, eight research objectives were formulated:

1. *Conduct* a literature review on the main aspects related to the study, namely anomaly and fraud detection, autoencoder neural networks, social network analysis, and self-organising maps;
2. *Establish* an appropriate autoencoder architecture to detect anomalous behaviour;
3. *Extend* the formulated autoencoder model by *identifying* and *incorporating* applicable social network analysis metrics into the proposed machine learning-based anomaly detection framework;
4. *Develop* a self-organising map functionality to form part of the proposed machine learning-based anomaly detection framework. More specifically, the new functionality should assist with *categorising* behaviours into clusters and provide an additional intuitive understanding of which anomaly-related variables contributed to the decision-making process;
5. *Demonstrate* the adequacy and validity of the developed techniques using existing example data sets;
6. *Identify* and *implement* a real-world application data set to demonstrate and validate the performance of the proposed machine learning-based anomaly detection framework and its underlying techniques;
7. *Develop* and *propose* a final machine learning-based anomaly detection framework that provides a user-friendly interface that is capable of visually presenting results, as well as answering appropriate what-if questions; and

8. *Recommend* applicable future research opportunities based on the work presented by this thesis.

The four main parts of the study were designed to address each of the stated objectives and are summarised below.

9.2.2 Part I: Literature review

Part I involved a detailed investigation of research related to autoencoders for anomaly detection, social network analysis, self-organising maps, and combining machine learning approaches in anomaly detection systems, and consists of Chapters 2 and 3. Starting with Chapter 2, research related to fraud and anomaly detection, including an introduction to credit card fraud, fraud detection systems, approaches to fraud detection via supervised and unsupervised machine learning, and class imbalances problems were conducted. Autoencoders were then introduced, including a discussion on the various architectures, namely undercomplete, regularised, denoising, contractive, and variational autoencoders. Network architecture is an important consideration for autoencoders, and therefore the chapter concluded with a discussion on architecture considerations followed by the various applications for applying autoencoders for anomaly detection. The study of related works determined that self-organising maps are popular techniques for clustering and feature interpretation, with applications ranging from risk and anomaly detection to medical and industrial applications.

Chapter 3 followed with a literature review on social network analysis and self-organising maps. A review of self-organising maps introduced the chapter and topics related to the building, training, evaluating, and interpreting of self-organising maps were discussed. Various visualisation techniques were introduced as a means of interpreting the output of self-organising map models. Examples of the techniques discussed include the clustering map, u-matrix, and component plane visualisations, which were all found to complement one another in understanding clustering decisions from multiple analytical perspectives. A variety of works related to the application of self-organising maps were then presented.

Following this, the social network analysis component of the review was conducted, covering various graph theory-related concepts, such as undirected, directed, and weighted graphs. The use of network metrics in anomaly detection was then introduced, including motivation for using network metrics to augment features in a machine learning context. It was discovered that multiple network metrics could be suitable node features, specifically those related to centrality, clique, clustering, link analysis and assortativity. The chapter concludes with a summary of other studies related to the application of social network analysis, indicating broad adoption

of the techniques across fraud detection, healthcare, geographical, organisational, networking, and computing environments.

Part I answered objective 1, related to *conducting* a literature review on the main aspects of the study, namely anomaly and fraud detection, autoencoder neural networks, social network analysis, and self-organising maps.

9.2.3 Part II: Development and illustrative applications

The second part of the study pertained to the development and illustrative applications and contains a single chapter, Chapter 4. Initially, a discussion on the use of autoencoders for anomaly detection in a transaction-based system was presented. A public data set derived from *Kaggle* was used to illustrate the practical application of autoencoders. A description of the data set used for illustration purposes, involving the visual representation of the latent space using t-SNE as well as a sampling approach to manage the class imbalance in the data set was presented. This was followed by one of the key contributions of the study, *i.e.*, the unified approach for anomaly detection. The unified approach is comprised of three parts, including a neural architecture search, threshold optimisation, and a Gaussian scaling process. The neural architecture search process entailed simulating a set of candidate autoencoder architectures to retrieve an optimal model by determining a balanced score. According to the literature, the balanced score was proposed as an average of three important classification metrics for fraud detection problems. The three scores used in the balanced score were the area under the curve, average precision, and a normalised Matthews correlation coefficient. A proposed threshold optimisation algorithm was included in the unified approach to determine the optimal classification threshold value across all candidate architectures.

After the optimal model was determined, Gaussian scaling was applied to the output of the model to transform the raw anomaly scores into a distribution of scores that are simpler to interpret and manage in comparison with raw anomaly scores. An evaluation of the performance of the optimal architecture was then conducted, involving error analysis through the construction of a confusion matrix. Chapter 4 was concluded via an evaluation of the results, including a proposal to extend the unified approach by including network metrics in the feature space. The success of Part II formed the cornerstone of the process that leads to the final proposed anomaly detection framework, ensuring that the modelling generates accurate and effective results.

Part II answered objectives 2 and 5, which related to *establishing* an appropriate autoencoder architecture to detect anomalous behaviour and *demonstrating* the adequacy and validity of the developed techniques, respectively using existing example data sets.

9.2.4 Part III: Model extension and real-world validation

The penultimate section of the study, Part III, involved the proposed model extension and real-world validation and includes Chapters 5, 6, and 7. Firstly, in Chapter 5, a real-world data set, sourced from an industry partner to validate the approaches presented in Chapter 4 was introduced into the study. A description of the data set, including the imbalance between the non-anomalous and anomalous classes, as well as the standardisation approach applied, was presented. To understand whether some discrimination between the classes was possible, t-SNE was applied to the features to highlight possible clusters within the latent space. A specialised stratified sampling process was conducted to prevent bias when sampling from the data set for model training, validation and testing. Performing a neural architecture search over a large set of candidate architectures, in conjunction with the proposed threshold optimisation algorithm, yielded an optimal architecture configuration that performed well across all metrics within the balanced score. Inspecting the latent representation of the autoencoder in two dimensions confirmed that significant separation between the classes was achieved. The model evaluation involved both error analysis via the interpretation of a confusion matrix, as well as the introduction of Shapley values to measure feature contribution. Objective 6, which was to *identify* and *implement* a real-world data set to demonstrate and validate the performance of the proposed machine learning-based anomaly detection framework and its underlying techniques, was achieved in Chapter 5.

An extension of the proposed unified approach was introduced in Chapter 6. The chapter began with a description of the network data, which was used to connect users within the financial ecosystem using links such as shared IP addresses, cellular devices, or payment details. Various node-level network metrics were introduced and calculated, including degree and eigenvector centrality, as well as the popular PageRank algorithm metric. Confirmed anomalous nodes were then mapped onto the resulting network of linked users, where Dijkstra’s algorithm was used to determine the shortest path between all nodes in the network. The shortest paths between all nodes and their closest associated anomalous nodes were determined and included in the set of network metrics. The extension of the unified approach involved the inclusion of the network analysis metrics in the feature set, where a second autoencoder was trained on the combined transactional and network metrics feature set, to enhance the prediction abilities of the original model.

The combined transactional and network metrics feature was used to construct an optimal architecture, determined through a pass of the proposed unified approach. Shapley values were calculated on the optimal model, to determine whether the network metrics feature contributed to the detection of anomalies. Practical considerations for the implementation of the approach followed, where topics related to the

ease of constructing model, model run times, model complexity, and the prerequisite knowledge required for implementation were discussed. Finally, recommendations were made, based on the work presented in this chapter. Objective 3, which was to *extend* the formulated autoencoder model by *identifying* and *incorporating* applicable social network analysis metrics into the proposed machine learning-based anomaly detection framework, was achieved in Chapter 6.

Self-organising maps were introduced in Chapter 7 to provide additional insights into the contributions of the features used within the modelling process. Firstly, a discussion on model building for self-organising maps was introduced, covering topics related to the design of the lattice onto which the lower-dimensional representation produced by the model is projected. A discussion on model training followed, which included an analysis and interpretation of the topographic and quantisation errors. An evaluation of the model outcomes was then conducted, which entailed the production of various visualisations, namely the node activation frequency, class distribution, class proportion, u-matrix, and clustering map visualisations. The class distribution and u-matrix visualisations proved to be particularly informative from an anomaly detection standpoint, while the class proportion visualisation appeared to be dominated by the majority class, therefore proving to be less effective at detecting anomalies. Feature contribution was then measured via the construction of component plane visualisations. Finally, a discussion of the results ensued, where the benefits of the clustering approach of the self-organising map model are proposed as useful in augmenting a classification approach. Objective 4 was achieved in Chapter 7, which was related to *developing* a self-organising map functionality to form part of the proposed machine learning-based anomaly detection framework. More specifically, the new functionality assisted with *categorising* behaviours into clusters and provided an additional intuitive understanding of which anomaly-related features contributed to the decision-making process.

9.2.5 Part IV: Final framework and conclusion

The final part of the study dealt with the presentation of the proposed framework and the conclusion of the study. In Chapter 8, a discussion on details related to the design, development and implementation of the framework, followed by the illustration of a computer-based representation of the framework was included. The proposed anomaly detection framework was made up of five components, two sources of input data, as well as a database for storage purposes. Social network analysis, anomaly detection, self-organising map, visual analysis, and what-if analysis made up the five components of the framework. Two sources of input included the raw files for creating the network metrics and transactional anomalies. A database was used to store all features, anomaly and scaled scores, the classifications made by the model, and the top contributing features per classification decision. After the def-

inition of the framework components and artefacts, the visualisation of the output results was presented. Some of the significant features in the visualisation included a toggle for manual threshold selection, as well as a threshold visualisation comparing a baseline threshold to any user-specified threshold. Various performance metrics were also introduced, including those related to the balanced score, as well as the raw numbers for each type of correct or incorrect classification. A sample of the top n features per anomaly detected was also displayed, indicating the importance of the network metrics for many of the anomalies detected. Finally, a screen dedicated to the calculation of the total cost of fraud was presented. This screen is intended to provide users with an interface whereby operational costs can be included in the process in order to include relevant costs in any thresholding decisions made. Objective 7, which aimed to **develop** and **propose** a final machine learning-based anomaly detection framework that provided a user-friendly interface that was capable of visually presenting results as well as answering appropriate what-if questions, was achieved in Chapter 8.

In Chapter 9, which concluded the study, there is a description of how the aims and objectives of the study have been achieved. An appraisal of the contributions as well as recommendations for further and follow-up studies was provided. Objective 8, which relates to **recommending** applicable future research opportunities, based on the work presented in the thesis, was achieved in Chapter 9.

9.3 Contributions of the study

The primary contribution of this study is the design, development and proposal of a new machine learning-based framework for anomaly detection. The framework contributes to anomaly detection systems by combining multiple machine learning approaches to inform whether a transactional behaviour is likely to be anomalous. This section contains the important contributions that are extended by the proposed anomaly detection framework.

Contribution I *The literature review produced a valuable framework, beyond that in which this study was grounded, serving as a useful resource and reference framework for future research related to the field of machine learning-based anomaly detection models and frameworks*

The literature reviews conducted in Chapter 2 and Chapter 3 provided an evaluation of the topics related to fraud and anomaly detection, autoencoders, self-organising maps, and social network analysis. More specifically, these reviews uncovered the important theoretical underpinnings of anomaly detection, as well as how each of the aforementioned techniques contributes to the detection of anomalies in a transactional setting. Concepts critical to the design of fraud detection systems were

reviewed, including label feedback loops, relevant performance metrics, supervised versus unsupervised learning, class imbalance, concept drift, and streaming data. This was followed by an appraisal of autoencoders, as well as the benefits of applying these models to detect anomalies in high-dimensional feature sets. Self-organising maps were then reviewed which contributed to the understanding of their use as a clustering and feature contribution technique. Finally, a study of network metrics was conducted, informing the decision to include these metrics as features in any anomaly detection efforts.

Contribution II *The proposal of a novel neural architecture search methodology, which was empirically tested and illustrated using an example data set. The methodology contributes to determining an optimal architecture for autoencoder models*

Autoencoder models have been deployed in several transactional environments to detect anomalies that differ significantly from normal transactions (Pang *et al.*, 2021). Determining an effective autoencoder architecture can be achieved via several techniques, namely manual search, grid search, or randomly searching the parameter space (Bergstra and Bengio, 2012). A neural architecture search is an approach for automatically engineering relevant architectures for neural network models, and due to the seemingly infinite number of hyperparameter configurations, neural architecture search is considered more effective when training complex models, such as autoencoders (Feurer and Hutter, 2019).

A novel, unified approach for autoencoder architecture was proposed in Chapter 4 of this study to address the shortcomings of the manual and grid search techniques. This was achieved by incorporating a novel threshold optimisation algorithm into the architecture search process, where the optimal threshold is determined for each randomly generated candidate architecture. The optimal threshold was determined by optimising for a balanced score, which combined three classification metrics proven to be effective for anomaly detection, namely the area under the curve, average precision, and Matthews correlation coefficient scores. The unified approach also included a Gaussian scaling process to transform the resulting raw anomaly scores into an interpretable range of scores. Sampling for the approach was conducted in a stratified manner, to remove the bias imposed by the class imbalance. Shapley values were also computed on the output scores generated by the optimal autoencoder model, providing an element of model interpretation and feature contribution to the unified approach.

Contribution III *Social network analysis extends the feature space of the initial anomaly detection model, by capturing the relationships between transactions within the system. The inclusion of the social network analysis metrics leads to improved prediction capability of the model*

The transactional features used to train the autoencoder model in Chapter 5 were useful in capturing the suspicious behaviours exhibited by individual users within the system, although they lacked sufficient information regarding the connectedness of users within the financial ecosystem. This shortcoming was circumvented by calculating node-level network metrics on data used to link users within the system, following which the network metrics were introduced into the feature space. The network metrics were calculated using social network analysis algorithms, where the literature showed that these were important indicators of influence within a network. Dijkstra’s algorithm was applied to the network to determine the shortest path between each node. The anomalous labels were then projected onto the graph to detect the closest anomalous users to each user within the network. These closest anomalous users’ calculations were then transformed into features to augment the computed network metrics. Following the training of an autoencoder on the feature set inclusive of the network metrics, it was shown that the network metrics improved model performance. Computing Shapley values for the anomalous records illustrated that network metrics were significant contributors to detecting anomalies, confirming the success of the inclusion of the network metrics.

Contribution IV *The introduction of self-organising map analysis provided techniques to analyse variables and detect different anomalous behaviours*

By framing the fraud detection problem presented in Chapter 5 as a binary classification problem, the ability to isolate distinct anomalous subgroups becomes a challenge. Anomaly detection problems are generally modelled as a binary classification problem, with a majority class of normal records, and a minority class of anomalous records. An anomaly is by definition something that differs significantly from the norm. Therefore it is assumed that the nonanomalous class is somewhat homogenous. Conversely, it is assumed that anomalous behaviour is not necessarily homogenous, rather it simply differs significantly from the normal, majority class. Thus, several potential anomalous subgroups can occur, which binary classification algorithms are not suited to detecting.

The use of self-organising maps was proposed to explore the possible anomalous subgroups. Introducing self-organising maps in Chapter 7 provided an unsupervised approach for clustering the data without the use of the target variable. By projecting the target variable onto the clustering output, it was possible to detect more than one type of anomalous behaviour, manifesting as multiple clusters containing anomalous records. Analysis of the trained model weight parameters revealed the significant and contributing features for each cluster, which subsequently confirmed the different behavioural traits of these anomalous groups. The component planes generated by the model output also contributed to determining the influence of the network metrics in detecting anomalies in the data, further supporting **Contribution III**.

Contribution V *Applying the proposed models and techniques in a real-world setting validated the approaches*

The practical applicability of the proposed unified approach was evaluated using real-world data in Part III of this study. The unified approach was shown to be successful at overcoming various shortcomings of anomaly detection approaches mentioned in the literature. More specifically, the effectiveness of the approach was illustrated in the context of a real-world setting through the application of the approach to a data set containing anomalies sourced from an organisation in the financial services industry in Chapter 5. The data used in the study was labelled instances of credit card fraud experienced by customers of the organisation. The set comprised transactional data relating to the financial behaviour of individual users, as well as network data which measured the connectedness of the same users within the system.

Considering that the amount of fraud in the system was significantly lower than the number of normal transactions, there was a strong imbalance in data relating to the normal transactions. Regardless of the imbalance, which is typical of anomaly detection problems, an optimal autoencoder model was trained through the application of the unified approach. The optimal model achieved a validation balanced score of 89.7% when trained on the transactional features only, and 94% when trained on the combined transactional and network metrics feature set. Incorporating Gaussian scaling into the approach yielded output scores that are universally understood, as risk scores are typically in the range $[0;1]$. The results of the application of the approach to real-world data were summarised and discussed in more detail in Chapter 6, Section 5.

Contribution VI *The proposal to combine the above-mentioned work into a comprehensive machine learning-based framework for anomaly detection*

The anomaly detection framework proposed in this study addresses several shortcomings in modelling approaches identified in the literature. Firstly, the proposed anomaly detection framework presented in Chapter 4 is flexible enough so that it can be applied in any number of anomaly detection contexts, and was not designed to cater for fraud detection problems only. The framework was designed in a modular fashion, therefore the components can be amended, depending on the application context. Secondly, the framework can adapt to different input data. If the social network analysis component were to be removed from the framework, then the input dimension of the autoencoder would be reduced as part of the neural architecture search process by the reduced dimensionality of the feature space. Alternatively, if additional network metrics were to be included as input data, then the autoencoder and its subsequent processes would be adjusted accordingly. Thirdly, the proposed framework implements multiple modelling approaches, which are performed in a log-

ical sequence to achieve the desired outcome. These approaches include the training of an autoencoder in an unsupervised manner, where labels are used for validation and testing purposes. The problem is also approached from a purely unsupervised learning perspective through the training of self-organising maps, where clustering is used to detect anomalous subgroups and to understand their evolving anomalous behaviours. Social network analysis techniques are also incorporated into the framework, to detect the influence of users within a broader linked network of user activity. Lastly, the proposed framework provides a user interface where operational users can perform what-if analyses to simulate different threshold and cost scenarios, as well as visual analysis which allows users of the system to track performance metrics and monitor the process in detail.

Contribution VII *Suitable recommendations of ideas for follow-up work and further research opportunities*

The final contribution of the study is summarised in Section 9.4. Suggestions for future research opportunities are presented to promote research related to machine learning-based frameworks for anomaly detection. This is achieved by highlighting several suitable avenues of research as potential follow-up research to the contributions of this study.

9.4 Limitations and directions for future research

The study focused on the aims and objectives presented in Chapter 1. The following limitations of the study provide direction for future research.

In this study, no experiments were conducted with upper bounds for the anomaly scores. It is expected that upper bounds may impact the distribution of probability scores – and by achieving a better probability distribution score, the classification performance of the model may increase further. In addition to this, although the proposed neural architecture search procedure delivered excellent results, there are other techniques, such as genetic algorithms (Liu *et al.*, 2021c; Whalen *et al.*, 2019) that may complement the newly proposed approach. Additionally, experiments with other algorithms may prove useful.

During the neural architecture search process of the unified approach, certain architecture design constraints were stated upfront to ensure comparability between simulations. One of these constraints was capping the dimension of the first hidden layer in Chapter 6 to the same limit applied to the first hidden layer in Chapter 5. The dimension of the feature set used to train the autoencoder in Chapter 6 was larger than that used in Chapter 5, which is the reason for applying the limit. Further research could be conducted on understanding the impact of these limits.

Another observation regarding the neural architecture search process is understanding the impact of the size of the decrease or increase in node count from each subsequent layer, as well as how performance is affected when combining certain node count progressions with architectures of varying depths. Determining how these two components impact model performance could be a valuable research area (Matthews *et al.*, 2018).

When training the self-organising map component of the anomaly detection framework, the model is trained from scratch using the same combined feature set used to train the autoencoder model. There is an opportunity to leverage the dimensionality reduction functionality of the autoencoder instead, by using the encoder portion of the autoencoder to reduce the dimensionality of the features to train the self-organising map on the resulting low-dimensional latent space (Yang *et al.*, 2019). This approach could be explored to potentially enhance the utility of the clustering process.

9.5 Chapter conclusion

In Chapter 9, the study was concluded. The chapter began with a synopsis of the study, in which each of the individual chapters and parts was briefly discussed. The synopsis included the identification of the objectives of the study, as well as how each was achieved. The contributions of the study were then presented, followed by an overview of the limitations of the study. Finally, directions for future research were highlighted.

References

- Abe, T., Akazawa, Y., Toyoda, A., Niki, H. and Baba, T. 2020. Batch-learning self-organizing map identifies horizontal gene transfer candidates and their origins in entire genomes. *Frontiers in Microbiology*, 11, Article 1486.
- Acevedo-Viloria, J.D., Roa, L., Adeshina, S., Olazo, C.C., Rodriguez-Rey, A., Ramos, J.A. and Correa-Bahnsen, A. 2021. Relational graph neural networks for fraud detection in a super-app environment. *KDD-MLF Virtual Workshop*.
- Aggarwal, C.C. 2007. *Data streams: models and algorithms*. Vol. 31. Advances in Database Systems. New York: Springer.
- Agrawal, S. and Agrawal, J. 2015. Survey on anomaly detection using data mining techniques. In: *19th International Conference on Knowledge Based and Intelligent Information and Engineering Systems*. Singapore, 708–713.
- Ahmad, T., Li, X.J., Seet, B.C. and Cano, J.C. 2020. Social network analysis based localization technique with clustered closeness centrality for 3D wireless sensor networks. *Electronics*, 9(5), Article 738.
- Ahmed, M., Mahmood, A.N. and Islam, M.D.R. 2016. A survey of anomaly detection techniques in financial domain. *Future Generation Computer Systems*, 55:278–288.
- Akinduko, A.A., Mirkes, E.M. and Gorban, A.N. 2016. SOM: stochastic initialisation versus principal components. *Information Sciences*, 364:213–221.
- Akoglu, L., Tong, H. and Koutra, D. 2015. Graph-based anomaly detection and description: a survey. *Data Mining and Knowledge Discovery*, 29(3):626–688.
- Al-Shabi, M.A. 2019. Credit card fraud detection using autoencoder model in unbalanced datasets. *Journal of Advances in Mathematics and Computer Science*, 33(5):1–16.
- Alazizi, A., Habrard, A., Jacquenet, F., He-Guelton, L. and Oble, F. 2020. Dual sequential variational autoencoders for fraud detection. *Advances in Intelligent Data Analysis*, 12080(18):14–26.

- Alfeo, A.L., Cimino, M.G.C.A., Manco, G., Ritacco, E. and Vaglini, G. 2020. Using an autoencoder in the design of an anomaly detector for smart manufacturing. *Pattern Recognition Letters*, 136:272–278.
- Aly, S. and Almotairi, S. 2020. Deep convolutional self-organizing map network for robust handwritten digit recognition. *IEEE Access*, 8:107035–107045.
- An, J. and Cho, S. 2015. Variational autoencoder-based anomaly detection using reconstruction probability. *SNU Data Mining Center, 2015-2 Special Lecture on IE*.
- Anand, K., Kumar, J. and Anand, K. 2017. Anomaly detection in online social network: a survey. In: *International Conference on Inventive Communication and Computational Technologies (ICICCT)*. Coimbatore, India, 456–459.
- Anderson, R. 2007. *The credit scoring toolkit: theory and practice for retail credit risk management and decision automation*. New York: Oxford University Press.
- Antwarg, L., Miller, R.M., Shapira, B. and Rokach, L. 2021. Explaining anomalies detected by autoencoders using Shapley additive explanations. *Expert Systems with Applications*, 186, Article 115736.
- Araya, D.B., Grolinger, K., Yamany, H.F. El, Capretz, M.A.M. and Bitsuamlak, G. 2016. Collective contextual anomaly detection framework for smart buildings. In: *International Joint Conference on Neural Networks (IJCNN)*. Vancouver, Canada, 511–518.
- Arpit, D., Zhou, Y., Ngo, H. and Govindaraju, V. 2016. Why regularized autoencoders learn sparse representation. In: *Proceedings of the 33rd International Conference on Machine Learning*. New York, USA, 136–144.
- Athanasakis, E. 2020. *Data analysis in environmental and traffic data for Thessaloniki Greece*. Thessaloniki, Greece: Aristotle University of Thessaloniki (Dissertation - MS).
- Axenie, C., Tudoran, R., Bortoli, S., Hassan, M.A.H., Sanchez, C.S. and Brasche, G. 2019. Dimensionality reduction for low-latency high-throughput fraud detection on datastreams. In: *18th International Conference on Machine Learning and Applications*. Boca Raton, Florida, USA.
- Aygun, R.C. and Yavuz, A. Gokhan 2017. Network anomaly detection with stochastically improved autoencoder based models. In: *4th International Conference on Cyber Security and Cloud Computing*. Washington DC, USA, 193–198.
- Bahnsen, A.C., Aouada, D. and Ottersten, B. 2014. Example-dependent cost-sensitive logistic regression for credit scoring. In: *13th International Conference on Machine Learning and Applications*. Detroit, MI, USA, 263–269.
- Bahnsen, A.C., Aouada, D. and Ottersten, B. 2015. Example dependent cost-sensitive decision trees. *Expert Systems with Applications*, 42(19):6609–6619.

- Bahnsen, A.C., Aouada, D., Stojanovic, A. and Ottersten, B. 2016. Feature engineering strategies for credit card fraud detection. *Expert Systems with Applications*, 51:134–142.
- Bahnsen, A.C., Stojanovic, A., Aouada, D. and Ottersten, B. 2013. Cost-sensitive credit card fraud detection using bayes minimum risk. In: *12th International Conference on Machine Learning and Applications (ICMLA)*. Miami, Florida, USA, 333–338.
- Balakrishnan, R. and Ranganathan, K. 2012. *A textbook of graph theory*. Vol. 2. New York: Springer.
- Balasubramanian, M., Polimeni, J.R. and Schwartz, E.L. 2009. Exact geodesics and shortest paths on polyhedral surfaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(6):1006–1016.
- Barabasi, A. 2016. *Network science*. Vol. 1. Cambridge, United Kingdom: Cambridge University Press.
- Barrat, A., Barthélemy, M., Pastor-Satorras, R. and Vespignani, A. 2004. The architecture of complex weighted networks. In: *Proceedings of the National Academy of Sciences*. Vol. 101(11): 3747–3752.
- Behera, R.K., Naik, D., Ramesh, D. and Rath, S.K. 2020. MR-IBC: MapReduce-based incremental betweenness centrality in large-scale complex networks. *Social Network Analysis and Mining*, 10(25):1–13.
- Bender, E.A. and Williamson, S.G. 2010. Lists, decisions and graphs. With an introduction to probability. *San Diego: Edward A. Bender & S. Gill Williamson*.
- Beneish, M.D. and Vorst, P. 2021. The cost of fraud prediction errors. *Kelley School of Business Research Paper No. 2020-551–59*.
- Bengio, Y. 2009. Learning deep architectures for AI. *Foundations and Trends in Machine Learning*, 2(1):1–55.
- Bengio, Y., Courville, A. and Vincent, P. 2013. Representation learning: a review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828.
- Bengio, Y., Lamblin, P., Popovici, D. and Larochelle, H. 2007. Greedy layer-wise training of deep networks. *Advances in Neural Information Processing Systems (NIPS)*, 19:153–160.
- Bergstra, J. and Bengio, Y. 2012. Random search for hyperparameter optimization. *Journal of Machine Learning Research*, 13:281–305.
- Bhaskara, A. 2017. No free lunch theorem. *Theory of Machine Learning, CS 5966 class notes, University of Utah*.
- Bhatla, T.J., Prabhu, V. and Dua, A. 2003. Understanding credit card frauds. *Cards Business Review, Tata Consultancy Services*, 1(6):1–15.
- Bhattacharyya, S., Jha, S., Tharakunnel, K. and Westland, J.C. 2011. Data mining for credit card fraud: A comparative study. *Decision Support Systems*, 50(3):602–613.

- Bhogaram, P., Wu, X., He, M. and Okenwa, O. 2020. Optimal and critical path analysis of state transportation network using Neo4J. *International Journal of Urban and Civil Engineering*, 14(10):312–317.
- Bifet, A., Francisci Morales, G. de, Read, J., Holmes, G. and Pfahringer, B. 2015. Efficient online evaluation of big data stream classifiers. In: *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Sydney, Australia, 59–68.
- Bodaghi, A. and Teimourpour, B. 2018. Automobile insurance fraud detection using social network analysis. *Applications of Data Management and Analysis. Lecture Notes in Social Networks*. Springer, 11–16.
- Bolton, R.J. and Hand, D.J. 2001. Unsupervised profiling methods for fraud detection. In: *7th Conference on Credit Scoring and Credit Control*. London, England.
- Bonelli-Toro, A.G. and Gomez, M.P. 2020. Machine learning applied to acoustic emission tool wear classification during milling of composite materials. *An International Forum for the AE Science and Technology*, 37:11–20.
- Bonilla, J.P. 2018. *Detection of fraud in financial blockchain-based transactions through big data analytics*. Madrid, Spain: Universidad Carlos III de Madrid (Dissertation - MA).
- Borgatti, S., Mehra, A., Brass, D. and Labianca, G. 2009. Network Analysis in the Social Sciences. *Science (New York, N.Y.)*, 323:892–895.
- Borghesi, A., Bartolini, A., Lombardi, M., Milano, M. and Benini, L. 2019. A semi-supervised autoencoder-based approach for anomaly detection in high performance computing systems. *Engineering Applications of Artificial Intelligence*, 85:634–644.
- Bouchet, A., Song, X. and Sun, L. 2020. CEO network centrality and corporate social responsibility. *Social Responsibility Journal*, 18(1):106–127.
- Boughorbel, S., Jarray, F. and El-Anbari, M. 2017. Optimal classifier for imbalanced data using matthews correlation coefficient metrics. *PLoS ONE*, 12(6), Article e0177678.
- Brandes, U. 2008. On variants of shortest-path betweenness centrality and their generic computation. *Social Networks*, 30(2):136–145.
- Breard, G. 2017. Evaluating self-organizing map quality measures as convergence criteria. *Rhode Island, NE: University of Rhode Island (Dissertation - MS)*.
- Buda, M., Maki, A. and Mazurowski, M.A. 2018. A systemic study of the class imbalance problem in convolutional neural networks. *Neural Networks*, 106:249–259.
- Burkhardt, M.C. and Shan, K. 2020. Deep low-density separation for semi-supervised classification. In: *International Conference on Computational Science*. Amsterdam, The Netherlands, 297–311.

- Burnaev, E., Erofeev, P. and Papanov, A. 2015. Influence of resampling on accuracy of imbalanced classification. In: *8th International Conference on Machine Vision*. Vol. 9875: Barcelona, Spain, 423–427.
- Busa-Fekete, R. and Dembczynski, K. 2015. Online F-measure optimisation. *Neural Information Processing Systems*, 28:1–9.
- Bussmann, N., Giudici, P., Marinelli, D. and Papenbrock, J. 2020. Explainable machine learning in credit risk management. *Computational Economics*, 57(1):203–216.
- Cacheda, F., Fernandez, D., Novoa, F.J. and Carneiro, V. 2019. Early detection of depression: social network analysis and random forest techniques. *Journal of Medical Internet Research*, 21(6), Article e12554.
- Caelen, O. 2017. A Bayesian interpretation of the confusion matrix. *Annals of Mathematics and Artificial Intelligence*, 81(3):429–450.
- Campos, G.O., Zimek, A., Sander, J., Campello, R.J.G.B., Micenkova, B., Schubert, E., Assent, I. and Houle, M.E. 2016. On the evaluation of unsupervised outlier detection: measures, datasets, and an empirical study. *Data Mining and Knowledge Discovery*, 30:891–927.
- Can, U. and Alatas, B. 2019. A new direction in social network analysis: online social network analysis problems and applications. *Physica A: Statistical Mechanics and its Applications*, 535, Article 122372.
- Cao, V.L., Nicolau, M. and Mcdermott, J. 2016. A hybrid autoencoder and density estimation model for anomaly detection. *Parallel Problem Solving from Nature*, 9921:717–726.
- Carcillo, F., LeBorgne, Y., Caelen, O., Kessaci, Y., Oble, F. and Bontempi, G. 2019. Combining unsupervised and supervised learning in credit card fraud detection. *Information Sciences*, 557:317–331.
- Carneiro, N., Figueira, G. and Costa, M. 2017. A data mining-based system for credit card fraud detection in e-tail. *Decision Support Systems*, 95:91–101.
- Chai, T. and Draxler, R.R. 2014. Root mean square error (RMSE) or mean absolute error (MAE)? – arguments against avoiding RMSE in the literature. *Geoscientific Model Development*, 7(3):1247–1250.
- Chalapathy, R. and Chawla, S. 2019. Deep learning for anomaly detection: a survey. *arXiv preprint arXiv:1901.03407*.
- Chandola, V., Banerjee, A. and Kumar, V. 2009. Anomaly detection: a survey. *ACM Computing Surveys*, 41(3):1–58.
- Chang, C.H. 2020. Managing credit card fraud risk by autoencoders. In: *International Conference on Pervasive Artificial Intelligence*. Calabria, Italy, 118–122.
- Chang, Y., Tu, Z., Xie, W. and Yuan, J. 2020. Clustering-driven deep autoencoder for video anomaly detection. In: *European Conference on Computer Vision*. Glasgow, United Kingdom, 329–345.

- Chawla, A., Jacob, P., Lee, B. and Fallon, S. 2019. Bidirectional LSTM autoencoder for sequence based anomaly detection in cyber security. *International Journal of Simulation - Systems, Science & Technology*.
- Chawla, N.V. 2009. *Data mining for imbalanced datasets: an overview*, *Data Mining and Knowledge Discovery Handbook*. New York: Springer 875–886.
- Chawla, N.V., Bowyer, K.W., Hall, L.O. and Kegelmeyer, W.P. 2002. SMOTE: Synthetic Minority Oversampling Technique. *Journal of Artificial Intelligence Research (JAIR)*, 16:321–357.
- Chen, C.Y. and Huang, J.J. 2020. A novel centrality for finding key persons in a social network by the bi-directional influence map. *Symmetry*, 12(10), Article 1747.
- Chen, J., Sathe, S., Aggarwal, C. and Turaga, D. 2017. Outlier detection with autoencoder ensembles. In: *Proceedings of the SIAM International Conference on Data Mining*. Houston, Texas, USA, 90–98.
- Chen, J.I. and Lai, K. 2021. Deep convolution neural network model for credit card fraud detection and alert. *Journal of Artificial Intelligence and Capsule Networks*, 3(2):101–112.
- Chen, N., Chen, L., Ma, Y. and Chen, A. 2019. Regional disaster risk assessment of China based on self-organizing map: clustering, visualization and ranking. *International Journal of Disaster Risk Reduction*, 33:196–206.
- Chen, T., Tang, L., Sun, Y., Chen, Z. and Zhang, K. 2016. Entity embedding-based anomaly detection for heterogeneous categorical events. In: *Proceedings of the 25th International Joint Conference on Artificial Intelligence*. New York, USA, 1396–1403.
- Chen, Y., Ashizawa, N., Yean, S., Yeo, C.K. and Yanai, N. 2020. Self-organizing map-assisted deep autoencoding Gaussian mixture model for intrusion detection. *arXiv preprint arXiv:2008.12686*.
- Chen, Y., Ashizawa, N., Yeo, C.K., Yanai, N. and Yean, S. 2021. Multi-scale self-organising map assisted deep autoencoding Gaussian mixture model for unsupervised intrusion detection. *Knowledge-Based Systems*, 224, Article 107086.
- Cheng, Z., Zou, C. and Dong, J. 2019. Outlier detection using isolation forest and local outlier factor. In: *Proceedings of the Conference on Research in Adaptive and Convergent Systems*. Chongqing, China, 161–168.
- Chicco, D. and Jurman, G. 2020. The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation. *BMC Genomics*, 21(1):1–13.
- Chicco, D., Sadowski, P. and Baldi, P. 2014. Deep autoencoder neural networks for gene ontology annotation predictions. In: *Proceedings of the 5th ACM Conference on Bioinformatics, Computational Biology, and Health Informatics*. Newport Beach, California, USA, 533–540.

- Choi, H., Kim, M., Lee, G. and Kim, W. 2019. Unsupervised learning approach for network intrusion detection system using autoencoders. *Journal of Supercomputing*, 75(9):5597–5621.
- Chollet, F. n.d. *Keras*. <https://keras.io>. Accessed: 2021-06-16.
- Chow, J.K., Su, Z., Wu, J., Tan, P.S., Mao, X. and Wang, Y.H. 2020. Anomaly detection of defects on concrete structures with the convolutional autoencoder. *Advanced Engineering Informatics*, 45, Article 101105.
- Claeskens, G. and Hjort, N.L. 2008. *Model selection and model averaging*. Cambridge, UK: Cambridge Books.
- Colladon, A.F. and Naldi, M. 2020. Distinctiveness centrality in social networks. *Plos one*, 15(5), Article e0233276.
- Colladon, A.F. and Remondi, E. 2017. Using social network analysis to prevent money laundering. *Expert Systems with Applications*, 67:49–58.
- Collobert, R. and Bengio, S. 2004. Links between perceptrons, MLPs and SVMs. In: *Proceeding of the 21st International Conference on Machine Learning*. Banff, Alberta, Canada.
- Cottrell, M., Olteanu, M., Rossi, F. and Villa-Vialaneix, N.N. 2018. Self-organising maps: theory and applications. *Revista de Investigacion Operacional*, 39(1):1–22.
- Cynthia, P. Caroline and George, S. Thomas 2021. An outlier detection approach to credit card fraud detection using machine learning: a comparative analysis on supervised and unsupervised learning. *Intelligence in Big Data Technologies - Beyond the Hype, Advances in Intelligent Systems and Computing*, 1167:125–135.
- Dal Pozzolo, A. 2015. *Adaptive machine learning for credit card fraud detection*. Brussels, Belgium: Université Libre De Bruxelles (Thesis - PhD).
- Dal Pozzolo, A., Boracchi, G., Caelen, O., Alippi, C. and Bontempi, G. 2015. Credit card fraud detection and concept-drift adaption with delayed supervised information. In: *International Joint Conference on Neural Networks (IJCNN)*. Killarney, Ireland, 1–8.
- Das, K., Mahapatra, R., Samanta, S. and Pal, A. 2020. *Influential nodes in social networks: centrality measures*. Hershey, PA: IGI Global, 371–385.
- Dau, A., Ciesielski, V. and Song, A. 2014. Anomaly detection using replicator neural networks trained on examples of one class. In: *Asia-Pacific Conference on Simulated Evolution and Learning*. Dunedin, New Zealand, 311–322.
- Delamaire, L., Abdou, H. and Pointon, J. 2009. Credit card fraud and detection techniques: a review. *Banks and Bank Systems*, 4(2):57–68.
- Dias, A., Ruthes, S., Lima, L., Campra, E., Silva, M., Sousa, M.B. de and Porto, G. 2020. Network centrality analysis in management and accounting sciences. *RAUSP Management Journal*, 55:207–226.
- Dijkstra, E.W. 1959. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271.

- Domingos, P. and Hulten, G. 2000. Mining high-speed data streams. In: *Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Boston, Massachusetts, USA, 71–80.
- Drummond, C. and Holte, R.C. 2003. C4.5, class imbalance, and cost sensitivity: why under-sampling beats over-sampling. *Workshop on Learning from Imbalanced Datasets II*, 11:1–8.
- Duman, E. and Ozcelik, M.H. 2011. Detecting credit card fraud by genetic algorithm and scatter search. *Expert Systems with Applications*, 38(10):13057–13063.
- Elkan, C. 2001. The foundations of cost-sensitive learning. In: *International Joint Conference on Artificial Intelligence*. Vol. 17(1): Seattle, USA, 973–978.
- Elsayed, M.S., Le-Khac, N.A., Dev, S. and Jurcut, A.D. 2020. Network anomaly detection using LSTM based autoencoder. In: *Proceedings of the 16th ACM Symposium on QoS and Security for Wireless and Mobile Networks*. Alicante, Spain, 37–45.
- Elsken, T. 2019. Neural architecture search: a survey. *Journal of Machine Learning Research*, 20(1):1–21.
- Engels, C., Kumar, K. and Philip, D. 2019. Financial literacy and fraud detection. *The European Journal of Finance*, 4(5):420–442.
- Eppstein, D., Loffler, M. and Strash, D. 2013. Listing all maximal cliques in large sparse real-world graphs in near-optimal time. *Journal of Experimental Algorithmics*, 18(3):1–21.
- Erfani, S.M., Rajasegarar, S., Karunasekera, S. and Leckie, C. 2016. High-dimensional and large-scale anomaly detection using a linear one-class SVM with deep learning. *Pattern Recognition*, 58:121–134.
- Erwin, E., Obermayer, K. and Schulten, K. 1992. Self-organising maps: ordering, convergence properties and energy functions. *Biological Cybernetics*, 67(1):47–55.
- Fakhraei, S., Foulds, J., Shashanka, M. and Getoor, L. 2015. Collective spammer detection in evolving multi-relational social networks. In: *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Sydney, Australia, 1769–1778.
- Fan, C., Xiao, F., Zhao, Y. and Wang, J. 2018. Analytical investigation of autoencoder-based methods for unsupervised anomaly detection in building energy data. *Applied Energy*, 211:1123–1135.
- Fan, G. and Zhu, M. 2011. Detection of rare items with target. *Statistics and Its Interface*, 4(1):11–17.
- Fan, W., Stolfo, S.J., Zhang, J. and Chan, P.K. 1999. Adacost: misclassification cost-sensitive boosting. In: *16th International Conference on Machine Learning*. Bled, Slovenia, 97–105.
- Feng, S., Keung, J., Yu, X., Xiao, Y., Bennin, K.E., Kabir, M.A. and Zhang, M. 2021. COSTE: complexity-based oversampling technique to alleviate the class

- imbalance problem in software defect prediction. *Information and Software Technology*, 129:1–16.
- Ferles, C., Papanikolaou, Y. and Naidoo, K.J. 2018. Denoising autoencoder self-organizing map (DASOM). *Neural Networks*, 105:112–131.
- Feurer, M. and Hutter, F. 2019. Hyperparameter optimization. *Automated Machine Learning*, 3–34.
- Fire, M. and Puzis, R. 2016. Organisation mining using online social networks. *Networks and Spatial Economics*, 16(2):545–578.
- Flach, P.A. 2019. Performance evaluation in machine learning: the good, the bad, the ugly, and the way forward. In: *Proceedings of the 33rd AAAI Conference on Artificial Intelligence*, 9808–9814.
- Foutch, D., Pham, B. and Shen, T. 2021. Protein conformational switch discerned via network centrality properties. *Computational and Structural Biotechnology Journal*, 19:3599–3608.
- Gama, J., Fernandes, R. and Rocha, R. 2006. Decision trees for mining data streams. *Intelligent Data Analysis*, 10(1):23–45.
- Gao, J., Fan, W., Han, J. and Philip, S.Y. 2007. A general framework for mining concept-drift data streams with skewed distributions. In: *Proceedings of the 2007 SIAM International Conference on Data Mining (SDM)*. Minneapolis, Minnesota, USA, 3–14.
- Gardner, W., Maliki, R., Cutts, S.M., Muir, B.W., Ballabio, D., Winkler, D.A. and Pigram, P.J. 2020. Self-organizing map and relational perspective mapping for the accurate visualization of high-dimensional hyperspectral data. *Analytical Chemistry*, 92(15):10450–10459.
- Gavrikov, P. 2020. *visualkeras*. <https://github.com/paulgavrikov/visualkeras>. Accessed: 2021-06-05.
- Glorot, X., Bordes, A. and Bengio, Y. 2011. Deep sparse rectifier neural networks. In: *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics*. Ft. Lauderdale, Florida, USA, 315–323.
- Gogoi, P., Borah, B. and Bhattacharyya, D.K. 2010. Anomaly detection analysis of intrusion data using supervised and unsupervised approach. *Journal of Convergence Information Technology*, 5(1):95–110.
- Goh, K.I., Kahng, B. and Kim, D. 2001. Universal behaviour of load distribution in scale-free networks. *Physical Review Letters*, 87(27):1–4.
- Gomes, C., Jin, Z. and Yang, H. 2021. Insurance fraud detection with unsupervised deep learning. *Journal of Risk and Insurance*, 88(3):591–624.
- Goodfellow, I., Bengio, Y. and Courville, A. 2016. *Deep Learning*. <http://www.deeplearningbook.org>. Accessed: 2020-11-21.

- Gorgoglione, A., Castro, A., Gioia, A. and Iacobellis, V. 2020. Application of the self-organizing map (SOM) to characterize nutrient urban runoff. In: *International Conference on Computational Science and Its Applications*, 680–692.
- Goz, F., Mutlu, A., Kucuk, K., Temur, M. and Gun, A. 2021. Effect of centrality measures for keyword extraction from Turkish documents. In: *29th Signal Processing and Communications Applications Conference (SIU)*. Istanbul, Turkey, 1–4.
- Grando, F., Granville, L.Z. and Lamb, L.C. 2018. Machine learning in network centrality measures: tutorial and outlook. *ACM Computing Surveys*, 51(5):1–32.
- Gryzb, J., Klikowski, J. and Wozniak, M. 2021. Hellinger distance weighted ensemble for imbalanced data stream classification. *Journal of Computational Science*, 51:1–43.
- Hagberg, A.A., Schult, D.A. and Swart, P.J. 2008. Exploring network structure, dynamics, and function using NetworkX. In: *Proceedings of the 7th Python in Science Conference*. Pasadena, California, USA.
- Hakak, S., Khan, W.Z., Irfan, M., Choo, K.R. and Shoaib, M. 2020. Have you been a victim of Covid-19-related cyber incidents? Survey, taxonomy, and mitigation strategies. *IEEE Access*, 8:124134–124144.
- Hamilton, K., Mintz, T., Date, P. and Schuman, C.D. 2020. Spike-based graph centrality measures. In: *International Conference on Neuromorphic Systems*. Oak Ridge, TN, USA, 1–8.
- Hamilton, W.L. 2020. Graph representation learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 14(3):1–159.
- Hancock, J. and Khoshgoftaar, T.M. 2021. Performance of catboost and xgboost in medicare fraud detection. In: *19th IEEE International Conference on Machine Learning and Applications (ICMLA)*. Miami, Florida, USA, 572–579.
- Hand, D.J. 2009. Measuring classifier performance: a coherent alternative to the area under the ROC curve. *Machine Learning*, 77(1):103–123.
- Hand, D.J. and Christen, P. 2018. A note on using the F-measure for evaluating record linkage algorithms. *Statistics and Computing*, 28(3):539–547.
- Harwani, H., Jain, J., Jadhav, C. and Hodavdekar, M. 2020. Credit card fraud detection technique using hybrid approach: an amalgamation of self organising maps and neural networks. *International Research Journal of Engineering and Technology (IRJET)*, 7(7):5071–5075.
- Haykin, S. 2009. *Neural networks and learning machines*. Vol. 3. London, United Kingdom: Pearson Education.
- He, H., Bai, Y., Garcia, E.A. and Li, S. 2008. ADASYN: Adaptive synthetic sampling approach for imbalanced learning. In: *International Joint Conference on Neural Networks*. Hong Kong, China, 1322–1328.
- He, H. and Ma, Y. 2013. *Imbalanced learning: foundations, algorithms, and applications*. Hoboken, NJ: Wiley-IEEE Press.

- He, K., Lu, Y. and Sclaroff, S. 2018. Local descriptors optimized for average precision. In: *Conference on Computer Vision and Pattern Recognition*. Salt Lake City, UT, USA, 596–605.
- Hinton, G.E. and Salakhutdinov, R.R. 2006. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507.
- Hoens, T.R., Polikar, R. and Chawla, N.V. 2012. Learning from streaming data with concept drift and imbalance: an overview. *Progress in Artificial Intelligence*, 1(1):89–101.
- Huang, D., Mu, D., Yang, L. and Cai, X. 2018. CoDetect: financial fraud detection with anomaly feature detection. *IEEE Access*, 6:19161–19174.
- Hulten, G., Spencer, L. and Domingos, P. 2001. Mining time-changing data streams. In: *Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. San Francisco, CA, USA, 97–106.
- Hung, M., Lauren, E., Hon, E.S., Birmingham, W.C., Xu, J., Su, S., Hon, S.D., Park, J., Dang, P. and Lipsky, M.S. 2020. Social network analysis of COVID-19 sentiments: application of artificial intelligence. *Journal of Medical Internet Research*, 22(8), Article e22590.
- Jamali, S. and Mitchell, D. 2018. Centrality-based improvements to CDCL heuristics. In: *The 21st International Conference on Theory and Applications of Satisfiability Testing*. Oxford, United Kingdom, 122–131.
- James, G., Witten, D., Hastie, T. and Tibshirani, R. 2013. *An introduction to statistical learning with applications in R*. New York, NY: Springer.
- Janzing, D., Minorics, L. and Bloebaum, P. 2020. Feature relevance quantification in explainable AI: a causal problem. In: *International Conference on Artificial Intelligence and Statistics*, 2907–2916.
- Jeyasudha, J. and Usha, G. 2021. An intelligent centrality measures for influential node detection in COVID-19 environment. *Wireless Personal Communications*, 1–27.
- Jiang, B. and Harrie, L. 2004. Selection of streets from a network using self-organising maps. *Transactions in GIS*, 8(3):335–350.
- Jiang, X., Gao, J., Hong, X. and Cai, Z. 2014. Gaussian processes autoencoder for dimensionality reduction. In: *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Tainan, Taiwan, 62–73.
- Jin, X., Que, Z., Sun, Y., Guo, Y. and Qiao, W. 2019. A data-driven approach for bearing fault prognostics. *IEEE Transactions on Industry Applications*, 55(4):3394–3401.
- Johnson, J.M. and Khoshgoftaar, T.M. 2019. Survey on deep learning with class imbalance. *Journal of Big Data*, 6(27):59–69.
- Kaggle 2017. *Credit card fraud detection*. www.kaggle.com/mlg-ulb/creditcardfraud. Accessed: 2020-01-05.

- Kamath, C. and Fan, Y.J. 2018. Regression with small data sets: a case study using code surrogates in additive manufacturing. *Knowledge and Information Systems*, 57(2):475–493.
- Kang, S., Lee, K. and Shin, K. 2022. Are edge weights in summary graphs useful? A comparative study. In: *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Chengdu, China, 54–67.
- Kaufman, S., Rosset, S. and Perlich, C. 2011. Leakage in data mining: formulation, detection, and avoidance. *ACM Transactions on Knowledge Discovery and Data Mining*, 6(4):556–563.
- Kaur, R. and Singh, S. 2016. A survey of data mining and social network analysis based anomaly detection techniques. *Egyptian Informatics Journal*, 17(2):199–216.
- Khan, A.P., Mahajan, V.S., Shaikh, S.H. and Koli, A.B. 2013. Credit card fraud detection system through observation probability using hidden markov model. *International Journal of Thesis Projects and Dissertations*, 1:7–16.
- Khan, S. and Taati, B. 2017. Detecting unseen falls from wearable devices using channel-wise ensemble of autoencoders. *Expert Systems with Applications*, 87:280–290.
- Khare, N. and Sait, S.Y. 2018. Credit card fraud detection using machine learning models and collating machine learning models. *International Journal of Pure and Applied Mathematics*, 118(20):825–838.
- Kim, S., Jo, W.Y. and Shon, T. 2020. APAD: autoencoder-based payload anomaly detection for industrial IoE. *Applied Soft Computing*, 88, Article 106017.
- Kim, Y. and Sohn, S.Y. 2012. Stock fraud detection using peer group analysis. *Expert Systems with Applications*, 39(10):8986–8992.
- Kingma, D.P. and Ba, J. 2015. Adam: a method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.
- Kohonen, T. 2001. *Self-organizing maps*. Vol. 3. Berlin: Springer.
- Kohonen, T. 2013. Essentials of the self-organising map. *Neural Networks*, 37:52–65.
- Kohonen, T. 2014. *MATLAB implementations and applications of the self-organizing map*. Helsinki, Finland: Unigrafia Oy.
- Kou, Y., Lu, C., Sirwongwattana, S. and Huang, Y. 2011. Survey of fraud detection techniques. In: *IEEE International Conference on Networking, Sensing and Control*. Vol. 2: Delft, The Netherlands, 749–754.
- Kovach, S. and Ruggiero, W.V. 2011. Online banking fraud detection based on local and global behavior. In: *5th International Conference on Digital Society*. Gosier, Guadeloupe, France, 166–171.
- Kriegel, H., Kroger, P., Schubert, E. and Zimek, A. 2011. Interpreting and unifying outlier scores. In: *11th SIAM International Conference on Data Mining (SDM)*. Mesa, AZ, USA, 13–24.

- Krivko, M. 2010. A hybrid model for plastic credit card fraud detection systems. *Expert Systems with Applications*, 37(8):6070–6076.
- Lata, L.N., Koushika, I.A. and Hasan, S.S. 2015. A comprehensive survey of fraud detection techniques. *International Journal of Applied Information Systems*, 10(2):26–32.
- Lavin, A. and Ahmad, S. 2015. Evaluating real-time anomaly detection algorithms - the Numenta anomaly benchmark. In: *IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*. Miami, FL, USA, 38–44.
- Leevy, J.L., Khoshgoftaar, T.M., Bauder, R.A. and Seliya, N. 2018. A survey on addressing high-class imbalance in big data. *Journal of Big Data*, 5(1):1–30.
- Leite, R.A., Gschwandtner, T., Miksch, S., Gstrein, E. and Kuntner, J. 2018. Visual analytics for event detection: focusing on fraud. *Visual Informatics*, 2(4):198–212.
- Li, Z., Kamnitsas, K. and Glocker, B. 2021. Analyzing overfitting under class imbalance in neural networks for image segmentation. *IEEE Transactions on Medical Imaging*, 40(3):1065–1077.
- Liashchynskyi, P. and Liashchynskyi, P. 2019. Grid search, random search, genetic algorithms: a big comparison for NAS. *arXiv preprint arXiv:1912.06059*.
- Lichtenwalter, R.N. and Chawla, N.V. 2009. Adaptive methods for classification in arbitrarily imbalanced and drifting data streams. In: *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Bangkok, Thailand, 53–75.
- Lind, P.D., Gonzalez, M.C. and Herrmann, H.J. 2005. Cycles and clustering in bipartite networks. *Physical Review E*, 72(5), Article 056127.
- Lipton, Z.C. 2018. The mythos of model interpretability: in machine learning, the concept of interpretability is both important and slippery. *Queue*, 16(3):31–57.
- Litan, A. 2012. Who’s who and what’s what in the enterprise fraud and misuse management market. *Gartner Research*.
- Liu, A., Lu, J. and Zhang, G. 2021a. Diverse instance-weighting ensemble based on region drift disagreement for concept drift adaption. *IEEE Transactions on Neural Networks and Learning Systems*, 32(1):293–307.
- Liu, A., Zhang, G. and Lu, J. 2017. Fuzzy time windowing for gradual concept drift adaption. In: *IEEE International Conference on Fuzzy Systems*. Naples, Italy, 1–6.
- Liu, J., Song, K., Feng, M., Yan, Y., Tu, Z. and Zhu, L. 2021b. Semi-supervised anomaly detection with dual prototypes autoencoder for industrial surface inspection. *Optics and Lasers in Engineering*, 136, Article 106324.
- Liu, Y., Cheng, J., Yan, C., Wu, X. and Chen, F. 2015. Research on the matthews correlation coefficients metrics of personalized recommendation algorithm evaluation. *International Journal of Hybrid Information Technology*, 8(1):163–172.

- Liu, Y., Sun, Y., Xue, B., Zhang, M., Yen, G.G. and Tan, K.C. 2021c. A survey on evolutionary neural architecture search. *IEEE Transactions on Neural Networks and Learning Systems*, 1:1–21.
- Lizardo, O. 2021. SOCIOL 111 Social Networks lecture notes. *Los Angeles, CA: UCLA*.
- Llantos, O.E., Nicolau, M. and Estuar, M.R.J.E. 2019. Characterizing instructional leader interactions in a social learning management system using social network analysis. In: *10th International Conference on Emerging Ubiquitous Systems and Pervasive Networks*. Vol. 160: Coimbra, Portugal, 149–156.
- Lokesh, S., Kumar, P.M., Devi, M.R., Parthasarathy, P. and Gokulnath, C. 2019. An automatic Tamil speech recognition system by using bidirectional recurrent neural network with self-organizing map. *Neural Computing and Applications*, 31(5):1521–1531.
- Lopez, A.U., Mateo, F., Navio-Marco, J., Martinez, J.M., Gomez-Sanchis, J., Vila-Frances, J. and Serrano-Lopez, A.J. 2019. Analysis of computer user behaviour, security incidents and fraud using self-organizing maps. *Computers and Security*, 83:38–51.
- Lu, W. and Yan, X. 2020. Deep fisher autoencoder combined with self-organizing map for visual industrial process monitoring. *Journal of Manufacturing Systems*, 56:241–251.
- Lundberg, S.M. and Lee, S. 2017. A unified approach to interpreting model predictions. *Advances in Neural Information Processing Systems*, 30:4765–4774.
- Lutz, M.A., Vogt, S., Berkhout, V., Faulstich, S., Dienst, S., Steinmetz, U., Guck, C. and Ortega, A. 2020. Evaluation of anomaly detection of an autoencoder based on maintenance information and scada-data. *Maintenance Management of Wind Turbines*, 13(5), Article 1063.
- Maaten, L. Van der and Hinton, G. 2008. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9(11):2579–2605.
- Makhzani, A. and Frey, B. 2013. k-Sparse autoencoders. *arXiv preprint arXiv:1312.5663*.
- Mancini, R., Ritacco, A., Lanciano, G. and Cucinotta, T. 2020. XPySom: high-performance self-organizing maps. In: *Proceedings of IEEE 32nd International Symposium on Computer Architecture and High Performance Computing*. Porto, Portugal, 209–216.
- Manning, C.D., Raghavan, P. and Schütze, H. 2008. *Introduction to information retrieval*. New York, NY: Cambridge University Press.
- Marchiori, M. and Latora, V. 2000. Harmony in the small-world. *Physica A: Statistical Mechanics and its Applications*, 285(3):539–546.

- Matthews, A.G.D., Hron, J., Rowland, M., Turner, R.E. and Ghahramani, Z. 2018. Gaussian process behaviour in wide deep neural networks. *arXiv preprint arXiv:1804.11271*.
- Maurya, S.K., Liu, X. and Murata, T. 2019. Fast approximations of betweenness centrality with graph neural networks. In: *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. Beijing, China, 2149–2152.
- Mazin, A., Hawkins, S.H., Springfield, O., Dhillon, J., Manley, B.J., Jeong, D.K. and Raghunand, N. 2021. Identification of sarcomatoid differentiation in renal cell carcinoma by machine learning on multiparametric MRI. *Scientific Reports*, 11(1):1–12.
- McInnes, L. and Healy, J. 2018. UMAP: uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*.
- McKinney, W. 2010. Data structures for statistical computing in Python. In: *Proceedings of the 9th Python in Science Conference*. Austin, TX, USA, 56–61.
- Mekterovic, I., Karan, M., Pintar, D. and Brkic, L. 2021. Credit card fraud detection in card-not-present transactions: where to invest? *Applied Sciences*, 11(15), Article 6766.
- Mermillod, M., Bugaiska, A. and Bonin, P. 2013. The stability-plasticity dilemma: investigating the continuum from catastrophic forgetting to age-limited learning effects. *Frontiers in Psychology*, 4(504).
- Merrill, N. and Eskandarian, A. 2020. Modified autoencoder training and scoring for robust unsupervised anomaly detection in deep learning. *IEEE Access*, 8:101824–101833.
- Misbahulmunir, S., Ramachandaramurthy, V.K. and Thayoob, Y.H. 2020. Improved self-organizing map clustering of power transformer dissolved gas analysis using inputs pre-processing. *IEEE Access*, 8:71798–71811.
- Mohapatra, P., Jawahar, C.V. and Kumar, M.P. 2014. Efficient optimization for average precision SVM. *Advances in Neural Information Processing Systems*, 271–9.
- Mohri, M., Rostamizadeh, A. and Talwalkar, A. 2012. *Foundations of machine learning*. Cambridge, MA: The MIT Press.
- Molnar, C. 2019. *Interpretable machine learning*. <https://christophm.github.io/interpretable-ml-book/>. Accessed: 2021-08-17.
- Muhammad, G., Hossain, M.S. and Garg, S. 2020. Stacked autoencoder-based intrusion detection system to combat financial fraudulent. *IEEE Internet of Things Journal*, 1–1.
- Nair, V. and Hinton, G.E. 2010. Rectified linear units improve restricted Boltzmann machines. In: *Proceedings of the 27th International Conference on Machine Learning*. Haifa, Israel.

- Napitupulu, H., Carnia, E., Anggriani, N. and Supriatna, A.K. 2021. Centrality measures in transportation networks for Unpad campus route. *Journal of Physics: Conference Series*, 1722(1):1–7.
- Natita, W., Wiboonsak, W. and Dusadee, S. 2016. Appropriate learning rate and neighbourhood function of self-organising map for specific humidity pattern classification over southern Thailand. *International Journal of Modelling and Optimization*, 6(1):61–65.
- Needham, M. and Hodler, A.E. 2021. *Graph algorithms*. Sebastopol, CA: O’Reilly.
- Negro, A. 2021. *Graph-powered machine learning*. Shelter Island, NY: Manning.
- Nejem, S.M. and Kadeem, S.M. 2021. A survey on fraud detection techniques in e-commerce. *Tech-Knowledge*, 1(1):33–47.
- Newman, M.E.J. 2010. *Networks: an introduction*. New York, NY: Oxford University Press.
- Ng, A. 2011. Sparse autoencoder. *CS294A lecture notes 72, Stanford University*, 1–19.
- Nilashi, M., Ahmadi, H., Manaf, A.A., Rashid, T.A., Samad, S., Shahmoradi, L., Aljojo, N. and Akbari, E. 2020. Coronary heart disease diagnosis through self-organizing map and fuzzy support vector machine with incremental updates. *International Journal of Fuzzy Systems*, 22(4):1376–1388.
- Nilson 2020. *Card fraud losses reach \$28.65 billion*. <https://nilsonreport.com/mention/1313/1link/>. Accessed: 2022-02-21.
- Nolle, T., Luetzgen, S., Seeliger, A. and Muhlhauser, M. 2018. Analyzing business process anomalies using autoencoders. *Machine Learning*, 107(11):1875–1893.
- Notsu, A., Yasuda, K., Unukata, S. and Honda, K. 2020. Online state space generation by a growing self-organizing map and differential learning for reinforcement learning. *Applied Soft Computing*, 97, Article 106723.
- Nurek, M. and Michalski, R. 2020. Combining machine learning and social network analysis to reveal the organisational structures. *Applied Sciences*, 10(5):1–43.
- Oh, S.J., Schiele, B. and Fritz, M. 2019. Towards reverse-engineering black-box neural networks. *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*, 121–144.
- Oliveira, J.C., Zorita, E., Koul, V., Ludwig, T. and Baehr, J. 2020. Forecast opportunities for European summer climate ensemble predictions using self-organising maps. In: *Proceedings of the 10th International Conference on Climate Informatics*. New York, NY, USA, 67–71.
- Olszewski, D. 2014. Fraud detection using self-organizing map visualizing the user profiles. *Knowledge-based Systems*, 70:324–334.
- O’Malley, A.J. and Onnela, J. 2014. Topics in social network analysis and network science. *arXiv preprint arXiv:1404.0067*.

- Onnela, J., Saramaki, J., Kertesz, J. and Kaski, K. 2005. Intensity and coherence of motifs in weighted complex networks. *Physical Review E*, 71(6):1–5.
- Page, L., Brin, S., Motwani, R. and Winograd, T. 1999. The PageRank citation ranking: bringing order to the web. *Stanford InfoLab*, 1–17.
- Pang, G. 2019. Non-IID outlier detection with coupled outlier factors. *Sydney, Australia: University of Technology Sydney (Thesis - PhD)*.
- Pang, G., Cao, L., Chen, L. and Liu, H. 2016. Unsupervised feature selection for outlier detection by modelling hierarchical value-feature couplings. In: *IEEE 16th International Conference on Data Mining (ICDM)*. Barcelona, Spain, 410–419.
- Pang, G., Shen, C., Cao, L. and Hengel, A. Van den 2021. Deep learning for anomaly detection: a review. *ACM Computing Surveys*, 54(2):1–38.
- Parsons, V.L. 2017. *Stratified sampling*. <https://onlinelibrary.wiley.com/doi/10.1002/9781118445112.stat05999.pub2>. Accessed: 2020-12-03.
- Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M. and Duchesnay, E. 2011. Scikit-learn: machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.
- Perez, J.L.R., Ribeiro, B. and Perez, C.M. 2016. Mahalanobis distance metric learning algorithm for instance-based data stream classification. In: *International Joint Conference on Neural Networks*. Vancouver, BC, Canada, 1857–1862.
- Phua, C., Lee, V., Smith, K. and Gayler, R. 2010. A comprehensive survey of data mining-based fraud detection research. *arXiv preprint arXiv:1009.6119*.
- Phuong, V.T.H., Czygan, M. and Kumar, A. 2017. *Python: data analytics and visualization*. Birmingham, United Kingdom: Packt Publishing.
- Piraveenan, M., Prokopenko, M. and Zomaya, A.Y. 2008. Local assortativeness in scale-free networks. *Europhysics Letters*, 84(2), Article 28002.
- Polzlbauer, G. and Rauber, A. 2004. Survey and comparison of quality measures for self-organizing maps. In: *Proceedings of the Fifth Workshop on Data Analysis*. Vysoké Tatry, Slovakia, 67–82.
- Ponmalai, R. and Kamath, C. 2019. Self-organizing maps and their applications to data analysis. *Lawrence Livermore National Laboratory*, 1–51.
- Popereshnyaki, S. and Yurchuk, I. 2021. Social networks: analysis, algorithms and their implementation. In: *5th International Conference on Computational Linguistics and Intelligent Systems*. Lviv, Ukraine, 811–821.
- Pourhabibi, T., Ong, K., Kam, B.H. and Boo, Y.L. 2020. Fraud detection: a systematic literature review of graph-based anomaly detection approaches. *Decision Support Systems*, 133, Article 113303.
- Powers, D.M.W. 2007. Evaluation: From precision, recall and F-factor to ROC, informedness, markedness and correlation. *arXiv preprint arXiv:2010.16061*.

- Powers, D.M.W. 2014. What the F-measure doesn't measure: features, flaws, fallacies and fixes. *arXiv preprint arXiv:1503.06410*.
- PricewaterhouseCoopers 2022. Global Economic Crime and Fraud Survey. Accessed: 2022-01-07.
- Provost, F. 2000. Machine learning from imbalanced data sets 101. In: *Proceedings of the AAAI'2000 Workshop on Imbalanced Data Sets*. Vol. 68(2000): Austin, TX, USA, 1–3.
- Provost, F., Fawcett, T. and Kohavi, R. 1998. The case against accuracy estimation for comparing induction algorithms. In: *Proceedings of the 15th International Conference on Machine Learning*. Vol. 98: San Francisco, CA, USA, 445–453.
- Purwanto, Y., Kuspriyanto, K., Hendrawan, H. and Rahardjo, B. 2018. Minimal triangle area Mahalanobis distance for stream homogeneous group-based DDoS classification. *International Journal on Electrical Engineering and Informatics*, 10(2):369–383.
- Qi, J., Ma, G., Navarro-Alarcon, D., Zhang, H. and Lyu, Y. 2022. Towards latent space based manipulation of elastic rods using autoencoder models and robust centerline extractions. *Advanced Robotics*, 36(3):101–115.
- Qian, J., Nguyen, N.P., Oya, Y., Kikugawa, G., Okabe, T., Huang, Y. and Ohuchi, F.S. 2019. Introducing self-organized maps (SOM) as a visualization tool for materials research and education. *Results in Materials*, 4, Article 100020.
- Raghavan, P. and El Gayar, N. 2019. Fraud detection using machine learning and deep learning. In: *International Conference on Computational Intelligence and Knowledge Economy (ICCIKE)*. Dubai, United Arab Emirates, 334–339.
- Ramos, A.D., Lopez-Rubio, E. and Palomo, E.J. 2019. The forbidden region self-organizing map neural network. *IEEE Transactions on Neural Networks and Learning Systems*, 31(1):201–211.
- Revaud, J., Almazan, J., Rezende, R.S and Sousa, C.R. De 2019. Learning with average precision: training image retrieval with a listwise loss. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. Seoul, South Korea, 5107–5116.
- Ribeiro, M., Lazzaretti, A.E. and Lopes, H.S. 2018. A study of deep convolutional autoencoders for anomaly detection in videos. *Pattern Recognition Letter*, 105:13–22.
- Ribeiro, M.T., Singh, S. and Guestrin, C. 2016. Why should I trust you: explaining the predictions of any classifier. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. San Francisco, CA, USA, 1135–1144.
- Rifai, S., Vincent, P., Muller, X., Glorot, X. and Bengio, Y. 2011. Contractive autoencoders: explicit invariance during feature extraction. In: *Proceedings of the*

- 28th International Conference on Machine Learning. Bellevue, Washington, USA, 833–840.
- Riveiro, M., Pallotta, G. and Vespe, M. 2018. Maritime anomaly detection: a review. *Wiley Interdisciplinary Reviews. Data Mining and Knowledge Discovery*, 8(5):1–34.
- Robertson, D.J. 2018. Face recognition: security contexts, super-recognisers, and sophisticated fraud. *The Journal of The United States Homeland Defence and Security Information Analysis Center (HDIAC)*, 5(1):6–10.
- Rocchini, C. 2006. *Centrality*. <https://en.wikipedia.org/wiki/User:Rocchini>. Accessed: 2021-03-14.
- Rodrigues, L. and Omar, N. 2014. Auto claim fraud detection using multi-classifier system. *Journal of Computer Science and Information Technology*, 14.
- Sadgali, I., Sael, N. and Benabbou, F. 2018. Detection of credit card fraud: state of art. *International Journal of Computer Science and Network Security*, 18(11):76–83.
- Sadgali, I., Sael, N. and Benabbou, F. 2020. Fraud detection in credit card transaction using machine learning techniques. In: *1st International Conference on Smart Systems and Data Science (ICSSD)*. Rabat, Morocco, 1–4.
- Saito, T. and Rehmsmeier, M. 2015. The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets. *PloS one*, 10(3), Article e0118432.
- Salahuddin, M.A., Bari, M.F., Alameddine, H.A., Pourahmadi, V. and Boutaba, R. 2020. Time-based anomaly detection using autoencoder. In: *International Conference on Network and Service Management*, 1–9.
- Salehi, A. and Masoumi, B. 2020. Katz centrality with biogeography-based optimization for influence maximization problem. *Journal of Combinatorial Optimization*, 40(1):205–226.
- Saltelli, A., Aleksankina, K., Becker, W., Fennell, P., Ferretti, F., Holst, N., Li, S. and Wu, Q. 2019. Why so many published sensitivity analyses are false: A systematic review of sensitivity analysis practices. *Environmental Modelling and Software*, 114:29–39.
- Sammut, C. and Webb, G. 2010. *Encyclopedia of machine learning*. New York, NY: Springer.
- Sanni, L. and Nurcan, S. 2015. A framework for occupational fraud detection by social network analysis. *CAISE Forum, CEUR*, 1367.
- Saqlain, S.M., Sher, M., Shah, F.A., Khan, I., Ashraf, M.U., Awais, M. and Ghani, A. 2019. Fisher score and matthews correlation coefficient-based feature subset selection for heart disease diagnosis using support vector machines. *Knowledge and Information Systems*, 58(1):139–167.

- Saqr, M., Viberg, O. and Vartiainen, H. 2020. Capturing the participation and social dimensions of computer-supported collaborative learning through social network analysis: which method and measures matter? *International Journal of Computer-Supported Collaborative Learning*, 15(2):227–248.
- Schmidt, M., Fung, G. and Rosales, R. 2007. Fast optimization methods for L1 regularization: a comparative study and two new approaches. In: *Proceedings of the 18th European Conference on Machine Learning*. Warsaw, Poland, 286–297.
- Schreck, T. 2010. Visual-interactive analysis with self-organizing maps - advances and research. *London, United Kingdom: IntechOpen*, 83–96.
- Schreyer, M., Sattarov, T., Borth, D., Dengel, A. and Reimer, B. 2018. Detection of anomalies in large-scale accounting data using deep autoencoder networks. *arXiv preprint arXiv:1709.05254*.
- Schultz, M. and Tropmann-Frick, M. 2020. Autoencoder neural networks versus external auditors: detecting unusual journal entries in financial statement audits. In: *Proceedings of the 53rd Hawaii International Conference on System Sciences*, 5421–5430.
- Siddiqi, N. 2012. *Credit Risk Scorecards: Developing and Implementing Intelligent Credit Scoring*. Vol. 3. Hoboken, NJ: Wiley.
- Sommer, R. and Paxson, V. 2010. Outside the closed world: on using machine learning for network intrusion detection. In: *IEEE Symposium on Security and Privacy*. Berkeley, CA, USA, 305–316.
- Souza, V.M.A., Reis, D.M. dos, Maletzke, A.G. and Batista, G.E.A.P.A. 2020. Challenges in benchmarking stream learning algorithms with real-world data. *Data Mining and Knowledge Discovery*, 34(6):1805–1858.
- Stripe, N. 2020. *Crime in England and Wales: year ending March 2020*. <https://www.ons.gov.uk/releases/crimeinenglandandwalesyearendingmarch2020>. Accessed: 2021-08-10.
- Sudjianto, A., Nair, S., Yuan, M., Zhang, A., Kern, D. and Cela-Diaz, F. 2010. Statistical methods for fighting financial crimes. *Technometrics*, 52(1):5–19.
- Sugar, C. and James, G. 2003. Finding the number of clusters in a data set. *Journal of the American Statistical Association*, 98(463):750–763.
- Sun, J., Zhu, Q., Liu, Z., Liu, X., Lee, J., Su, Z., Shi, L., Huang, L. and Xu, W. 2018. FraudVis: understanding unsupervised fraud detection algorithms. In: *IEEE Pacific Visualization Symposium (PacificVis)*. Kobe, Japan, 170–174.
- Sundararajan, M. and Najmi, A. 2020. The many Shapley values for model explanation. In: *International Conference on Machine Learning*. Vienna, Austria, 9269–9278.
- Suryanarayana, S.V., Balaji, G.N. and Rao, G. Venkateswara 2018. Machine learning approaches for credit card fraud detection. *International Journal of Engineering and Technology*, 7(2):917–920.

- Takan, S. 2020. Creation of mutants by using centrality criteria in social network analysis. *PeerJ Computer Science*, 6:1–22.
- Tervonen, J., Puttonen, S., Sillanpaa, M.J., Hopsu, L., Homorodi, Z., Keranen, J., Pajukanta, J., Tolonen, A., Lamsa, A. and Mantyjarvi, J. 2020. Personalized mental stress detection with self-organizing map: from laboratory to the field. *Computers in Biology and Medicine*, 124, Article 103935.
- Thompson, S.K. 2012. *Sampling*. Hoboken, NJ: Wiley.
- Tu, L.A. and Hoan, N.Q. 2016. Improving feature map quality of SOM based on adjusting the neighborhood function. *International Journal of Computer Science and Information Security*, 14(9):746–757.
- Valeri, M. and Baggio, R. 2020. Social network analysis: organizational implications in tourism managements. *International Journal of Organizational Analysis*, 29(2):342–353.
- Valverde-Albacete, F.J., De-Albornoz, J.C. and Pelaez-Moreno, C. 2013. A proposal for new evaluation metrics and result visualization technique for sentiment analysis tasks. In: *International Conference of the Cross-Language Evaluation Forum for European Languages*. Valencia, Spain, 41–52.
- Van Vlasselaer, V., Bravo, C., Caelen, O., Eliassi-Rad, T., Akoglu, L., Snoek, M. and Baesens, B. 2015. APATE: a novel approach for automated credit card transaction fraud detection using network-based extensions. *Decision Support Systems*, 75:38–48.
- Varmedja, D., Karanovic, M., Sladojevic, S., Arsenovic, M. and Anderla, A. 2019. Credit card fraud detection - machine learning methods. In: *18th International Symposium INFOTEH*. Jahorina, East Sarajevo, 1–5.
- Veeramachaneni, K., Arnaldo, I., Korrapati, V., Bassias, C. and Li, K. 2016. AI2: training a big data machine to defend. In: *2nd International Conference on Big Data Security on Cloud*. New York, NY, USA, 49–54.
- Vellido, A., Angulo, C. and Gibert, K. 2021. A self-organizing world: special issue of the 13th edition of the workshop on self-organizing maps and learning vector quantization, clustering and data visualization. *Neural Computing and Applications*, 1–3.
- Vettigli, G. 2018. *MiniSom: minimalistic and NumPy-based implementation of the Self Organizing Map*. <https://github.com/JustGlowing/minisom/>. Accessed: 2020-07-30.
- Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y. and Manzagol, P. 2010. Stacked denoising autoencoders: learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, 11(12):3371–3408.
- Wan, K., Ma, F. and McKinnon, T. 2022. Covid-19 and cyber fraud: emerging threats during the pandemic. *Journal of Financial Crime*, 29(2):433–446.

- Wang, L., Chen, Y., Jiang, H. and Yao, J. 2020a. Imbalanced credit risk evaluation based on multiple sampling, multiple kernel fuzzy self-organizing map and local accuracy ensemble. *Applied Soft Computing*, 91, Article 106262.
- Wang, X., Du, Y., Lin, S., Cui, P., Shen, Y. and Yang, Y. 2020b. adVAE: A self-adversarial variational autoencoder with Gaussian anomaly prior knowledge for anomaly detection. *Knowledge-Based Systems*, 190, Article 105187.
- Wei, W., Li, J., Cao, L., Ou, Y. and Chen, J. 2013. Effective detection of sophisticated online banking fraud on extremely imbalanced data. *World Wide Web*, 16(4):449–475.
- Whalen, Z. Lu I., Boddeti, V., Dhebar, Y., Deb, K., Goodman, E. and Banzhaf, W. 2019. NSGA-Net: a multi-objective genetic algorithm for neural architecture search. In: *International Conference on Learning Representations*. New Orleans, USA, 1–29.
- Whitrow, C., Hand, D.J., Juszczak, P., Weston, D.J. and Adams, N.M. 2008. Transactions aggregation as a strategy for credit card fraud detection. *Data Mining and Knowledge Discovery*, 18(1):30–55.
- Wilhelm, W.K. 2004. The fraud management lifecycle theory: a holistic approach to fraud management. *Journal of Economic Crime Management*, 2(2):1–38.
- Will, M., Groeneveld, J., Frank, K. and Muller, B. 2020. Combining social network analysis and agent-based modelling to explore dynamics of human interaction: a review. *Socio-Environmental Systems Modelling*, 2, Article 16325.
- Wu, J., Zhao, Z., Sun, C., Yan, R. and Chen, X. 2020. Fault-attention generative probabilistic adversarial autoencoder for machine anomaly detection. *IEEE Transactions on Industrial Informatics*, 16(12):7479–7488.
- Xie, Y., Liu, G., Cao, R., Li, Z., Yan, C. and Jiang, C. 2019. A feature extraction method for credit card fraud detection. In: *2nd International Conference on Intelligent Autonomous Systems (ICoIAS)*. Singapore, 70–75.
- Xu, D., Ricci, E., Yan, Y., Song, J. and Sebe, N. 2015. Learning deep representations of appearance and motion for anomalous event detection. In: *Proceedings of the British Machine Vision Conference (BMVC)*. Swansea, United Kingdom, 1–12.
- Xu, H., Chen, W., Zhao, N., Li, Z., Bu, J., Li, Z., Liu, Y., Zhao, Y., Pei, D., Feng, Y., Chen, J., Wang, Z. and Qiao, H. 2018. Unsupervised anomaly detection via variational autoencoder for seasonal KPIs in web applications. In: *Proceedings of the World Wide Web Conference*. Lyon, France, 187–196.
- Yamanaka, Y., Iwata, T., Takahashi, H., Yamada, M. and Kanai, S. 2019. Autoencoding binary classifiers for supervised anomaly detection. In: *Pacific Rim International Conference on Artificial Intelligence*. Cuvu, Fiji, 647–659.
- Yang, X., Deng, C., Zheng, F., Yan, J. and Liu, W. 2019. Deep spectral clustering using dual autoencoder network. In: *Proceedings of the IEEE/CVF Conference*

- on *Computer Vision and Pattern Recognition (CVPR)*. Long Beach, CA, USA, 4066–4075.
- Yin, C., Zhang, S., Wang, J. and Xiong, N.N. 2020. Anomaly detection based on convolutional recurrent autoencoder for IoT time series. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 52(1):112–122.
- Yin, H., Benson, A.R. and Leskovec, J. 2019. The local closure coefficient: a new perspective on network clustering. In: *12th ACM International Conference on Web Search and Data Mining*. Melbourne, Australia, 303–311.
- Yu, W., Cheng, W., Aggarwal, C.C., Zhang, K., Chen, H. and Wang, W. 2018. NetWalk: a flexible deep embedding approach for anomaly detection in dynamic networks. In: *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. London, United Kingdom, 2672–2681.
- Zhang, P., Wang, J., Li, X., Li, M., Di, Z. and Fan, Y. 2008. Clustering coefficient and community structure of bipartite networks. *Physica A: Statistical Mechanics and its Applications*, 387(27):6869–6875.
- Zhao, J., Kim, Y., Zhang, K., Rush, A. and LeCun, Y. 2018. Adversarially regularized autoencoders. In: *Proceedings of the 35th International Conference on Machine Learning (PMLR)*. Stockholm, Sweden, 5902–5911.
- Zhou, C. and Paffenroth, R.C. 2017. Anomaly detection with robust deep autoencoders. In: *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Halifax, NS, Canada, 665–674.
- Zhou, Z.H. and Liu, X.Y. 2010. On multi-class cost-sensitive learning. *Computational Intelligence*, 26(3):232–257.
- Zhu, M. 2004. Recall, precision and average precision. *Department of Statistics and Actuarial Science*, University of Waterloo, Waterloo Working paper.
- Zhu, R., Ji, X., Yu, D., Tan, Z., Zhao, L., Li, J. and Xia, X. 2020. KNN-based approximate outlier detection algorithm over IoT streaming data. *IEEE Special Section on Innovation and Application of Internet of Things and Emerging Technologies in Smart Sensing*, 8:42749–42759.
- Zimek, A. and Schubert, E. 2017. Outlier detection. *Encyclopedia of Database Systems*, 2635–2639.
- Zimek, A., Schubert, E. and Kriegel, H. 2013. Outlier detection in high-dimensional data. In: *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Gold Coast, QLD, Australia.
- Zimek, A., Schubert, E. and Kriegel, H. 2012. A survey on unsupervised outlier detection in high-dimensional numerical data. *Statistical Analysis and Data Mining*, 5(5):363–387.

Appendix A

Summary of published articles

During the course of this study, four research outputs were produced. The first research output was a full peer-reviewed conference paper titled "A unified approach to anomaly detection", which was based on Chapter 4, and presented at the proceedings of the *6th International Conference on Machine Learning, Optimization, and Data Science*, Siena, Italy, 19-23 July 2020.

The paper was published in Machine Learning, Optimization, and Data Science, which is part of the Springer book series *Lecture Notes in Computer Science*. This can be found at the following URL: https://link.springer.com/chapter/10.1007/978-3-030-64580-9_24

The second research output was a conference presentation titled "Deep anomaly detection using network features", which was based on Chapters 5 and 6, and presented at the *50th Annual Conference of the Operations Research Society of South Africa*, Virtual Conference, 12-15 September 2021.

The third research output was a journal article titled "Anomaly detection using autoencoders with network analysis features", which was based on Chapters 5 and 6. The journal article was accepted by ORiON in July 2022 and is awaiting publication.

Lastly, the fourth research output was a conference presentation titled "Self-organising maps for feature interpretation in anomaly detection", which was based on Chapter 7, and presented at the *51st Annual Conference of the Operations Research Society of South Africa*, Cape Town, South Africa, 12-14 September 2022.

Appendix B

Research output 1

Appendix B contains the full peer-reviewed conference paper titled "A unified approach to anomaly detection". This conference paper was presented at the *6th International Conference on Machine Learning, Optimization, and Data Science*, Siena, Italy, 19-23 July 2020.

A Unified Approach to Anomaly Detection

Richard Ball¹, Hennie Kruger² and Lynette Drevin³

¹ North-West University, Potchefstroom, South Africa, richardstevenball@gmail.com

² North-West University, Potchefstroom, South Africa, hennie.kruger@nwu.ac.za

³ North-West University, Potchefstroom, South Africa, lynette.drevin@nwu.ac.za

Abstract. Anomalous actors are becoming increasingly sophisticated both in methodology and technical ability. Fortunately, the companies impacted by anomalous behaviour are also generating more data and insights on potential anomalous cases than ever before. In this paper, a unified approach to managing the complexity of constructing a useable anomaly detection system is presented. The unified approach is comprised of three algorithms, a Neural Architecture Search (NAS) implementation for autoencoders, an anomaly score threshold optimisation algorithm, and a Gaussian scaling function for anomaly scores. NAS is applied to a data set containing instances of credit card fraud. The NAS algorithm is used to simulate a population of 50 candidate deep learning architectures, with the best performing architecture being selected based on a balanced score, comprised of an average of the Area under the ROC, Average Precision and normalised Matthews Correlation Coefficient scores. The threshold optimisation algorithm is used to determine the appropriate threshold between the classes, for the purposes of producing the binary classification outcome of each architecture. The Gaussian scaling algorithm is applied to the raw anomaly scores of the optimal architecture into order to generate useable probability scores. Not only did the proposed unified approach simplify the process of selecting an optimal neural architecture whose output is interpretable by business practitioners and comparable with other probability score producing models, but it also contributes to anomaly detection in a transactional setting by eliminating subjective thresholds when classifying anomalous transactions.

Keywords: Anomaly detection, artificial intelligence, decision support systems, neural networks, evolutionary computations.

1 Introduction

The aim of this paper is to report on the creation of a unified system for detecting anomalies in a transactional setting. Building a neural network-based anomaly detection capability not only requires an effective implementation of a chosen algorithm, but it should also include processes relating to the usability of the capability within an operational environment. Unsupervised learning techniques have proven to be effective for detecting anomalies [13] although their usage contains some uncertainties regarding the choice of an optimal number of clusters in the case of clustering [16], or the choice of a suitable threshold in the case of an autoencoder [10]. An approach is required that supports the experimental nature of unsupervised learning, but also caters for the structured needs of an organization.

The unified system begins with the development of a sparse autoencoder for detecting anomalies. The choice of structure for the sparse autoencoder architecture is justified by means of an implementation of neural architecture search. That is, multiple candidate architectures are simulated, with the simulated architecture that produces the best evaluation metric being selected. The determination of a suitable threshold for classifying anomalies is generally subjective, and this paper proposes an approach for finding the optimal threshold. Following the application of the optimal threshold, the sparse autoencoder produces an anomaly score per instance. These are determined by the mean square error, and are used to determine how anomalous an instance might be. The larger the anomaly score, the more likely that the instance is anomalous. A challenge with these anomaly scores however is that they are difficult to interpret, beyond the fact that higher means more anomalous [11]. A Gaussian scaling function is proposed in order to transform the anomaly scores into a distribution ranging from 0 to 1, a form which is both comparable across different models and understood universally. These transformed scores can therefore be considered the probability that the instance is anomalous in nature.

The motivation for the study is founded in the observation that anomaly detection projects often fail to be deployed in an operational setting due to issues around a lack of interpretability and effective evaluation [15]. The unified approach provides practitioners with a means of constructing a performant architecture, as evaluated against multiple preferred performance metrics for highly imbalanced classes, and producing an output in the form of a probability score.

The remainder of this paper is organised as follows: Section 2 provides some background on the use of unsupervised learning, more specifically autoencoders, for anomaly detection. Following this, Section 3 presents the data set used for experimental research. The implementation of neural architecture search, threshold optimisation, and Gaussian scaling are then illustrated in Section 4, followed by a discussion of the results of the study in Section 5. Finally, conclusions about the research study are made in Section 6.

2 Autoencoders for anomaly detection

For a typical anomaly detection problem, the instances of labelled records indicating anomalous behaviour are far outweighed by those labelled as nonanomalous. This presents a class imbalance problem i.e. one in which the majority class far outweighs the minority. A typical approach to resolving this problem is through the use of various sampling techniques including under sampling the majority or over sampling the minority classes [5].

A problem with under sampling is that potentially useful data may be discarded, whereas with over sampling the problem of overfitting arises through the use of copies (synthetic or otherwise) of existing data [17]. Another problem with using a supervised learning approach to solve the anomaly detection problem is that labels are known *a posteriori*, either through a customer compliant or chargeback from the bank [2].

In contrast to supervised learning, unsupervised learning techniques do not require knowledge of the labels, but rather seek to characterise the distributions of the data related to the problem space. An autoencoder is used for the anomaly detection component of the study. Autoencoders are unsupervised neural networks that are used to learn efficient data encodings and which have been used to detect instances of accounting anomalies [14]. Autoencoders can be described mathematically with lots of computational detail [12], although due to a page limit this detail has been omitted.

According to [8], an autoencoder is a neural network that is trained to try copy its input to its output. In terms of structure, an autoencoder has a hidden layer $\mathbf{h}$ that generates a function used to describe the model's input. It also contains two distinct parts, namely an encoder function $\mathbf{h} = \mathbf{f}(\mathbf{x})$ and a decoder function $\mathbf{r} = \mathbf{g}(\mathbf{h})$. The decoder function $\mathbf{r}$ attempts to produce a reconstruction of the input, hence it is termed reconstruction. It is not feasible nor useful for the autoencoder to simply learn to copy the input $\mathbf{x}$ perfectly. Instead the model is restricted in the sense that they can only copy input that resembles the training data, and that they are only allowed to copy approximately. Because of these restrictions, the model is likely to learn useful representations of the data.

According to [1] training an autoencoder is generally easier than training a deep neural network. Therefore autoencoders have been used as kinds of building blocks for deep neural networks, where each individual layer in the deep network can be represented by an autoencoder that can be trained in isolation. Fig. 1 illustrates the general structure of an autoencoder.

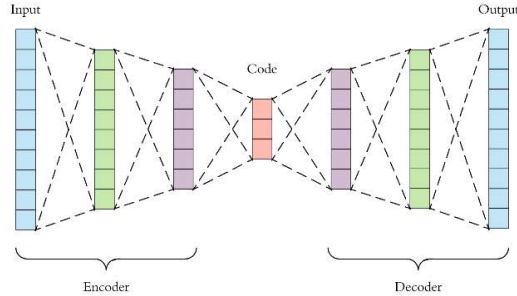


Fig. 1. An example of an autoencoder

3 Data

Data related to anomaly detection is notoriously difficult to obtain according to [13]. Organizations are traditionally somewhat conservative when it comes to publishing data related to cases of in-house anomalies. The reasons for this could be that exposing their internal anomalies might pose reputational risk, or the data may present an opportunity for prospective bad actors to capitalize on potential perceived gaps in the system.

The study was conducted using a publicly available anomaly data set that has been cited in the literature [5]. This particular data set's use is intended to illustrate the concepts for the purposes of this paper, but in follow-up studies the intention is to use real-world data. The public data set was sourced from Kaggle, a website dedicated to hosting machine learning competitions [9], and was collected by Worldline S.A., a European payments provider.

The data contains transactions made over a 2-day period from September 2013. This is a European user credit card data set, and contains over 285,000 transactions of which only 492 (or 0.172%) are confirmed as fraudulent. The data set is labelled and can therefore be used for either classification or anomaly detection purposes. Principal Component Analysis (PCA) has been performed on most of the features, with the resultant data set containing 28 unknown features. There is therefore no way in which to infer identities or additional user information from them. There are also two additional features that have not been processed via PCA, namely the time and amount of the transaction. The labels consist of 1s and 0s, with 1 indicating a confirmed case of fraud and 0 indicating a genuine transaction. Although the data used for this study is transactional in nature, the unified approach can be extended to any anomaly detection application.

4 Methodology

The autoencoder model discussed in this section follows from the discussion in Section 2. In this section Neural Architecture Search, threshold optimization, and Gaussian scaling are applied to the credit card fraud detection data set presented in Section 3.

Neural Architecture Search (NAS) is the process of automatically engineering neural network architectures. According to [6] there are three dimensions to NAS. They are defining the search space, the search strategy, and the performance estimation strategy. The search space relates to the neural architectures that will be simulated; the search strategy relates to how the search space is intended to be explored, while the performance estimation strategy relates to how the performance of each architecture is measured.

The following implementation of NAS simulates a population of candidate architectures. In general researchers typically use a single metric to evaluate the performance of an architecture. In this paper we propose an average of the Area Under the ROC Curve [3], the Average Precision [7], and a normalized Matthews Correlation Coefficient [4] to derive a balanced score that can be used to select the best performing architecture. Each candidate architecture is created by means of assigning it a random depth and a random number of nodes, with an upper bound set to the size of the input layer, for each layer of the corresponding depth. For the purposes of the study, each architecture has the same regularization constraint placed on it.

The search space for this research are chain-structured densely connected neural network architectures, created through the assignment of a randomly generated length and number of nodes per architectural element. The search strategy is deduced by simulating the search for a number of m iterations. Finally, the performance estimation strategy is defined by the balanced score mentioned above. This implementation is presented as Algorithm 1 below.

Algorithm 1 (Neural architecture search algorithm)

Input: number of n training epochs, intended population size m of generated architectures, size of the input layer l

Output: An array $S = \{s_1, \dots, s_m\}$ containing the AUC scores of the m simulated architectures.

```

1 for  $i = 1$  to  $m$  do
2   generate random depth  $d$ ;
3   generate random architecture  $A$ , with a node upper bound  $l$  and length  $d$ ;
4   for  $j = 1$  to  $d$  do
5     if  $A[j] < \min$  then
6        $\min \leftarrow A[j]$ ;
7   end
8 end
```

```

9  remove  $min$  from  $A$ ;
10 encArchitecture  $\leftarrow A$  sorted in descending order;
11 decArchitecture  $\leftarrow A$  sorted in ascending order;
12 build input layer of size  $l$ ;
13 for  $k = 1$  to  $d - 1$  do
14   build encoder layer with encArchitecture[ $k$ ] nodes;
15 end
16 build bottleneck with  $min$  nodes;
17 for  $n = 1$  to  $d - 1$  do
18   build decoder layer with decArchitecture[ $n$ ] nodes;
19 end
20 build output layer of size  $l$ ;
21 train the autoencoder;
22 calculate the balanced score  $s_i$  using test data;
23  $S \leftarrow s_i$ ;
24 end
25 output  $S$ ;

```

When computing the balanced score in Algorithm 1 (step 22), the AUROC and Average Precision metrics can be calculated directly from the raw anomaly score as produced by the autoencoder. For the Matthews Correlation Coefficient (MCC) however, a binary outcome is needed. That one may determine these binary outcomes, a threshold for the raw anomaly scores is required in order to split the scores into a binary classification outcome. A method for determining the optimal threshold for the MCC (or any other binary classification) is presented as Algorithm 2. The threshold optimization process is illustrated in Fig. 3, where it is apparent that the highest MCC score is achieved at a threshold of around 4. The threshold is represented by the gold line in Fig. 2 which attempts to best split the two classes, with the non-anomalous classes mainly occurring below the threshold due to their low reconstruction error. The converse is true for the anomalous class, where most have a high reconstruction error and are therefore above the threshold. Algorithm 2 effectively finds the best fit for the threshold, as determined by the MCC.

Algorithm 2 (Anomaly score threshold optimisation algorithm)

Input: $Y = \{y_1, \dots, y_m\}$ is a linearly spaced vector, starting at zero, and extending to a value of n , containing m samples.

Output: An optimal threshold value $Y[index]$

```

1 for  $i = 1$  to  $m$  do
2   read  $y_i$  from  $Y$ ;
3   calculate the binary classification outcomes for  $y_1$ ;
4   calculate the  $MCC$  score from the binary classification outcomes ;
5   populate array  $R \leftarrow r_i$ , the  $MCC$  value computed ;
6 end
7  $max \leftarrow R[0]$ ;
8 for  $i = 1$  to  $m$  do

```

```

9  if  $R[i] > max$  then
10   $max \leftarrow R[i]$ ;
11   $index \leftarrow i$ ;
12  end
13 end
14 output  $Y[index]$ ;

```



Fig. 2. Distribution of reconstruction errors

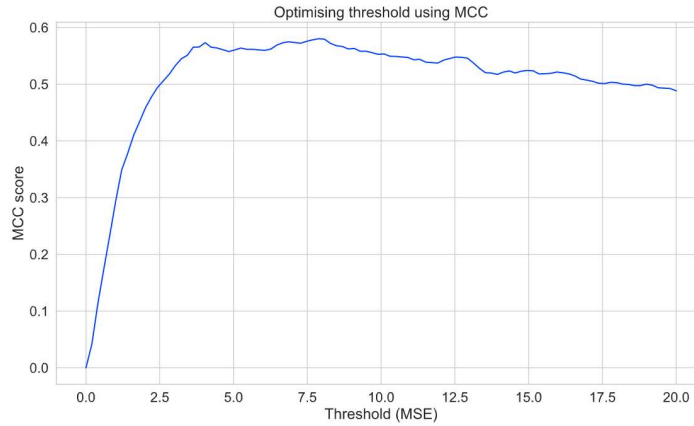


Fig. 3. Threshold optimization using MCC

Following the implementation of Algorithms 1 and 2, a winning architecture is returned. An example of 50 simulated architectures with their corresponding balanced scores is presented in Fig. 4. The best performing architecture is the 8th architecture with a balanced score of 87.6%. This architecture has 10 hidden layers 4687 learnable parameters. The distribution of the 50 balanced scores exhibits a reasonable amount of variance, which is expected considering the random nature of the architecture construction process. Due to the upper bound and depth constraints placed on the con-

struction process, there are no exceedingly bad or failed architectures, with the lowest balanced score at 74.3%.

Fig. 5 illustrates the model loss during training. In order to confirm that the model effectively discriminates between the two classes, the model was reshaped to include a bottleneck of 2 nodes, as presented in Fig. 6. It is clear that there is some degree of separation between the classes in the 2-dimensional latent space.

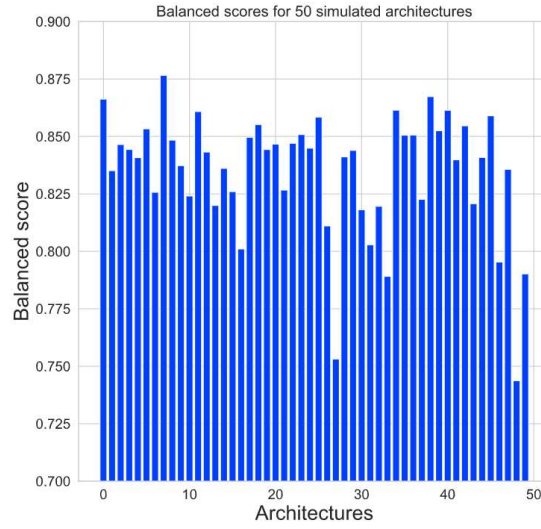


Fig. 4. Balanced scores for 50 simulated architectures

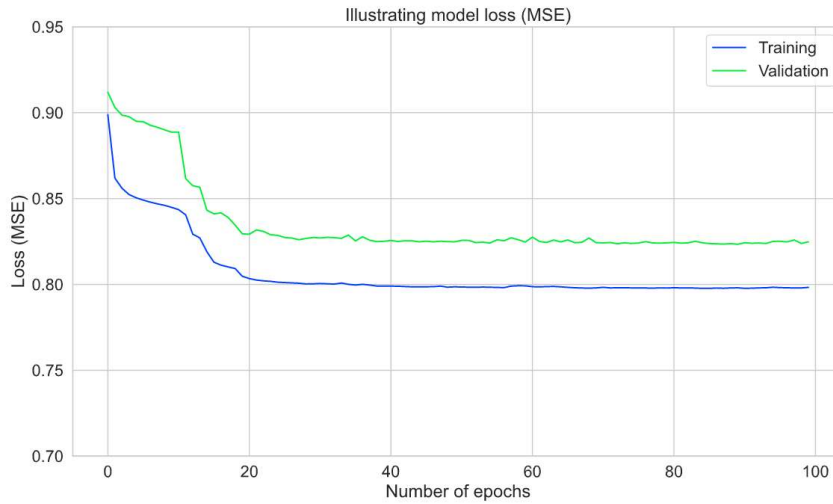


Fig. 5. Illustrating model loss

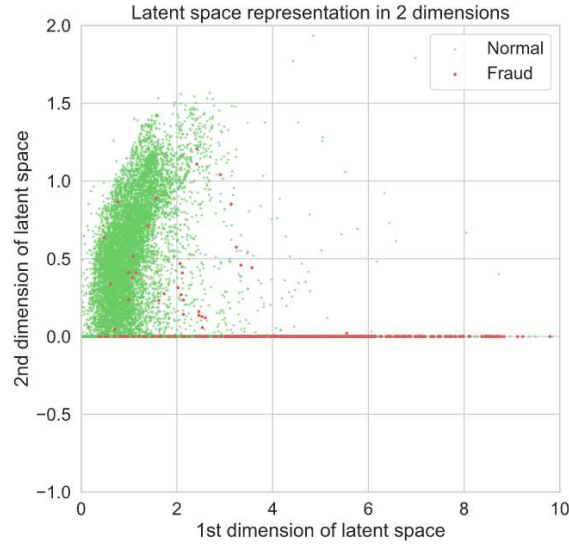


Fig. 6. Latent space representation in 2 dimensions

The output of the winning architecture is an array of reconstruction scores, one for each transaction. These reconstruction scores, as calculated by the MSE, are not entirely interpretable in their current form, as illustrated in Fig. 2. In order to determine the probability that the transaction is anomalous, a normalization approach needs to be taken in order to scale the data to a range of $[0; 1]$. One approach is to apply linear scaling to the scores, although due to the nature of distribution of anomaly scores, this approach does not effectively capture the separation between the classes as presented in Fig. 7.

An alternative approach, applying a Gaussian scaling function to the scores, is proposed in Algorithm 3. [11] propose that Gaussian scaling is a good general choice function for transforming raw anomaly scores. As is illustrated in Fig. 8, the resulting distribution of probability scores shows a much better separation between the classes. With the non-anomalous transactions in green using the left y-axis as its reference, and the anomalous transactions in red using the right y-axis as its reference, a superior split and one which captures the distribution of the underlying anomaly scores is presented. According to Fig. 8, the majority of the normal instances fall within the lowest score tier, whereas the majority of the anomalies fall within the 90%+ tier. This is a favorable outcome for an operational function that makes use of probability scores such as these for their decision-making processes.

Algorithm 3 (Gaussian scaling of anomaly score algorithm)

Input: $S = \{s_1, \dots, s_m\}$ is an array of anomaly scores, containing m samples.

Output: An array $U = \{u_1, \dots, u_m\}$ containing the associated unified scores for m samples.

1 in one pass of array S

2 calculate μ_s ; // mean

3 calculate σ_s ; // standard deviation

4 $cdf = \frac{S(o) - \mu_s}{\sigma_s \sqrt{2}}$; // calculate the cumulative distribution function of S

5 $erf(cdf)$; // transform the cdf with the Gaussian error function

6 normalise the transformation to the range $[0; 1]$;

7 **output** U ;

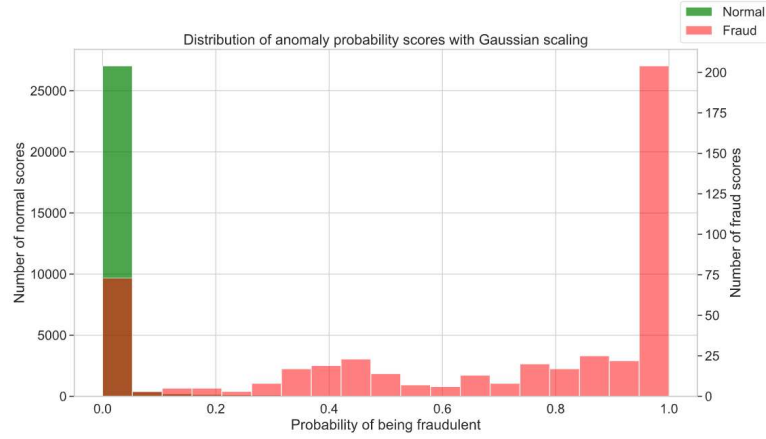


Fig. 7. Distribution of probability scores with linear scaling

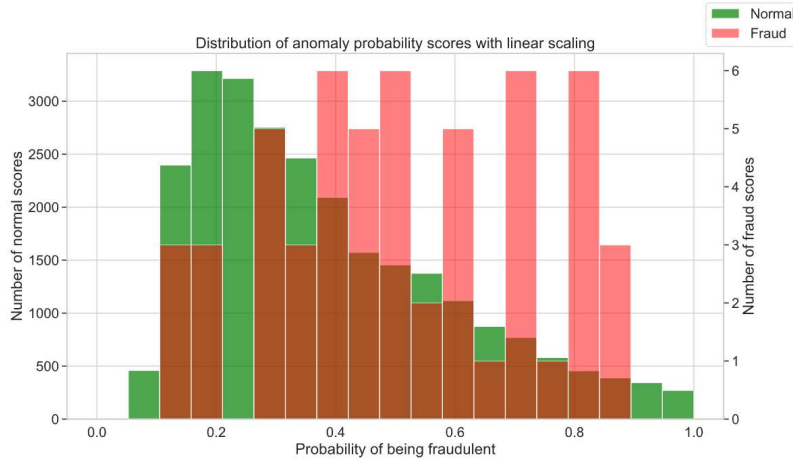


Fig. 8. Distribution of probability scores with Gaussian scaling

5 Results and discussion

Combining NAS and the combination of the 3 metrics for performance measurement provides a formal method for deciding on the appropriate architecture. The approach is straightforward to implement and understand, and the results show an increase in accuracy of around 15% when compared with a trial and error approach. The initial trial and error experiments are not reported in this paper however.

The determination of an optimized threshold is advantageous over the visual inspection approach often used in practice. While it may be possible to visually discriminate effectively, it is unlikely that this will yield a solution that is optimal according to a performance metric well suited to the distribution of the classes. Without the optimized threshold it is a subjective decision to decide which transactions are fraudulent. The optimized threshold approach therefore adds structure and consistency to the anomaly detection problem.

The Gaussian scaling function effectively transforms the raw anomaly scores into a distribution that is indicative of an anomaly detection problem. According to Fig. 7 using linear scaling yields a distribution that contains some level of discrimination between the classes, but as is shown by Fig. 8, Gaussian scaling helps to represent the scores in a manner that represents the discrimination between the classes evident in Fig. 2.

6 Conclusion

The purpose of this study was to report on the creation of a unified approach for detecting anomalies in a transactional setting. An example credit card fraud data set was used throughout the study to evaluate the unified approach.

A population of candidate models was simulated by means of NAS, with the optimal model being selected via the determination of a balanced score, namely an average of three effective evaluation metrics for anomaly detection problems. A threshold optimization algorithm was used to determine the binary classification outcomes within the NAS process. Finally, a Gaussian scaling function was applied to the resulting anomaly scores in order to transform them into a probability score. The resultant scores are easier to interpret than anomaly scores, and provide a more accurate representation of the distribution when compared with a linear scaling approach. Furthermore, the unified approach provides practitioners with a structured method for applying anomaly detection in an organizational context.

Future efforts in this space may be dedicated towards determining an upper bound on the anomaly scores, to investigate whether a better distribution of probability scores can be constructed. Other evolutionary approaches, such as genetic algorithms, may further enhance the NAS process.

References

1. BENGIO Y, 2009, *Learning deep architectures for AI*, Foundations and Trends in Machine Learning, 2(1), p.1-127.
2. CARCILLO F, LE BORGNE Y, CAELEN O, KESSACI Y, OBLE F & BONTEMPI G, 2019, *Combining unsupervised and supervised learning in credit card fraud detection*, Information Sciences, <https://doi.org/10.1016/j.ins.2019.05.042>.
3. CHAWLA NV, 2005, *Data mining for imbalanced datasets: an overview*, Data Mining and Knowledge Discovery Handbook, Springer, Boston, MA, p.853-867.
4. CHICCO D & JURMAN G, 2020, *The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation*, BMC Genomics, 21(1), p.6.
5. DAL POZZOLO A, 2015, *Adaptive machine learning for credit card fraud detection*, PhD thesis, Université Libre De Bruxelles.
6. ELSKEN T, 2019, *Neural architecture search: a survey*, Journal of Machine Learning Research, 20, p.1-21.
7. FAN G & ZHU M, 2011, *Detection of rare items with target*, Statistics and Its Interface 4, p.11-17.
8. GOODFELLOW I, BENGIO Y & COURVILLE A, 2016, *Deep learning*, MIT Press, Available from www.deeplearningbook.org.
9. KAGGLE, 2017, *Credit card fraud detection*, Machine Learning Group - ULB, Available from www.kaggle.com/mlg-ulb/creditcardfraud.
10. KHAN S & TAATI B, 2017, *Detecting unseen falls from wearable devices using channel-wise ensemble of autoencoders*, Expert Systems with Applications, 87, p.280-290.
11. KRIEGLER H, KROGER P, SCHUBERT E & ZIMEK A, 2011, *Interpreting and unifying outlier scores*, 11th SIAM International Conference on Data Mining (SDM), Mesa, AZ, 2011.
12. NG A, 2011, *Sparse autoencoder*, CS294A lecture notes, Stanford University, p.1-19.
13. PHUA C, LEE V, SMITH K & GAYLER R, 2010, *A comprehensive survey of data mining-based fraud detection research*, Intelligent Computation Technology and Automation (ICICTA), p.50-53.
14. SCHREYER M, SATTAROV T, SCHULZE C, REIMER B & BORTH D, 2019, *Detection of accounting anomalies in the latent space using adversarial autoencoder neural networks*, KDD-ADF, Association for Computing Machinery.
15. SOMMER R & PAXSON V, 2010, *Outside the closed world: on using machine learning for network intrusion detection*, IEEE Symposium on Security and Privacy, Berkeley, CA.
16. SUGAR C & JAMES G, 2003, *Finding the number of clusters in a data set*, Journal of the American Statistical Association. 98(463), p.750-763.
17. WEISS GM, MCCARTHY K & ZABAR B, 2007, *Cost-sensitive learning vs sampling: which is best for handling unbalanced classes with unequal error costs*, Proceedings of the 2007 International Conference on Data Mining (DMIN-07), CSREA Press, Las Vegas, NV.

Appendix C

Research output 2

Appendix C contains the conference abstract titled "Deep anomaly detection using network features", presented at the *50th Annual Conference of the Operations Research Society of South Africa*, Virtual Conference, 12-15 September 2021.

Deep anomaly detection using network features

Richard Ball, Hennie Kruger, Lynette Drevin*

Abstract

Features for detecting anomalies in transactional systems are typically related to Recency Frequency Monetary (RFM) attributes that are used to understand transactional behaviour from a user perspective. While these features have proven to be useful in detecting anomalies in transactional data, they fail to effectively capture the interactions and relationships between the users themselves.

In this paper, a unified approach to detect anomalies is proposed. The approach includes using a neural architecture search technique to simulate a number of candidate autoencoder architectures for the purposes of designing an optimal architecture that can be used to classify anomalies. A threshold optimisation algorithm is then applied in order to determine the optimal threshold on validation and test sets, while Gaussian scaling is used to represent the anomaly scores in an interpretable form. The data used in the unified approach is a real-world data set sourced from an organisation in the financial services industry.

To provide for the problem of interactions and relationships between users, an extension to the unified approach is proposed. Additional network metric features are used to augment the standard transactional features, in order to determine whether these network metric features contribute significantly to model performance.

From an organisational perspective, interactions between users can be represented as a network. The individual users in the network act as nodes, whereas the interactions between nodes are represented as edges. Once defined as a network, the influence of any node within the broader network can be established by calculating various network metrics.

Using network metric features in the unified approach yielded a performant anomaly detection model with an acceptable false positive rate. Applying SHAP to the model predictions indicated that a number of the network metric features contributed significantly to the detection of anomalies in a holdout test set.

Appendix D

Research output 3

Appendix D contains the journal article accepted for publication by ORiON, titled "Anomaly detection using autoencoders with network analysis features".

Abstract

Fraudulent activity within a financial ecosystem often involves the coordinated efforts of several bad actors. Expressing the interactions between participants in a system as a mathematical graph allows researchers to apply social network analysis to understand the nature of these relationships better. This article proposes and extends a unified approach using an autoencoder to detect anomalies in a transactional setting. The methodology begins with a neural architecture search to determine the best autoencoder model architecture configuration. This is followed by a threshold optimisation process to find a reconstruction error that best discriminates between non-anomalous and anomalous classes. Gaussian scaling is applied to the raw anomaly scores in order to represent the output in an interpretable and universally transferable form. The unified approach is extended by selecting and including network metrics as features, for the purpose of producing a model that can detect anomalies from both standard transactional data and network data representing the relationships between users within a financial system. Applying SHAP on the model output highlighted the strongest contributing or offsetting network metric features for all anomalies detected. The PageRank and degree centrality network metrics were most important in detecting anomalous instances within the data. Including network metrics in the feature space generated encouraging model performance results, leading to a potential low operational cost of fraud.

Keywords: Anomaly detection, neural networks, autoencoders, neural architecture search, social network analysis, Shapley values

1. Introduction

As the financial landscape becomes more competitive through globalisation and the mass adoption of technology (Skare and Soriano, 2021), the number

of financial transactions being executed via digital channels is increasing. Together with benefits such as efficiency and profitability (Del Gaudio *et al.*, 2020), this also brings more opportunities for fraudulent actors to take advantage of these channels, hence increasing the volume and complexity of the fraud, extortion and money laundering being conducted in the industry (Nikkel, 2020). Due to the increase in suspicious activities, many studies have been conducted to try and find appropriate ways of combatting the nefarious use of digital financial channels (Asha and Kumar, 2021; Henderson, 2020; Saia *et al.*, 2019). Despite these studies, detecting fraudulent transactions remains a non-trivial exercise and has become a common challenge for many organisations. One of the most popular approaches used for detecting anomalous transactions is the use of anomaly detection techniques. An anomaly can be seen as data that deviates substantially from the norm (Mehrotra *et al.*, 2017). There are no universally accepted criteria to define anomaly detection as different mechanisms in different systems may be the cause of an anomaly (Ren *et al.*, 2017; Zhou *et al.*, 2021). A general description, which will be accepted for this study is that anomaly detection is the process of identifying rare observations which differ substantially from the majority of the data from where they are drawn. Various anomaly detection techniques have been proposed in the literature, and examples include the use of outlier techniques (Ji *et al.*, 2020), isolation forests (Lesouple *et al.*, 2021; Liu *et al.*, 2012), and a one-class support vector machine (Wu *et al.*, 2019) algorithms. Other examples can be found in (Lindemann *et al.*, 2021; Xie *et al.*, 2021; Zhou *et al.*, 2021).

Another popular approach for anomaly detection that has gained momentum recently is the use of autoencoders (Misra *et al.*, 2020; Nonnenmacher *et al.*, 2021; Sabetti and Heijmans, 2021; Salahuddin *et al.*, 2020). An autoencoder, which is classified as an unsupervised learning technique, offers a different approach to anomaly detection. The technique aims to learn some low-

dimensional feature representation space on which the given data instances can be well reconstructed (Salahuddin *et al.*, 2020). In a fraud detection context, an autoencoder can be trained to learn to reproduce non-fraudulent transactions only, due to the fraudulent transactions being removed from the training set. When fraudulent transactions are introduced, the autoencoder's reconstruction performance worsens, and this deteriorated performance is then used as the method whereby anomalies are detected. Another characteristic of autoencoders that makes them suitable for anomaly detection is that they are mainly applied to tabular data, which is typically the format of financial transactions. However, attempts have been made to apply them to graphs (Lu *et al.*, 2017) and sequence data (Ding *et al.*, 2019).

Financial transactions in a system typically consist of technical details such as client details and specifics about the transaction such as the amount. A drawback of these financial transactions is that they exclude any information concerning the underlying relationship that may exist among clients, devices, types of transactions etc. Traditional methods of constructing the handcrafted features required for tabular or transactional data may in specific scenarios be less expressive of the underlying phenomena making it again much more complex for fraud investigations. Therefore, network data approaches have been shown to improve model performance by learning a more expressive representation of the problem (Lu *et al.*, 2017). For example, when analysing the behaviour of their customers, organisations frequently resort to understanding how users of a system are connected to one another. These connections are particularly useful when attempting to detect suspicious activity. An example of performing such an analysis can be found in cases where financial crime investigators use social network analysis (SNA) to represent the interconnectedness of multiple users within a system (Burcher and Whelan, 2018).

In this paper, it will be argued that, given the importance of the underlying relationships among transactions in unsupervised anomaly detection, the use of certain basic network metrics in SNA can assist in detecting anomalies by providing metrics that aim to measure the influence that nodes or users have within a transactional system. This includes understanding how the nodes are connected by edges that describe the relationship between users. Having this information available will then improve the accuracy of anomaly detection systems.

The choice of SNA as an additional tool to enhance the unsupervised anomaly detection process offers several advantages. As mentioned, modern organisations are typically networked, meaning that the relationships between their employees, customers, and suppliers can in some sense be linked using data. Examples of data that could link users within a transactional system include shared IP addresses, credit cards, and digital devices. Fraud can naturally occur in any part of the network, and SNA can therefore be used to reveal the connections and behaviours that would be difficult to detect through raw data alone (Burcher and Whelan, 2018). Furthermore, SNA also proves to be beneficial due to the dynamic nature of fraudulent activity. Incorporating network analysis techniques into the fraud detection process helps to augment the performance of human-in-the-loop analysts who typically work in conjunction with static rule-based systems. It is expected that SNA data will provide additional insights that these rule-based systems fail to detect. The incorporation of social network metrics will also provide an alternative viewpoint that highlights connections in a broader network, thereby providing a more comprehensive view of each fraud investigation in the context of a more extensive network. Advantages of a more technical nature include the visual and mathematical capabilities of SNA and the relatively easy translation of data into matrix form, allowing for the calculation of various network metrics that aim to uncover the theoretical

properties of the graph (Lookman and Nurcan, 2015). This would further help to characterise the network as a whole and assist with conducting queries of the social network graph data.

While SNA is an established technique for identifying fraudulent behaviour in financial networks, using it for visualisation purposes alone has proved challenging to work with as the size and complexity of the network increases (Burcher and Whelan, 2018). The network metrics derived from SNA can instead act as features that are used to train a machine learning model able to detect anomalies in a transactional setting. This approach removes the burden on financial investigators having to struggle to interpret large network visualisations and placing less emphasis on them having to explicitly interpret the importance of each of the possible network metrics that typical SNA platforms calculate.

Based on the brief preceding motivational introduction, this paper proposes an approach that combines the strengths of using an autoencoder for unsupervised learning, with selected metrics offered by SNA to perform anomaly detection. This involves using both standard transactional data and network metrics to formulate features used to train a model capable of detecting anomalies in a real-world context. In order to assess the performance of the proposed method, traditional classification metrics suited for fraud detection models and Shapley values are considered (Antwarg *et al.*, 2021). The recommended method implements a previously proposed unified approach to anomaly detection (Ball *et al.*, 2020), where multiple candidate autoencoder architectures were simulated via a neural architecture search.

The remainder of this article is organised as follows: Section 2 provides some literature background on the use of SNA in an anomaly detection context. Section 3 presents the industry data set used for experimental research. A proposed unified approach is then introduced and applied in Section 4, including the use of network metrics as features. The effectiveness of the

approach is evaluated using both Shapley values and standard classification performance metrics, with the results being discussed in Section 5. Finally, the conclusion of the study is presented in Section 6.

2. Related work

Autoencoders and SNA are two different approaches for modelling relationships in data. While the former is frequently used for detecting anomalies in data, SNA can be used to represent the importance of participants in a graph structure. To contextualise the proposed methodology, a cursory overview of examples of existing work related to autoencoders and SNA is presented. Although there are many examples of these concepts in the literature, this study will highlight only a few key examples to put the topics into context and show the wide variety of applications for the approaches. Furthermore, the references aim to provide an understanding of previous attempts to combine both autoencoders and SNA in order to produce a more effective anomaly detection solution.

2.1 Autoencoders for anomaly detection

Various autoencoders have been developed to detect anomalies in different application areas (Chalapathy *et al.*, 2020; Chen *et al.*, 2018; An and Cho, 2015). These can be categorised according to the different architectural variations of the autoencoder, where examples from the literature for some of the categories are presented below.

Schreyer *et al.*, (2018) used a *sparse autoencoder* to detect anomalies in large-scale accounting data. Nine distinct architectures were evaluated and the approach produced reconstruction errors that could be used for a highly adaptive anomaly assessment of journal entry data. In another study, Liang *et al.*, (2020) demonstrated applying a sparse autoencoder to detect pump fault anomalies. Classical principal component analysis is compared with the autoencoder, where the latter is shown to perform better at detecting

anomalies and isolate abnormal features. An ensemble approach is proposed by Narayana *et al.*, (2021), where a sparse autoencoder is applied to detect anomalous attacks in an intrusion detection context. Sparse autoencoders are regularly applied in different domains, and additional examples of the application for anomaly detection can be found in Preethi and Khare, (2021), Mienye *et al.*, (2020) and Tun *et al.*, (2020).

In the category of *variational autoencoders*, Zavrak and Iskefiyeli, (2020) showed that these types of architectures can be used to detect anomalies in network traffic. A semi-supervised approach is utilised to detect network intrusions using flow-based data, and the experimental results illustrated that this approach outperformed a one-class support vector machine baseline model. Variational autoencoders were also found to be helpful in detecting anomalies in multidimensional time-series data (Guo *et al.*, 2018). The authors proposed an approach where a Gaussian mixture model was employed to allow for a series of input distributions. Further anomaly detection applications of variational autoencoders are discussed in Boquet *et al.*, (2020), Chen *et al.*, (2020), and Li *et al.*, (2020).

Denoising autoencoders, another category of autoencoders, also form the basis of many studies. In the area of network anomaly detection, Mohamed *et al.*, (2019) trained a denoising autoencoder to detect network anomalies as part of an intrusion detection system. They found that the model forced the extraction of intrinsic features, which increased the detection accuracy of the intrusion detection system. Another application of denoising autoencoders is reported by Chen *et al.*, (2020). In their study, they achieved success in using stacked denoising autoencoders for detecting anomalies in wind turbine data, where the model was trained on standard operating data, using the reconstruction error as a monitoring indicator.

A popular architecture that is often used in autoencoders is a *convolutional* structure. Some examples to illustrate the use of this architecture include the

following. In a study by Chow *et al.*, (2020) a convolutional autoencoder was leveraged to detect defects (anomalies) in concrete structures. The model was trained in an unsupervised manner, saving the authors from conducting extensive labelling of the data. Superior precision and recall scores were reported in comparison with baseline segmentation methods. Another research project by Wang *et al.* (2021) illustrated that a convolutional autoencoder could be used for hyperspectral anomaly detection, with the intention of detecting observations that differ substantially from their surroundings. Convolutional structures also cater to problems that contain a time interval element. For example, Cui *et al.*, (2018) proposed a convolutional autoencoder to detect system failures from console log data. To capture time dependencies in the logs data, a 2D matrix was constructed to reveal more informative system behaviours, ultimately aiding in determining system failures over time.

The final architecture for which examples from the literature are given is *Long Short-Term Memory (LSTM)*. In a study by Nguyen *et al.*, (2021), the LSTM architecture was demonstrated to be an effective technique for learning long-term dependencies and for representing the relationship between current and previous events. It was shown how LSTM can be included in an autoencoder, for the purposes of detecting anomalies in the supply chain management industry. In another example, Said Elsayed *et al.*, (2020) also employed an LSTM-based autoencoder to detect malicious attacks in network data. The actual anomaly detection classifications were made by a one-class support vector machine algorithm, with the LSTM autoencoder being used to compress the input network data into latent features. Other examples of studies employing LSTM for anomaly detection can be found in Homayouni *et al.*, (2020) and Chawla *et al.*, (2019).

2.2 Social network analysis

SNA has been applied to the problem of fraud and anomaly detection in many ways (Camacho *et al.*, 2020; Pourhabibi *et al.*, 2020; Kurka *et al.*, 2015). Some of these applications fall into the categories of visual analysis, network metrics, feature extraction, and SNA for fraud detection.

Visual analysis is a qualitative application of SNA, which according to Decuypere (2020), involves collecting and coding relational data, visualising network diagrams, analysing the form of network diagrams, and finally interpreting the form of these network diagrams. The study claims that using SNA for visualisation purposes offers the possibility to scrutinise and visualise networks without necessarily inferring causal relationships to the underlying network structures. The benefits of visual analysis were also displayed in the *PeakVis* network visualisation tool developed by Sanvido *et al.*, (2021), which analyses social media data, allowing users to identify relevant segments and grasp the subjects being discussed in real-time. Kucher *et al.*, (2021) agree that SNA can be used for the visual analysis of social and text-based data. Their research project highlighted that besides text data visualisation, SNA can also be used for network data visualisation and exploratory analysis. Additional examples of SNA for visual analysis applications can be found in Borner *et al.*, (2019) and Kivela *et al.*, (2019).

Another important application of SNA is the computation of *network metrics*. Centrality metrics are network metrics that are employed as a means to identify essential elements by determining their behaviour and roles within a complex network (Grando *et al.*, 2018). According to Zhang and Luo (2017), social networks focus on the relation of participants to each other within the network, with network metrics providing a method for surfacing the influence that participants have on the network as a whole. Their findings indicate that degree centrality, betweenness centrality, and closeness centrality are among the most critical network metrics in SNA. Another study by Riquelme *et al.*,

(2018) considered the impact that centrality metrics have on linear threshold models, with results confirming that network metrics were important in ranking actors and networks in a distinguishable way. Network metrics have also shown promise in psychological social networks (Bringmann *et al.*, 2019). The authors state that the three main centrality metrics mentioned previously should be augmented with additional, more relevant centrality metric approaches when dealing with psychological social network data. Suggested examples of more relevant centrality metrics include neighbour-inclusion and relative ranking-related metrics.

Zandian and Keyvanpour (2019) indicated that SNA could be used as a *feature extraction* method, whereby network features derived from SNA were combined with other transactional features to assist in detecting fraud in banking accounts. The authors claim that many fraud detection solutions do not consider both user-level and network-level features simultaneously, and having a method that is able to learn from both kinds of features would increase the accuracy of their overall fraud detection approach. A similar approach was adopted by Beigi *et al.*, (2020), who used linked network data to engineer features to build a signed link analysis model. Due to the sparse nature of the linked network data, other sources of information are included as features to increase model performance. Another example of using SNA for feature extraction is presented by Kumari *et al.*, (2020). In this research, data representing the topological structure of the network is used in conjunction with content-based information to build a model for link prediction. Finally, Oskarsdottir *et al.*, (2021) proposed implementing SNA in the insurance sector, where a network was created by linking claims data to all involved parties. Features from the network were extracted and combined with claim-specific transactional features to train a supervised learning model. This approach yielded results indicating that the combination

of network and claim-specific features improved the performance of supervised learning models.

SNA is also applied extensively in the *fraud detection* domain, where users who transact with one another are modelled as a network, with nodes and edges representing users and the transactions between them respectively. Colladon and Remondi (2017) proposed using SNA to prevent money laundering activities, where network metrics were explicitly created to model clients' risk profiles within the network. Interesting to note that the authors suggest the use of a hybrid approach, with SNA being used to determine the risk profiles through network metrics and create a visual analysis of the network to gain a broader understanding of suspicious behaviour. Pourhabibi *et al.* (2020) highlighted many of the key graph-based fraud and anomaly detection approaches. The review argued that social networks are typically used for anomaly detection in a two-step approach; namely for selecting and calculating network features and classifying observations from the feature space determined via the first step. Other researchers (Leite *et al.*, 2018) argue that visual analysis is also an essential tool for fraud detection, including examples of the technique being applied across multiple industries such as telecommunications, stock exchanges, insurance, and banking. Visual analysis is then considered in parallel to more analytical network metrics approaches, with the two seen as influencing each other during pre-processing, post-processing, integrated, and pure visualisation processes.

The preceding examples from the literature clearly indicate the importance and prominence of employing different architectures of autoencoders for anomaly detection. From the studies quoted, it is also clear that other existing techniques such as SNA can be used to evaluate and analyse data for fraudulent activities and that this technique is often used to complement other anomaly detection methodologies.

3. Modelling prerequisites

In this section, a brief review of the concepts that are necessary to construct the network features for unsupervised anomaly detection is presented. The relevant modelling concepts include autoencoders for anomaly detection, the previously proposed unified approach for anomaly detection (Ball *et al.*, 2020), and SNA.

3.1 Modelling basics of an autoencoder

Autoencoders are neural networks that can be used to detect anomalies in an unsupervised or semi-supervised manner. According to Goodfellow *et al.*, (2016), autoencoders are trained to reconstruct its input to its output, an example of which is provided in Fig. 1. The encoder in Fig. 1 is responsible for compressing the feature space down into a lower dimension, as determined by the bottleneck layer. The compressed data are then mapped back into a reconstructed vector by the decoder function, where the encoder and decoder are typically symmetrical architectures, each containing multiple neuron layers.

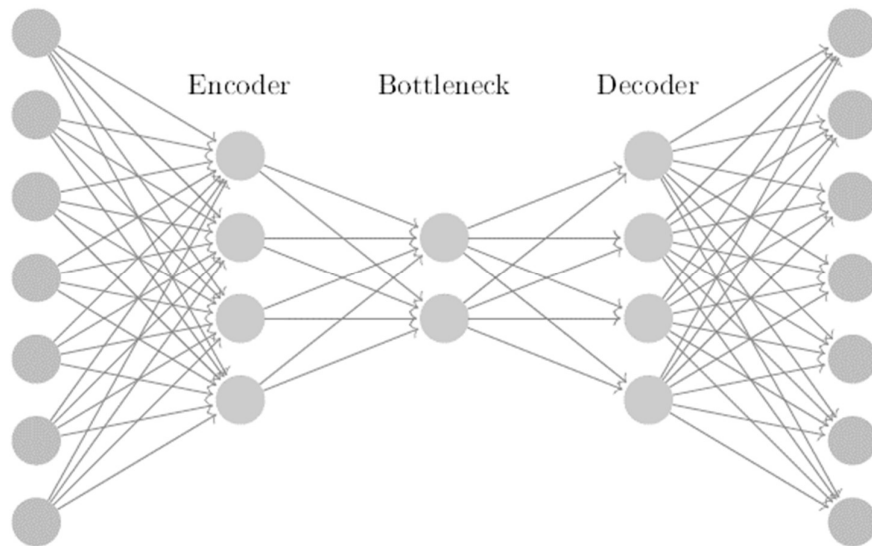


Fig. 1. An example of the architecture of an autoencoder

The main components of an autoencoder can be explained as follows (Schreyer *et al.*, 2018). The model is trained using a set of N aggregated records

$$X = \{x^1, x^2, \dots, x^n\}$$

where each of the aggregated records consists of a tuple of k transactional attributes

$$x^i = (x_1^i, x_2^i, \dots, x_j^i, \dots, x_k^i)$$

with x_j^i representing the j^{th} transactional attribute of the i^{th} aggregated record.

The network contains a hidden layer h which generates a function that describes the input of the model. The model architecture also includes an encoder function $h = f_\theta(x)$ and a decoder function $r = g_{\theta(h)}$, which both assist as nonlinear mappings. More specifically, the encoder function maps an input vector x^i into a compressed representation z^i within the latent space Z . The compressed representation is then mapped back onto a reconstructed vector $\hat{x}^i$ by the decoder function, with the reconstruction of the original input data being represented by $\hat{x}^i$. While not always the case, the encoder and decoder are typically symmetrical in nature, where deep autoencoders are those considered to contain a number of hidden layers within the encoder and decoder.

The encoder can be defined as

$$f_\theta^l(.) = \sigma^l \left(W^l \left(f_\theta^{l-1}(.) \right) + b^l \right)$$

while the decoder mapper is defined as

$$g_\theta^l(.) = \sigma'^l \left(W'^l \left(g_\theta^{l-1}(.) \right) + d^l \right)$$

where σ and σ' denote the nonlinear activations, which often takes the form of the rectified linear unit (ReLU) activation function. W , b and d represent

the model weights and biases respectively, while θ represents the parameters of the model. The number of hidden layers in the model are described by l .

Training the autoencoder entails attempting to achieve an accurate representation of the input data, with the model learning parameters θ that minimise the reconstruction error between the input and the reconstructed output. The training objective therefore optimises

$$\arg \min_{\theta} \|X - g_{\theta}(f_{\theta}(X))\|.$$

The approach is well suited for anomaly detection, where the autoencoder is used to learn the most salient features in a data set, with the intention of reproducing these salient features. The output, namely the reconstructed features, is the model's attempt at producing a reconstruction of the input data based on the data that was used to train it. If the input data were similar to the data used to train the model, then the output would be similar to the input, namely due to the model having learnt an effective input mapping function via the training process. This is relevant for anomaly detection purposes because input data that is dissimilar from the data used to train an autoencoder will produce an inferior reconstruction. The quality of the reconstruction can be measured by means of the mean squared error (MSE), which calculates the mean of the squares of the differences between the reconstruction and its input. The MSE is known as the reconstruction error in this context, where a higher reconstruction error is achieved in instances where the reconstruction deviates substantially from the input data. Conversely, a lower reconstruction error is achieved in cases where the reconstruction is more similar to the input data.

This approach of assessing the quality of the reconstruction using the MSE is used to detect input data that differs substantially from the data that was used to train the autoencoder. An autoencoder can therefore be trained using non-anomalous data only. As soon as anomalous data are introduced as the input,

the MSE would determine that the mapping function is not familiar with the data, thereby producing a high reconstruction error. This has powerful implications for fraud detection, where an autoencoder can be trained on non-anomalous transactional data only, for the purposes of detecting anomalous transactions. Of particular importance is the fact that, by definition, anomalous data occurs less frequently.

3.2 A unified approach

One of the frequent problems in implementing autoencoders, and neural networks in general, is determining an optimal model architecture. No single solution is guaranteed to deliver an optimal architecture, which is important because architecture design considerations substantially impact model results. Traditionally, researchers used a grid search method where a large number of hyperparameter combinations are explored and the best one selected (Sanchez *et al.*, 2021). There are also other suggestions in the literature (Pidhorskyi *et al.*, 2020; Principi *et al.*, 2019) on how to construct autoencoder architectures.

In this study, it was decided to use a unified approach previously proposed by Ball *et al.*, (2020) to ensure that the architecture of the chosen autoencoder is both derived using a structured methodology, and is optimal from a model performance perspective. The unified approach uses a combination of performance metrics proven to be well suited for anomaly detection problems and extensive experimentation conducted in Ball *et al.*, (2020), produced satisfactory and highly accurate results. Other benefits of the approach include the absence of having to make subjective visual inspections of the model threshold and having an interpretable model output that is transferable across business domains.

The approach is comprised of three main phases that are performed sequentially.

- A neural architecture search is performed by simulating multiple autoencoder architectures;
- A threshold optimisation algorithm is applied to the simulated architectures to assist with determining optimal fraudulent binary classification outcomes; and
- A Gaussian scaling process is implemented to prohibit any subjective interpretation of the raw anomaly scores

In the remainder of this section, the algorithms for the three main phases of the proposed approach will be highlighted, while the details as well as the initial experimental results can be found in Ball *et al.*, (2020).

Neural architecture search: Algorithm 1 presents the first phase of the approach that is employed to simulate a population of candidate autoencoder architectures. The architectures themselves are created by assigning them randomly generated depths and numbers of nodes. An upper bound is applied to the size of the input layer of each architecture, with each architecture operating under the same regularisation constraint. A balanced metric, comprised of the average of the area under the receiver operating characteristic (ROC) curve (AUC), average precision (AP), and normalised Matthews correlation coefficient (MCC), is derived in order to select the best performing architecture. ROC is a curve that plots the true positive rate against the false positive rate of a binary classifier, to determine the classification performance across different thresholds. The AUC (Johnson and Khoshgoftaar, 2019), AP (He *et al.*, 2018) and MCC (Chicco and Jurman, 2020) have all been used effectively as classification performance metrics for problems with a class imbalance. Combining these into an average or balanced score helps normalise the different magnitudes of each performance metric. This is especially important for the AP metric as the baseline for the AP is the fraction of positive classes (Saito and Rehmsmeier, 2015). In a typical anomaly detection problem, this could lead to an AP baseline of 1%

or lower depending on the severity of the imbalance. The AP baseline is in contrast to the AUC metric which has a baseline of 50%.

Algorithm 1: Neural architecture search algorithm

Input: number of n training epochs, intended population size m of generated architectures, size of the input layer l

Output: An array $S = \{s_1, \dots, s_m\}$ containing the balanced scores of the m simulated architectures.

```

1  for  $i = 1$  to  $m$  do
2      generate random depth  $d$ ;
3      generate random architecture  $A$ , with a node upper bound  $l$  and
        length  $d$ ;
4      for  $j = 1$  to  $d$  do
5          if  $A[j] < min$  then
6               $min \leftarrow A[j]$ ;
7          end
8      end
9      remove  $min$  from  $A$ ;
10     encArchitecture  $\leftarrow A$  sorted in descending order;
11     decArchitecture  $\leftarrow A$  sorted in ascending order;
12     build input layer of size  $l$ ;
13     for  $k = 1$  to  $d - 1$  do
14         build encoder layer with encArchitecture[ $k$ ] nodes;
15     end
16     build bottleneck with  $min$  nodes;
17     for  $n = 1$  to  $d - 1$  do
18         build decoder layer with decArchitecture[ $n$ ] nodes;
19     end
20     build output layer of size  $l$ ;

```

```

21   | train the autoencoder;
22   | calculate the balanced score  $s_i$  using test data;
23   |  $S \leftarrow s_i$ ;
24 end
25 output  $S$ ;

```

Threshold optimisation: The second phase of the approach entails further evaluating the MCC component of the balanced score obtained in Algorithm 1. The MCC requires that the output of the autoencoder architecture be presented in binary form, where the class labels come from two distinct groups, namely positive and negative classes. This is not necessarily the case for AUC and AP, which can be calculated directly from the raw anomaly score produced by the autoencoder. In order to represent the raw scores in binary form, a threshold is required to split the scores into a binary fraudulent classification outcome. Determining a threshold is subjective, where an optimal split between the non-anomalous and anomalous classes are sought. Traditionally, the maximum reconstruction error achieved on the training set was used as the threshold for classifying anomalies (Dau *et al.*, 2014). Other approaches suggest using the proportion of anomalous data in the training data as a heuristic for threshold determination (Khan *and* Taati, 2017). Neither of these approaches considers classification performance metrics as part of the thresholding decision, instead they evaluate the performance of their thresholds using classification metrics after the fact. A formalised method for determining an optimal threshold is introduced as Algorithm 2, where the balanced score is incorporated as part of the thresholding decision. More specifically, a range of possible threshold values is simulated to determine the impact that each threshold has on the balanced score. The threshold that achieves the highest overall balanced score is chosen as the optimal threshold.

Algorithm 2: Anomaly score threshold optimisation algorithm

Input: $Y = \{y_1, \dots, y_m\}$ is a linearly spaced vector, starting at zero and extending to a value of n , containing m samples.

Output: An optimal threshold value $Y[index]$

```
1  for  $i = 1$  to  $m$  do
2      read  $y_i$  from  $Y$ ;
3      calculate the binary classification outcomes for  $y_1$ ;
4      calculate the  $MCC, AUC$  and  $AP$  scores from the binary classification
      outcomes ;
5      populate array  $R \leftarrow r_i$  // the balanced score computed
6  end
7   $max \leftarrow R[0]$ ;
8  for  $i = 1$  to  $m$  do
9      if  $R[i] > max$  then
10          $max \leftarrow R[i]$ ;
11          $index \leftarrow i$ ;
12     end
13 end
14 output  $Y[index]$ ;
```

Gaussian scaling: The final phase in the unified approach is scaling the raw scores to prevent subjective interpretation. The optimal architecture can be discovered after implementing Algorithms 1 and 2, with the output of the autoencoder model taking the form of a reconstruction error. This error is the MSE between the actual input data and the reconstructed input data that the autoencoder produced. The reconstruction errors are the raw anomaly scores, with a higher score indicating that the underlying transaction has a high likelihood of being anomalous. Interpreting raw anomaly scores is a subjective exercise, which presents an opportunity to normalise the scores

into an easy to interpret range of [0; 1]. A simple approach entails applying linear scaling to the scores, although this approach does not capture the separation between the classes as required. Alternatively, a Gaussian scaling approach can be applied, as presented in Algorithm 3.

Algorithm 3: Gaussian scaling of anomaly score algorithm

Input: $S = \{s_1, \dots, s_m\}$ is an array of anomaly scores, containing m samples

Output: An array $U = \{u_1, \dots, u_m\}$ containing the associated unified scores for m samples

```

1  for one pass of array  $S$  do
2      calculate  $\mu_s$ ; // mean
3      calculate  $\sigma_s$ ; // standard deviation
4       $cdf = \frac{s(o) - \mu_s}{\sigma_s * \sqrt{2}}$ ; // calculate the cumulative distribution function of  $S$ 
5       $erf(cdf)$ ; // transform the  $cdf$  with the Gaussian error function
6      normalise the transformation to the range [0; 1];
7  output  $U$ ;
```

Kriegel *et al.*, (2011) found that the Gaussian technique was a good general choice for transforming raw anomaly scores from a wide variety of anomaly detection algorithms. Applying Algorithm 3 resulted in a far superior separation between the non-anomalous and anomalous classes, with the majority of the non-anomalous class residing in the lowest score tier and the majority of the anomalous class falling within the higher, 90%+ score tier. Having a distribution of scores that clearly discriminates between the two underlying classes is useful in an operational context, where fraud detection practitioners can easily discern the likelihood of a transaction being anomalous based on its probability score.

By combining a neural architecture search, threshold optimisation, and Gaussian scaling, the unified approach provides practitioners with a

structured methodology to create autoencoder models for anomaly detection purposes. The output of the process is a trained autoencoder model, with the threshold calibrated according to performance metrics relevant for problems with imbalanced classes.

3.3 Basic concepts of social network analysis

Social networks refer to the structure of social actors that share interests of activities (Sanni and Nurcan, 2015). O'Malley and Onnela, (2014) also describe SNA as the understanding of the interdependencies in behaviour related to configurations of social relations. Both the relationships between individuals and their attributes can be considered the observations of interest within a social network. The observations within a social network are thought to be dependent in part on other observations within the network, due to their interconnected nature. Therefore, the techniques used to analyse the dependent data in social networks differs somewhat from the traditional statistical methods used to model independent observations.

In SNA, a graph or network is a finite set of n nodes or vertices that are connected by a finite set of m edges. In a financial services context, these nodes could be considered users of a particular system, and the edges would represent the connections or relationships between the users. A graph is typically referred to as G and is formally expressed as $G = (V, E)$. V is a finite set, called the vertices of G , while E is a set of two-element subsets of V , called the edges of G . The edges E are unordered pairs of vertices (Bender and Williamson, 2010). Fig. 2 illustrates an example of a simple graph. The graph has five nodes as represented by the circles and six edges as represented by the pairs of two letters highlighting the relationships between the nodes. To give an example explaining how Fig.2 works, B and C represent nodes, whereas BC is the line or edge that connects these two nodes.

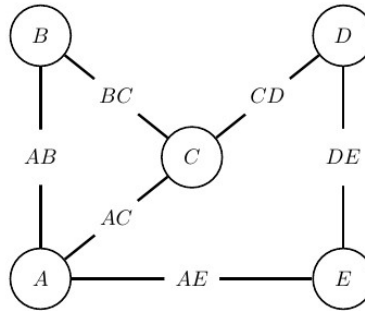


Fig. 2. An example of a simple undirected graph

The edges between nodes within a graph can also have a directional element to them, where the graph is referred to as a directed graph. A directed graph (or digraph) places importance on the direction of the relationship between the nodes. An example of a directed graph is displayed in Fig. 3. The direction of the arrow between nodes A and B, for example, confirms that the edge flows in the direction from node A to node B. Directed graphs can also have edges that flow in both directions as well as edges that flow from the starting node back to itself (self-loops).

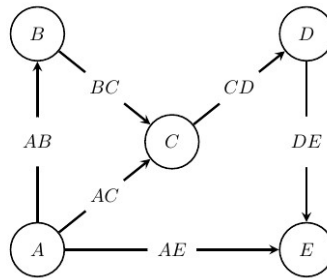


Fig. 3. An example of a directed graph

A numerical value can also be assigned to the edges within a graph to represent the importance of the edge, a concept known as a weighted graph. The importance can be incorporated by using numerical values or other drawing techniques, where the less important edges are indicated by the dashed lines in Fig. 4.

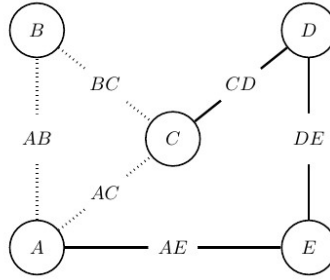


Fig. 4. An example of a weighted graph

Graphs can also be further divided into smaller components known as subgraphs. A subgraph H of graph G is considered a clique if each node in the subgraph is connected by an edge to every other node within H . A particular subgraph H can also be thought of as a maximal clique in G in cases where H is a clique and there is no node in G but not in H that sends an edge to every node of H (Newman, 2010).

SNA is a broad field of study and many relations and sub-specialisations have been identified and studied. Because this paper uses SNA and some of its associated metrics, it was necessary to mention only a few of the basic ideas of SNA. This section is by no means an exhaustive overview of graph theory or SNA. The field itself is a highly mathematical area of study, and although the mathematical details do not form part of the scope of this research, interested readers are referred to Balakrishnan and Ranganathan, (2012) for more detail on the mathematical foundations of graph theory.

4. Methodology and experimental work

The validation of the proposed approach using a real-world data set sourced from the financial services industry is presented in this section. The discussion includes the data used in the study, the creation of network features using SNA, conducting a practical experiment whereby the network features are included to train an anomaly detection model, and finally, a discussion on the experimental results of the study.

4.1 The transactional dataset

The study was conducted using a data set sourced from an organisation operating in the financial services industry. The data are anonymous in nature, and care was taken not to expose any sensitive user information during the data extraction process. Two components make up the data set, namely a set of standard transactional features and a set of links between the same users in the set of standard transactional features, which makes the calculation of SNA metrics possible.

The standard transactional features are comprised of 120,676 records, of which 439 records are confirmed as fraudulent. This equates to a class imbalance where only 0.36% of the data are attributed to the minority class. The reason for the imbalance is that levels of fraud usually are far lower than the number of valid transactions that an organisation process. Therefore, far more data are collected on valid transactions compared with anomalous transactions, hence the imbalance in favour of the valid transactions. Highly imbalanced data sets are also reported in the literature as commonplace in the fraud detection context (Makki *et al.*, 2019). While class imbalances generally occur in standard classification problems, highly imbalanced tend to fall into the anomaly detection domain, where algorithms have been specifically developed to handle these imbalances. Standard classification approaches are not well suited to highly imbalanced problems, as the majority class overwhelms the training process and performance metrics associated with these. An example of a sampling technique that addresses oversampling the minority class is the synthetic minority oversampling technique (SMOTE) (Drummond and Holte, 2003).

To address the problem of an imbalanced data set, this paper uses stratified sampling combined with a well-known and proven anomaly detection approach that trains a model on the majority class only, for which there is sufficient data (Gerych *et al.*, 2019). This combined methodology was

devised in order to prevent bias when creating the various data partitions required for effective model training and performance evaluation.

Stratified sampling is a sampling approach where the population data set is partitioned into non-overlapping groups known as strata (Howell *et al.*, 2020). In the case of a binary classification problem, the population data set can be partitioned into two strata, namely the positive and negative classes. When constructing training, validation, and test data sets, proportional allocation of the classes can be applied via stratified sampling. This means that any class imbalance exhibited in the population data set can be enforced in the subsequent training, validation, and test set samples, ultimately removing potential bias from the sampling process by ensuring that the balance of the classes is represented correctly in all samples.

The approach to train a model on the majority class only, produces a function that can be leveraged to detect non-anomalous transactions that are similar to the data that were used to train the model. Conversely, the anomaly detection model would flag anomalous transactions that are highly dissimilar from the data used to train the model. This is advantageous, as it allows researchers and practitioners to circumvent the issues related to training a binary classifier on data with limited instances of the minority class. In cases where there are no opportunities to collect additional positively labelled data, or where over and under-sampling techniques fall short, this approach becomes a viable option.

The sampling and model building, based on the above methodologies, is presented in Figure 5 and can be summarised as follows. The sampling process begins with $n = 120,676$ records, which are sorted in random order to prevent sampling bias. These are then split into training set T_1 and test set T_2 , where T_1 contains 90% and T_2 contains the remaining 10% of the transactional dataset D . T_1 and T_2 are split using stratified sampling, so that each resulting

partition contains a proportional share of the anomalous class. T_2 can then be considered the hold-out test set, while T_1 is used for further processing.

Training set T_1 is randomised again and subsequently split into a further training set T_3 and a validation set T_4 . T_3 comprises 90% of T_1 while T_4 is made up from the remaining 10%, with stratified sampling again ensuring that the partitions receive their proportional share of the minority class.

S_1 is the set of fraudulent instances in T_3 , and is removed so that the final training set contains data from the non-anomalous class only. On the other hand, validation set T_4 keeps its set of fraudulent instances S_2 . This enables training on the non-anomalous cases only while validating the performance on data including both classes. Following the determination of the optimal architecture, hold-out test set T_2 is used to make a final evaluation of the model's performance on unseen data.

The set of fraudulent instances (S_1) removed from T_3 are then added onto validation set T_4 to augment the validation process with additional fraudulent samples. Removing the set S_1 from the training set yields the final training set T_5 , whereas the updated validation set including S_1 becomes T_6 . The final validation set T_6 ends up including set S_1 , which contains 395 fraudulent instances, while the hold-out test set T_2 contains set S_2 with 44 fraudulent instances. Together these make up the set S , which includes all 439 fraudulent instances from the original transactional data set.

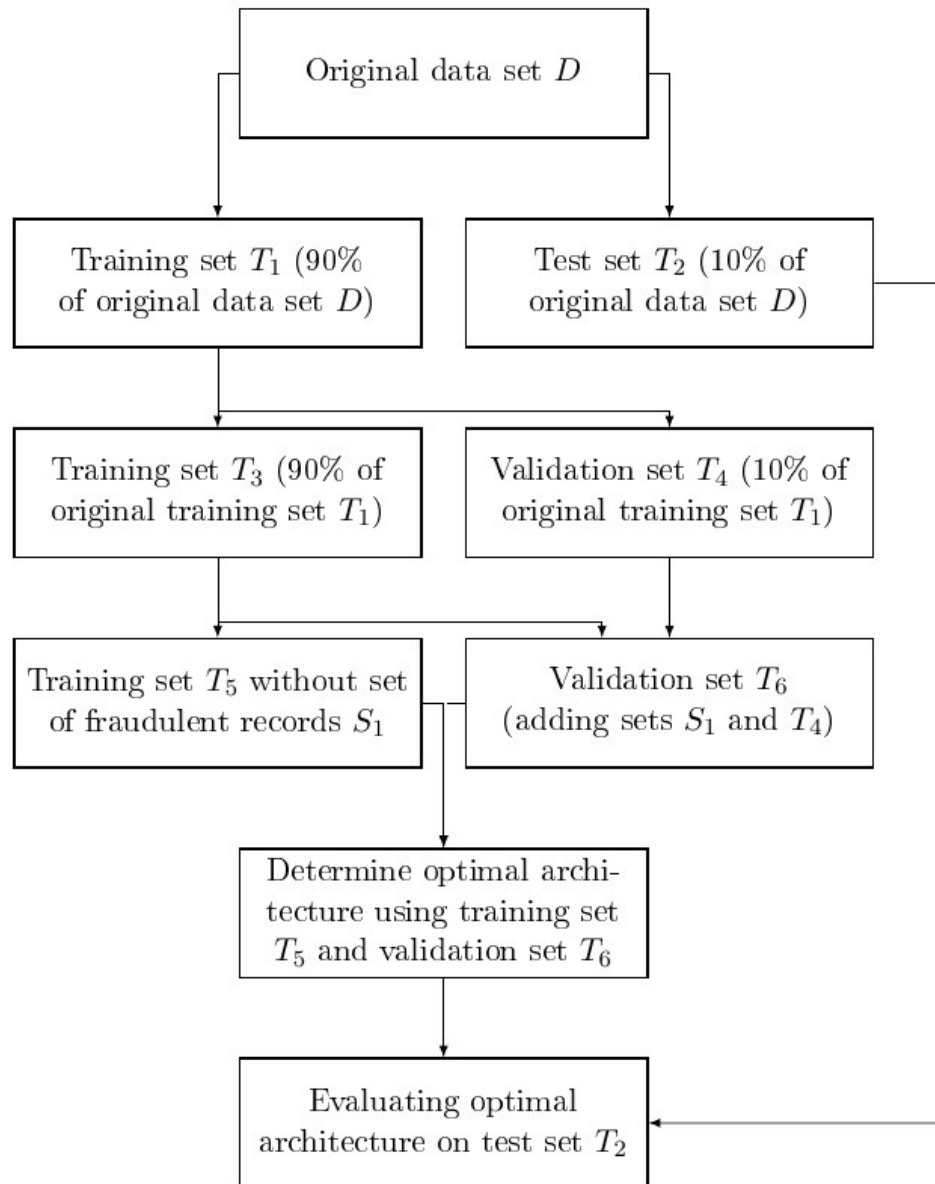


Fig. 5. Sampling process

There are 49 features in the data set, where one is the unique identifier of each user, and another is the class label or target variable indicating a confirmed credit card fraud case. Instances of confirmed fraud are assigned a target variable of 1, while instances of no credit card fraud are assigned a target

variable of 0. For the purposes of model training, the identifier is excluded but can be used to map the vectors and their respective prediction output back to each user's unique identifier. The class label is also kept aside but will be used for classification purposes. Therefore, the 49 features are reduced to 47 features for the purposes of training each model. No missing values are found in the data set, as the aggregation process used returns a 0 rather than a null value in instances where no values are counted for the respective feature. The real-world data features are transformed via standardisation to a $N(0,1)$ distribution. Standardisation entails centering each feature around its mean with a unit standard deviation. The values returned by the standardisation process are not defined to a particular range as is the case with normalisation, and therefore values greater than 1 are to be expected.

Due to the confidentiality of the data, and considering the sensitivity of financial data, the remaining 47 features will not be described in detail. These features are typical features found in a transactional database and relate to financial and behavioural attributes associated with each transaction. Examples include the IP address from where the transaction originates, how many cards the user has attempted to use, and whether there are differences between which country the user is signed in from and where their identity documents suggest they are located.

4.2 Extending the unified approach to include social network analysis metrics

Following on from the creation of the transactional features in Section 4.1, data representing the links or relationships between users are considered in order to structure the users and their interactions as a graph. Examples of links between users would include the same credit card, IP address, or login device used. Using the links between users as edges, and the users themselves as nodes, a network structure was formed, which could be explored via several network modelling techniques. This exploration led to the creation of features

describing the underlying network metrics. The intention is to extend the unified approach by incorporating SNA metrics into the transactional features so that a final, comprehensive feature set can be used for anomaly detection.

Considering the large size of the clusters of network data, only certain visual examples of the process to incorporate the social network metrics will be given. Firstly, a smaller cluster with predominantly non-anomalous nodes is shown in Fig. 6, highlighting the non-anomalous and locked nodes, as well as the edge connections between them. In Fig. 6, the white nodes indicate non-anomalous users, whereas the black node indicates a user account in a locked state. A locked state is defined as the state when an account has been closed and all of its associated functionality has been removed. This can be temporary or permanent, depending on the severity of the suspicious activity associated with the account. The edges are comprised of a combination of device or IP address links between users.

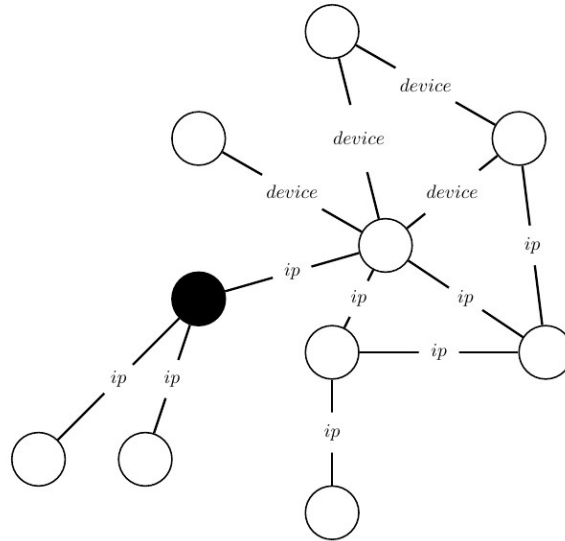


Fig. 6. An example of a subgraph within the real-world dataset

The largest cluster in the entire social network is presented in Fig. 7. This cluster is a densely connected web of anomalous, locked, and non-anomalous

accounts and contains 1,515 distinct nodes, including 8 instances of fraud as indicated by the large red nodes. Some segments of the cluster splinter off into smaller, yet connected clusters, while the majority of the remaining nodes form a dense mesh of hundreds of accounts. This cluster, while alarming, would be challenging to investigate manually.

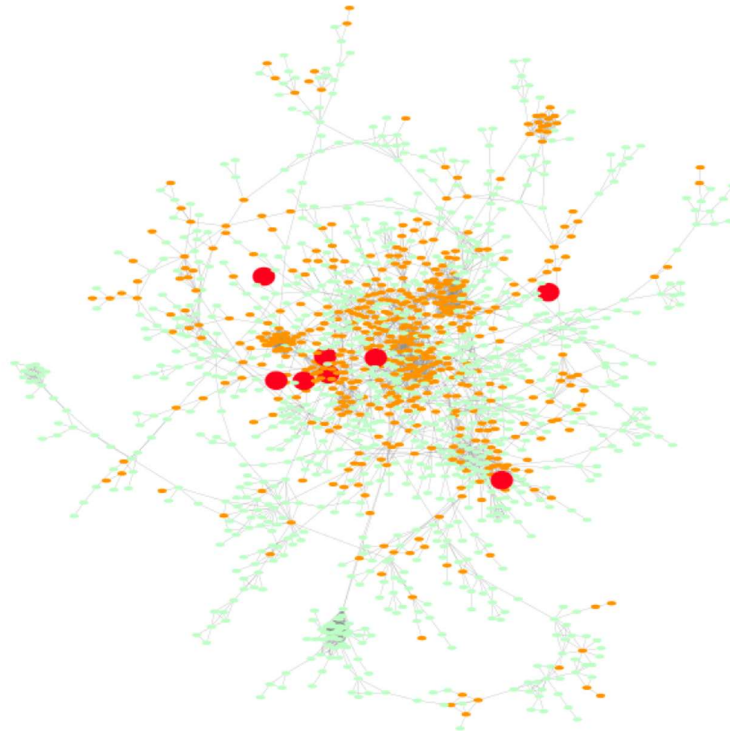


Fig. 7. Largest cluster in the overall social network

In terms of selecting the appropriate network metrics to calculate on this graph, a number of related studies were considered to provide assistance with these choices. Centrality metrics were an essential set of features used by Van Vlasselaer *et al.*, (2015) and Bonilla (2018), prompting the selection of the degree, eigenvector, closeness, betweenness, load, and harmonic centrality in this study. Van Vlasselaer *et al.*, (2015) combined transactional features with network-based features to effectively detect credit card fraud in an e-commerce setting, where the inclusion of the network features was shown to

have a powerful impact on the prediction results. Similarly, Bonilla (2018) used a variety of network features to detect blockchain-related fraud, with their feature importance process indicating that centrality metrics were strong indicators of fraud.

Degree centrality is the fraction of nodes that a node is connected to within a network and can be summarised as the number of ties that any nodes have to other nodes in a graph (Grando *et al.*, 2018). A higher degree centrality indicates that nodes are well connected within the network, which is not standard behaviour in a financial services context. Users should ideally not be sharing devices, credit cards, or IP addresses, and therefore the more connected a user is, the more likely they are involved in suspicious activity. The second centrality measure used is eigenvector centrality. Eigenvector centrality is similar to degree centrality, although it considers that connections to nodes that are themselves influential contribute more to the centrality score of the node. Therefore, the higher the eigenvector centrality score of a node, the more connected the node is to other nodes which themselves have high scores (Hamilton, 2020). Closeness centrality is the third centrality metric used, measuring the average length of the shortest path between the node in question and all other nodes within a graph (Newman, 2010). The fourth centrality metric is betweenness centrality, which determines the number of times that a node acts as a bridge along the shortest path between two other nodes in the network (Hamilton, 2010). Load centrality, the fifth centrality metric used, is similar to the betweenness centrality, and determines the fraction of all shortest paths that pass through each respective node (Brandes, 2008). The sixth and final centrality metric used, harmonic centrality, is considered to be a practical representation of an average shortest path (Needham and Hodler, 2020). Centrality metrics all measure the connectedness of nodes within a network in different ways, and higher

centrality metrics all tend to indicate a higher likelihood of anomalous behaviour in an anomaly detection context,

In addition to the choice of centrality metrics, various other network metric approaches were also considered. Hamilton (2020) found that the clustering coefficient metric was an important metric for measuring importance within graphs, while Yin *et al.*, (2019) motivated that clustering metrics can be used as powerful outlier detection signals. Link analysis metrics like PageRank were effectively used by Fakhraei *et al.*, (2015), with the clique metric also showing promise as an effective method for determining groups within graphs (Needham and Hodler, 2020). Finally, assortativity metrics were found to be strong indicators of anomalies, according to Piraveenan *et al.*, (2008).

Within the clustering group of metrics, both the clustering coefficient and square clustering coefficient metrics were used. Negro (2021) suggests that the clustering coefficient metric is vital in fraud detection, where nodes with a lower coefficient compared to the global average indicate that the nodes are connected to nodes that belong to independent groups. Being linked to an independent group can be suspicious in some contexts, as it suggests that users might be involved in a coordinated fraudulent syndicate. Similarly, the square clustering coefficient determines the probability that two neighbours of a node share another common neighbour (Lind *et al.*, 2005).

The clique family of network metrics were also considered, as the grouping of nodes has shown to be an essential method for determining the likeness of network participants (O'Malley and Onnela, 2014). A node clique number finds the largest clique containing a node, where a clique is a complete graph where all of the nodes are connected to each other. The number of cliques that a node belongs to is also determined. This metric determines the number of unique cliques that contain each node, which is complementary to the node clique number metric, which only captures the largest clique containing the node (Newman, 2010). The larger the number of cliques or node clique

number per node, the more connected the user is within the network, and ultimately more likely to act suspiciously.

In total, the overall network contains 27,361 nodes and 34,957 edges. Considering the large volume of nodes, the computed features for only the first five nodes in the network are displayed in Table 1. In addition to the network metrics described previously, there is an opportunity to compute custom network metrics using the labelled data. The first two network features listed in Table 1 are reliant on the industry organisation's internal labelling of the users in the data set as being fraudulent or having an account that is in a locked state. A user that is marked as fraudulent would always have a locked account, but a user with a locked account does not always imply that the user is indeed fraudulent. There are a number of operational reasons why a user might be locked, most of which are an indication of a previous negative event linked to the user. The assumptions with these features are that being closely linked to other users with a history of negative behaviour could potentially indicate fraud for the user in question. The proximity of all nodes to known fraudulent or locked nodes is determined by applying Dijkstra's algorithm (Hossain *et al.*, 2020) to determine the shortest path to any anomalous node.

The balance of the features in Table 1 are standard network modelling metrics related to the users' influence within the network of linked users. The network can be defined as the subset of the original industry data set where the underlying users have a connection to another user. Therefore, there is no guarantee that if a user exists in the original industry data that they will appear in the network.

Table 1. Raw network metrics for the first five nodes

Feature	Node 1	Node 2	Node 3	Node 4	Node 5
Distance closest fraud user	0	3	3	1	0
Distance closest locked user	0	1	1	1	0
Degree centrality	0.0001	0.00018	0.00014	0.00025	0.00014
Eigenvector centrality	1.2e-25	6.0e-07	6.0e-07	2.92e-06	2.5e-19
Closeness centrality	0.0000	0.00769	0.00769	0.00917	0.00007
Betweenness centrality	0.0000	0.00000	0.00000	0.00010	0.00000
Load centrality	0.0000	0.00000	0.00000	0.00010	0.00000
Harmonic centrality	0.0000	251.351	250.851	298.618	2.00000
Average neighbour degree	1.0000	5.40000	6.00000	9.57142	3.00000
Clustering coefficient	0.0000	0.33333	1.00000	0.60000	1.00000
Node clique number	1	3	3	4	3
Number of cliques	1	2	1	2	1
Square clustering coefficient	0.0000	0.63636	6.00000	0.07448	0.66666
PageRank	0.00003	0.00004	0.00003	0.000027	0.000037

In Appendix A, the network features are transformed to a $N(0,1)$ distribution via the same standardisation process, and joined to the original industry data set to form the expanded industry data set, including network metric features. The features in Appendix A represent an abstraction of the data following the standardisation process, and is taken from the industry data set including the network metrics. The network metrics set that was joined onto the transactional feature set end up being sparsely populated. This sparsity is due to the relatively smaller number of users in the network data set when compared with the full set of users in the transactional feature set. There are fewer users in the network data set because not all users have a connection to another user. Being connected to another user is somewhat suspicious in itself, and therefore it is expected that a small proportion of the users in the transactional features are represented as nodes in the network metric features data.

The final set of features is now available, including both the transactional features and the network metric features determined in Sections 4.1 and 4.2 respectively.

4.3 Practical experiment

In this section, the combined transactional and network metrics features constructed in Sections 4.1 and 4.2 are used to train an autoencoder model to detect anomalies in the data. The model training process involves applying the unified approach proposed by Ball *et al.*, (2020) and is detailed in Section 3.2. The unified approach starts by simulating 100 candidate autoencoder architectures via a neural architecture search (NAS), where the optimal architecture is selected based on a balanced score comprised of the mean of the average precision (AP), area under the ROC curve (AUC), and normalised Matthews correlation coefficient (MCC) scores.

The NAS process, as denoted by Algorithm 1, found that models trained on the combined transactional and network metrics feature set yielded an optimal architecture balanced score of 94.07%. The results of the NAS process are displayed in Table 2. The optimal architecture had nine hidden layers, with a bottleneck of three neurons. The architecture column indicates the number of neurons in each hidden layer of the autoencoder, ranging from the bottleneck through to the final hidden layer prior to the output layer, essentially representing the decoder component. Symmetric autoencoder architectures are assumed, therefore the encoder component can be considered to be the mirror image of the decoder. The “Param” column notes the number of trainable parameters that were generated, while the “Thre” column lists the optimal threshold for the architecture discovered while optimising for a maximal balanced score.

Table 2: Details of top 20 architectures using the combined transactional and network features

AP	AU	MC	Balanc	Architecture	Param	Thre
0.88	0.99	0.93	0.940	[3, 23, 34, 41, 42]	13402	4.242
0.88	0.99	0.93	0.939	[18, 19, 22, 26, 28, 38,	19607	3.636
0.85	0.99	0.92	0.925	[12, 21, 24, 25, 38, 41]	13101	3.737
0.85	0.99	0.92	0.923	[5, 7, 12, 17, 19, 26, 39]	9372	4.242
0.85	0.99	0.91	0.921	[4, 16, 23, 30, 34, 41, 45]	16695	4.242
0.84	0.99	0.91	0.918	[3, 4, 27, 31, 35, 40, 43]	15994	4.343
0.83	0.99	0.91	0.914	[13, 16, 17, 22, 27, 35]	9898	3.131
0.81	0.99	0.90	0.905	[8, 10, 14, 19, 22, 27, 35]	9479	4.343
0.81	0.99	0.89	0.902	[3, 7, 11, 20, 33, 38, 46]	13946	4.444
0.81	0.99	0.89	0.902	[4, 12, 15, 22, 25, 26, 34]	9765	4.242
0.80	0.99	0.89	0.897	[4, 14, 19, 24, 29]	6723	3.939
0.79	0.99	0.88	0.890	[8, 13, 15, 17, 20, 29, 35]	9575	3.939
0.79	0.99	0.88	0.889	[3, 24, 29, 30, 43]	11418	3.737
0.78	0.99	0.88	0.887	[3, 8, 18, 23, 26, 29, 32,	13956	4.343

0.78	0.99	0.87	0.885	[4, 12, 16, 18, 20, 22, 26,	9725	4.040
0.77	0.99	0.87	0.881	[3, 8, 18, 22, 25, 29, 31,	17392	4.242
0.77	0.98	0.87	0.880	[6, 22, 33, 42]	9873	2.424
0.76	0.99	0.87	0.877	[6, 9, 15, 20, 30, 35, 36]	11547	3.535
0.76	0.99	0.87	0.876	[4, 5, 27, 32, 35, 38, 46]	16477	4.040
0.76	0.99	0.87	0.876	[5, 7, 10, 11, 12, 19, 22,	8598	4.444

Once the optimal architecture has been established, the next step in the unified approach is to determine the optimal classification threshold for the autoencoder. The threshold is required in order to split the reconstruction errors produced by the trained model into the non-anomalous and anomalous groups. In the case of fraud detection, reconstruction errors above the threshold would indicate fraudulent transactions, while reconstruction errors below the threshold are deemed to be non-anomalous. Araya *et al.*, (2016) refer to a threshold as being typically defined by the mean square error (MSE). The optimal threshold for the autoencoder is determined by applying the threshold optimisation algorithm presented as Algorithm 2, where the optimal threshold for the autoencoder model was determined to be a MSE of 5.353.

The optimal threshold for the model is depicted in Fig. 8, where the figure indicates that the peak balanced score of just over 94% on the y-axis corresponds to a MSE threshold of 4.545 on the x-axis. Fig. 9 is a graphical representation of the reconstruction errors, where the bold blue line in the figure is the threshold that attempts to split the non-anomalous and fraudulent transactions

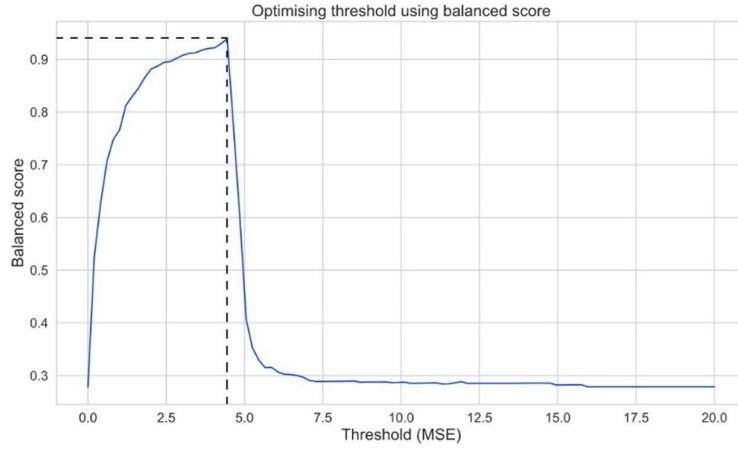


Fig. 8. Optimising combined model threshold selection using the balanced score

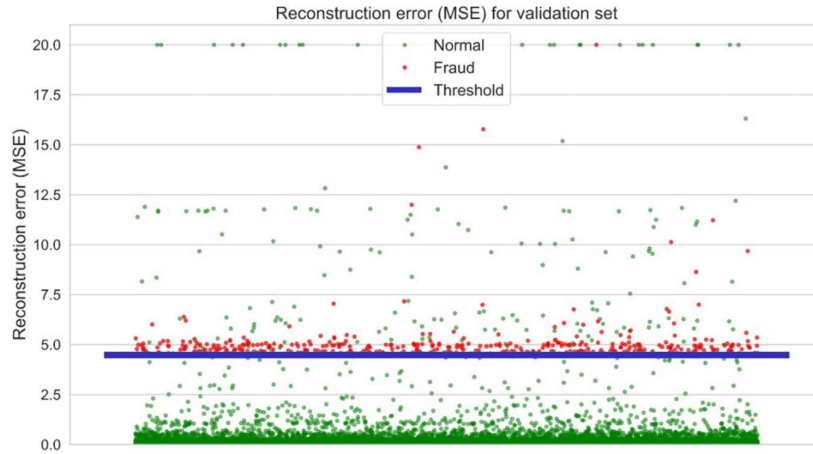


Fig. 9. Reconstruction scores (MSE) using the combined validation data

Following the establishment of the optimal thresholds, the third phase of the unified approach, Gaussian scaling, is applied to the raw anomalies scores, as described by Algorithm 3. This helps with scaling the raw scores into an interpretable $[0;1]$ range, which can be used and easily understood in an operational context. The Gaussian scaled anomaly scores for the model is illustrated by Fig. 10. The green bars indicate the non-anomalous transactions, which are expected to be scored lower in relation to the red bars

indicating the fraudulent transactions. Fig. 10 groups the fraudulent transactions in the 95%+ bucket, whereas the non-anomalous transactions are grouped into various buckets, although the vast majority are in the desired lowest bucket.

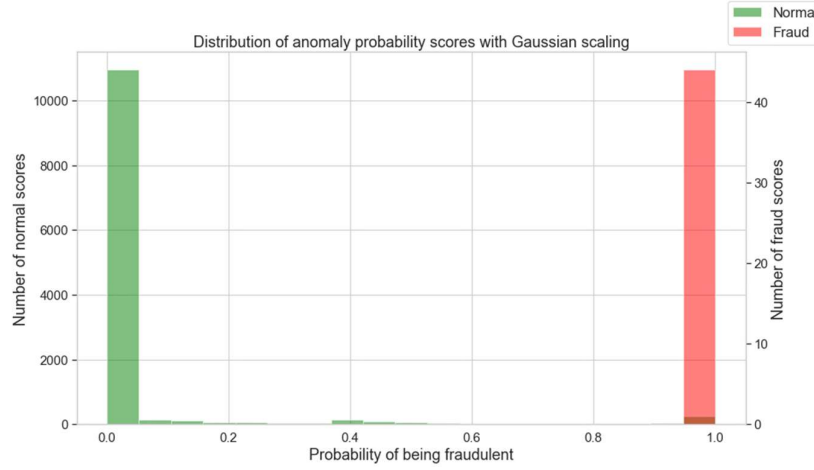


Fig. 10. Anomaly scores with Gaussian scaling

4.4 Experimental results

Table 3 captures the performance of the optimal autoencoder architecture on the feature hold-out test set. This test set contains 12,038 records, of which 44 are confirmed fraud cases. The optimal model correctly classified all 44 instances of fraud while misclassifying 125 non-anomalous transactions as fraudulent.

There are many performance measures that are traditionally used in the evaluation of a model. The most popular measures include average accuracy, average error rate, precision or sensitivity, recall and F_1 -score. In this research, it was decided to use the precision and recall measures as the primary performance indicators. Precision is defined as the fraction of relevant instances among the retrieved instances and, for a binary classification problem, it is mathematically expressed as

$$Precision = \frac{tp}{tp + fp}$$

where tp indicates the number of true positives, or the number of confirmed anomalous instances correctly classified as such, and fp indicates the number of false positives, or the number of confirmed non-anomalous instances that were incorrectly classified as anomalous. The relevant instances in the precision formula are those predicted as anomalous; therefore, precision is the number of correctly classified anomalous instances over the total number of predicted anomalous instances.

Recall pertains to the fraction of total relevant instances that were retrieved and for a binary classification problem is formally defined as

$$Recall = \frac{tp}{tp + fn}$$

where fn is the number of false negatives, or the number of confirmed anomalous instances that are incorrectly classified as non-anomalous. Recall can therefore be considered the number of correctly classified anomalous instances over the total number of confirmed anomalous instances.

The final results obtained was a precision score of 26% and a recall score of 100%. Due to the highly imbalanced nature of the data, it is expected that the model will realise low precision scores. Al-Shabi (2019) confirmed that the balance between precision and recall is a trade-off that needs to be evaluated by the organisation in question, as there are unique costs associated with both types of misclassifications. The trade-off between precision and recall can be quantified by applying a cost-based approach to capture the financial impact of fraud. This could be achieved by attaching a predetermined cost to each quadrant of the confusion matrix (see Table 3) to compute the total cost impact at differing precision-recall thresholds (Bahnsen *et al.*, 2013).

The F_1 -score (the harmonic mean between precision and recall) is also widely acknowledged as a valuable metric in many studies (Schutze *et al.*, 2008). However, it has faced some criticism and Hand and Christen, (2018), for example, criticised F_1 -scores for placing equal importance on precision and recall. This becomes problematic in practice, where different misclassifications carry different cost implications. The argument is important in fraud detection, where it is often more expensive to misclassify a positive instance of fraud as negative than a non-anomalous instance as positive (Sammut and Webb, 2011). The accuracy measure was also not used as a performance measure as it is often not considered an effective performance metric due to the imbalance between the classes. This is known as the accuracy paradox (Valverde-Albacete *et al.*, 2013).

Table 3: Confusion matrix of optimal architecture with combined features

		Predicted class		Total
		Fraud	Not fraud	
True class	Fraud	44	0	44
	Not fraud	125	11869	11994
Total		169	11869	12038

Another important factor in assessing the effectiveness of a trained model is model interpretability. To evaluate the model obtained, it was decided to apply a SHAP analysis to the results. SHAP is a technique for assigning feature importance for model predictions (Lundberg and Lee, 2017), where the contribution of each feature to specific predictions are explained (Molnar, 2019). The technique has also been adapted to the anomaly detection domain, where Antwarg *et al.*, (2021) showed success in applying SHAP on the output scores generated by autoencoder models. Applying SHAP on an autoencoder model output entails explicitly determining the connection between the

features that have a high reconstruction error and those features that are most important in affecting the reconstruction error of the model.

The results of applying SHAP to the model outputs are shown in Fig. 11. The transactional features are depicted by the blue bars, while the red bars highlight the contributions of the network features in descending order of influence. Features *v1* through *v47* indicate the transactional features, whereas features *v48* through *v61* indicate the network metrics features. It is clear that the most important transactional and network features each have an equal share of the top ten features displayed in Fig. 11.

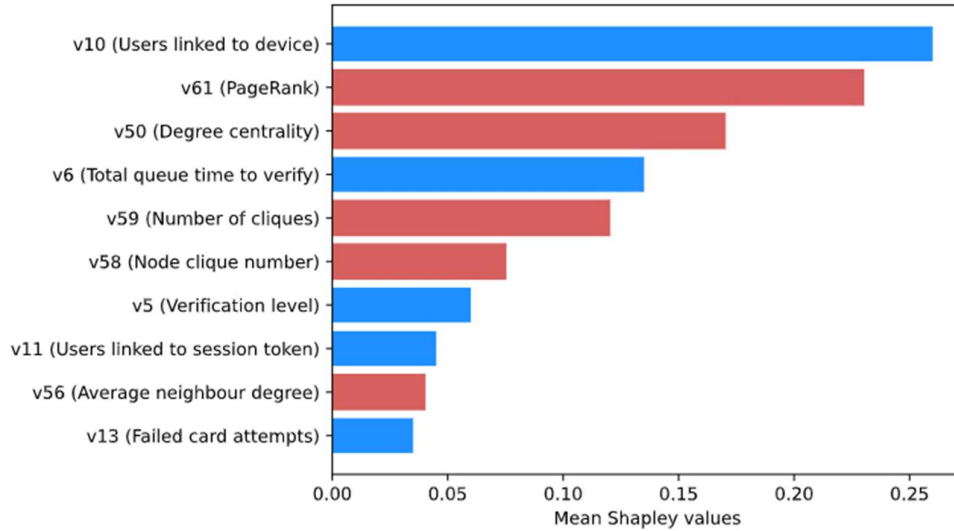


Fig. 11. Shapley values for autoencoder with combined features

As can be seen from Fig 11, the top ten features are comprised of five from the transactional data and five from the newly suggested network metrics. The most important transactional features (indicated in blue in Fig 11) in the final model include the following:

Users linked to device (v10): This feature is the most important of all and indicates whether other users are using the same registered mobile device. In the context of fraud detection, it is important to know whether users share a

registered device because it is assumed that non-anomalous users only use their own devices and do not allow others to transact via their devices.

Total queue time to verify (v6): The second most important transactional feature is v6 which is also ranked fourth among the top ten features. It shows whether a user progressed through the sign-up process swiftly, which may indicate that the user has signed up with multiple fictitious accounts previously. This signifies that the user has gained a substantial amount of experience in completing the sign-up process, which is not considered non-anomalous, where a non-anomalous user should only need to sign-up once. The feature may also detect an automated approach to completing the sign-up process, such as copying and pasting data from a template, which too is considered suspicious behaviour.

Verification level (v5): The third most important transactional feature, verification level, indicates the level of transactional restrictions placed on an account. A higher verification level denotes a lower level of restriction, as more is known about the customer from a KYC (Know Your Customer) perspective. Both non-anomalous and fraudulent users would benefit from having a high verification level so that they are able to transact with relatively large financial amounts.

Users linked via a session token (v11): The fourth most substantial transactional feature, users linked via a session token, measures the number of different users who transacted via a uniquely generated web token. As with users linked to device, the assumption is that non-anomalous users would not transact via the same web token assigned to another user. Web tokens are generated by the organisation, whereas devices are registered using their physical MAC (media access control) address, of which the first is seen to be more accurate in detecting user devices.

Failed card attempts (v13): The fifth and final important transactional feature, failed card attempts, indicates the number of times that a user attempted to load a card that was declined by the issuing bank. A high number of failed card attempts is suspicious behaviour, as it might suggest that a user is testing a number of stolen cards, an activity known as card testing.

The most important network metrics (indicated in red in Fig 11) that were identified by the SHAP analysis and which are proposed to augment the standard transactional variables are as follows:

PageRank (v61): The most important network metrics feature is the PageRank metric, which determines the importance of nodes in a graph based on the structure of the incoming edges. This results in a measure that captures the influence of a node within a graph, where influence is determined as the number of high-quality connections that a given node has to other nodes. High-quality connections are links to nodes which themselves are highly connected, which can be considered suspicious behaviour within a financial services ecosystem.

Degree centrality (v50): The second most important network metric feature is degree centrality. This feature computes how many connections nodes have compared to all other nodes within a network. Therefore, nodes connected to more users than other nodes are more likely to be anomalous. Conversely, it is assumed that non-anomalous users would not be strongly connected to other users in a financial ecosystem such as this.

Number of cliques (v59): Cliques or subgraphs are also strong contributing features, acting as another measure for how connected a node is to large clusters within the graph. The number of cliques determines the number of possible subgraphs that each node is connected to, and is the third most important network metrics feature for detecting anomalies in the data. The

higher the number of cliques, the more connected a user is within the network, which is deemed to be suspicious.

Node clique number (v58): The fourth most important network metrics feature is the node clique number. A larger node clique number indicates that the user is strongly connected to other users and helps to surface anomalous behaviour within the network. The node clique number represents the largest clique containing each given node.

Average neighbour degree (v56): The fifth and final strongly contributing network metrics feature is the average neighbour degree and is simply the average of the degree (number of connections to other nodes) of the neighbours that surround each node. A node that is closely associated with other densely connected nodes would generate a high average neighbour degree, an indication of potentially suspicious behaviour.

The above results indicate that the network metrics features are important contributors to detecting anomalies when using autoencoders. The two most important network metrics features, namely PageRank and degree centrality, make logical sense as both relate to the number of important links to the node within a network, which is important in anomaly detection.

5. Discussion and reflection

In line with the paper's objective, a methodology that combines an anomaly detection autoencoder model with additional network metrics was described in the preceding sections. The results are promising, and the combined feature model showed that adding the proposed network metrics features contributes substantially to detecting anomalies. The method followed in the study is based on a unified approach that is designed to determine a best autoencoder model architecture. The model is then extended by using a combination of transactional data and network metrics, typically found in SNA, to detect anomalous transactions in a real-world financial dataset. The discussion and

reflection in this section will be presented in three parts, *i.e.*, the unified approach to select a "best" model architecture, the selection and inclusion of network metrics to augment the model and general recommendations.

Construction of an anomaly detection autoencoder model using a unified approach

The unified approach used to build a suitable model was described in Section 3.2 and applied in Section 4.3. The methodology consists of 3 main phases, *i.e.*, a neural architecture search, a threshold optimisation and a Gaussian scaling phase.

The proposed unified approach provides several advantages that will help to ensure that a "best" model architecture is used. It is a substantial improvement over the simple grid search algorithm traditionally used as it finds an optimal set of hyperparameters for a model in a relatively short time. The random nature of the neural architecture search outperforms a grid search, especially in cases where the number of possible hyperparameters substantially influence the model's performance. A grid search often suffers from the curse of dimensionality where the number of possible parameters may grow exponentially as the number of hyperparameters increases. These challenges in a traditional grid search, which are addressed with the proposed unified approach, are confirmed in the literature (Bergstra and Bengio, 2012; Liashchynskyi and Liashchynskyi, 2019).

In fraud and anomaly detection problems, the unified approach also provides advantages. The optimal architecture is found by validating various hyperparameter configurations against performance metrics specifically chosen due to their suitability for datasets with imbalanced classes. These imbalanced datasets are often found in fraud-related datasets as the number of fraudulent transactions usually is far less than the valid ones. Three metrics appropriate to the anomaly detection context were chosen *i.e.*, average

precision, the area under the ROC curve and a normalised Mathews correlation coefficient. These metrics were then combined to simplify the underlying metrics into a single balanced score.

A typical problem encountered in anomaly detection is the subjective threshold decision to identify anomalous transactions. This is frequently determined through visual inspection of a potential threshold. The proposed unified approach provides, through its second phase, an advantage over other threshold determination approaches, such as using the maximum mean square error from the training process. The method explicitly determines a threshold that maximises a given objective, namely a balanced score. Finally, the third component of the unified approach provides scaled, universally transferable probability scores as the autoencoder model output. These scores, transformed via Gaussian scaling, remove the friction typically caused by comparing raw anomaly scores from models with different magnitudes of scores. The unified approach chains all of these different aspects of the anomaly detection process together to form a functional pipeline that can be extended for future academic and commercial applications.

Extending and incorporating appropriate network analysis metrics into the model

The network analysis metrics used in the anomaly detection model were described in Sections 3.3 and 4.2.

While the transactional features in this study capture the various transactional behaviours that users within the system exhibit, these features do not capture the relationships between users. Variables that explain relationships are essential in the context of fraud detection and often reveal additional information that may lead to the identification of anomalies. SNA provides an opportunity to transform data that captures the linked nature of relationships between users, effectively representing these relationships as a

mathematical graph. The implications for this representation are twofold. Firstly, graph data allows for visualising the relationships between users in graphical form and secondly, SNA algorithms can be applied to the linked data to determine metrics that describe the connectedness of the users within the entire graph of users. Neither of these is possible without data that captures these relationships between users, therefore specific networked data was included in the unified approach to expand the set of features beyond the transactional level of aggregation.

Including network metrics in the unified approach entails representing the transactional data in graph form. The process of porting the linked data into graph form brings specific challenges. The task of creating the network metric features is important from a computational perspective, where the application of the various graph algorithms on the graph increased the feature engineering time substantially. The run-time of the model training process itself was not impacted meaningfully by the addition of the network metrics features, with training taking slightly longer due to a more extensive training data input space. The same goes for the complexity of the model, as model complexity is determined by the hyperparameters chosen during the neural architecture search process. There is no direct impact on model complexity by merely including network metric features. Another challenge is the required knowledge necessary to implement network metrics. Calculating the network metric features requires specific knowledge on network analysis and how to use it to calculate the network metrics. Should the knowledge be available to perform network analysis, it is recommended to calculate network metric features, specifically those metrics associated with centrality, and include them as additional features when training an autoencoder for anomaly detection in a transactional setting.

The network metrics selected for this study were chosen after consulting related studies that had either used manual network metrics for fraud

detection purposes or used the network metrics as features to train a machine learning model. The autoencoder model trained using additional network metric features produced an output with a low number of false positives, which would lead to a low operational cost of fraud in the industry. By analysing the Shapley values for the trained model, the PageRank and degree centrality metrics showed the most substantial influence in detecting anomalies. These metrics highlight the importance of determining nodes with many links within a given network (in many cases, an indication of a possible anomalous transaction). The identified metrics are therefore helpful for anomaly detection purposes. Applying SHAP to the autoencoder output also proved to be useful as an approach to determine the strongest contributing or offsetting network metric features for any anomaly that may be detected. Another advantage in following this approach is that the method removes the need for investigation analysts to perform manual calculations to understand user relationships and interactions. Ultimately including network metrics in the unified approach brings an additional dimension to the problem of anomaly detection, as user behaviour can be modelled from both a transactional and network perspective. This was also supported by the encouraging model accuracy results, leading to a low operational cost of fraud in the industry.

General recommendations

Considering the results obtained in this study and the above discussion of the methodology, some final comments and recommendations can be summarised as follows.

- In a fraud detection scenario that is based on a transactional setting, it is recommended that an autoencoder neural network model be constructed to identify anomalous transactions. It would be advantageous to utilise the proposed unified approach to select a "best" model architecture.

- To make provision for underlying relationships and interactions of clients, which is a vital part of fraud detection techniques, the inclusion of specific SNA metrics as additional explanatory variables should be considered. This study has shown that centrality metrics are particularly important in detecting anomalies.
- Model performance metrics should be selected with care. In a fraud context, standard accuracy measures such as the average accuracy or F_1 -scores may be misleading. This is due to factors such as the existence of imbalanced datasets and the critical principle of trade-offs (e.g., the cost of false negatives).
- Finally, cognisance should be taken of practical considerations when considering the use of the proposed unified approach and the implementation of social network features. Some of these considerations include model construction complexity (a substantial amount of technical knowledge is required to build and implement the models); time complexity (model selection, model training and network feature engineering may take some time); model complexity (relates to the number of hyperparameters in the model); and prerequisite knowledge required (a good understanding of neural networks – autoencoders in particular – and SNA with the associated metrics are needed).

6. Conclusion

The objective of this paper was to extend a previously proposed unified approach by incorporating network metrics features. These network metrics features were calculated by applying SNA techniques on data capturing the interactions between users within a financial system.

The unified approach was applied on a combined feature set including both transactional and network metrics features. Performance analysis was conducted on the results of the unified approach, in order to determine whether the incorporation of the network metric features had a important

impact on the ability of the autoencoder to detect anomalies in a transactional context.

The performance evaluation indicated that a favourable balanced score was achieved after introducing the network metric features, which assisted in the model reporting a low number of false positives. Applying SHAP to the model outputs showed that five of the network features appeared in the top ten list of contributing features. The most important of these network metric features were the PageRank, degree centrality, and number of cliques features. It was found that the network metrics features were considerably more complex to generate in comparison with the transactional features however, and this complexity needs to be catered for in any real-world application of the approach.

7. Appendix A

The table presented as Table A.1 details the first twenty rows of the network metrics features data. A column index is used instead of a column name to conserve space in the visualisation. While the data appears to be sparsely populated, there is enough information in the network metrics features to help the autoencoder detect anomalous transactions.

Table A.1 First 20 samples of the network metrics features

0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	0	0	0	0	0	0	0	0	0	0	0	0	0.1
0	0	0	0	0	0	0	0	0	0	0	0	0	0.1
0	0	0	0	0	0	0	0	0	0	0	0	0	0.1
0	0	0	0	0	0	0	0	0	0	0	0	0	0.1
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0.3	0.1	0.1	0	0.7	0	0	0	0	1	0.2	0	0.0	0.0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0

0	0	0	0	0	0	0	0	0	0	0	0	0	0.1
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0.1
0.2	0.1	0.1	0	0.7	0	0	0.6	0.2	1	0.1	0	0.0	0.0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0.1
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0.1
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0.1
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0.1
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0.6
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0

0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0.1
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0

8. References

- Al-Shabi MA, 2019, *Credit card fraud detection using autoencoder model in unbalanced datasets*, Journal of Advances in Mathematics and Computer Science, 33(5), pg.1-16.
- An J, and Cho S, 2015, *Variational autoencoder based anomaly detection using reconstruction probability*, Special Lecture on IE, 2(1), pg.1-18.
- Antwarg L, Miller RM, Shapira B, and Rokach L, 2021, *Explaining anomalies detected by autoencoders using Shapley additive explanations*, Expert Systems with Applications, 186, pg.115736.
- Araya DB, Grolinger K, ElYamany HF, Capretz MAM, and Bitsuamlak G, 2016, *Collective contextual anomaly detection framework for smart buildings*, International Joint Conference on Neural Networks, pg.511-518.
- Asha RB, and Suresh Kumar KR, 2021, *Credit card fraud detection using artificial neural network*, Global Transitions Proceedings, 2(1), pg.35-41.
- Bahnsen AC, Stojanovic A, Aouada D, and Ottersten B, 2013, *Cost sensitive credit card fraud detection using Bayes minimum risk*, Machine Learning and Applications (ICMLA), 12(1), pg.333-338.
- Balakrishnan R and Ranganathan K, 2012, *A textbook of graph theory*, Springer Science & Business Media, New York.
- Ball RS, Kruger HA, and Drevin L, 2020, *A unified approach to anomaly detection*, In: Proceedings of the Sixth International Conference on Machine Learning, Optimization, and Data Science., pg.281-291.

- Beigi G, Tang J, and Liu H, 2020, *Social science-guided feature engineering: A novel approach to signed link analysis*, ACM Transactions on Intelligent Systems and Technology, 11(1), pg.1-27.
- Bender EA and Williamson SG, 2010, *Lists, decisions and graphs*, S. Gill Williamson.
- Bergstra J, and Bengio Y, 2012, *Random search for hyper-parameter optimization*, The Journal of Machine Learning Research, 13, pg.281-305.
- Bonilla JP, 2018, *Detection of fraud in financial blockchain-based transactions through big data analytics*, Master's dissertation, Universidad Carlos III de Madrid.
- Boquet G, Morell A, Serrano J, and Vicario JL, 2020, *A variational autoencoder solution for road traffic forecasting systems: missing data imputation, dimension reduction, model selection and anomaly detection*, Transportation Research Part C: Emerging Technologies, 115, pg.102622.
- Borner K, Eide O, Mchedlidze T, Rehbein M, and Scheuermann G, 2019, *Network visualisation in the humanities*, Dagstuhl Reports, 8(11), pg.139-153.
- Brandes U, 2008, *On variants of shortest-path betweenness centrality and their generic computation*, Social Networks, 30(2), pg.136-145.
- Bringmann LF, Elmer T, Epskamp S, Krause RW, Schoch D, Wichers M, Wigman J, and Snippe E, 2019, *What do centrality measures measure in psychological networks*, Journal of Abnormal Psychology, 128(8), pg.892-903.
- Burcher M, and Whelan C, 2018, *Social network analysis as a tool for criminal intelligence: understanding its potential from the perspectives of intelligence analysts*, Trends in Organized Crime (21), pg.278-294.

Camacho D, Panizo-Lledot A, Bello-Orgaz G, Gonzalez-Pardo A, and Cambria E, 2020, *The four dimensions of social network analysis: an overview of research methods, applications, and software tools*, Information Fusion, 63, pg.88-120.

Carcillo F, Borgne Y, Caelen O, Kessaci Y, Oble F, and Bontempi G, 2019, *Combining unsupervised and supervised learning in credit card fraud detection*, Information Sciences, 557, pg.317-331.

Chalapathy R, Khoa NLD, and Chawla S, 2020, *Robust deep learning methods for anomaly detection*, In: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pg.3507-3508.

Chawla A, Jacob P, Lee B, and Fallon S, 2019, *Bidirectional LSTM autoencoder for sequence-based anomaly detection in cyber security*, International Journal of Simulation – Systems, Science & Technology, 20(5), pg.1-6.

Chen J, Li J, Chen W, Wang Y, and Jiang T, 2020, *Anomaly detection for wind turbines based on the reconstruction of condition parameters using stacked denoising autoencoders*, Renewable Energy, 147(1), pg.1469-1480.

Chen T, Liu X, Xia B, Wang W, and Lai Y, 2020, *Unsupervised anomaly detection of industrial robots using sliding-window convolutional variational autoencoder*, IEEE Access, 8, pg.47072-47081.

Chen Z, Yeo CK, Lee BS, and Lau CT, 2018, *Autoencoder-based network anomaly detection*, In: 2018 Wireless Telecommunications Symposium (WTS), pg.1-5.

Chicco D, and Jurman G, 2020, *The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation*, BMC Genomics, 21(1), pg.1-13.

- Chow JK, Su Z, Wu J, Tan PS, Mao X, and Wang YH, 2020, *Anomaly detection of defects on concrete structures with the convolutional autoencoder*, Advanced Engineering Informatics, 45, pg.101105.
- Colladon AF, and Remondi E, 2017, *Using social network analysis to prevent money laundering*, Expert Systems with Applications, 67, pg.49-58.
- Cui Y, Sun Y, Hu J, and Sheng G, 2018, *A convolutional autoencoder method for anomaly detection on system logs*, In: 2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC), pg.3057-3062.
- Dal Pozzolo, A, 2015, *Adaptive machine learning for credit card fraud detection*, PhD thesis, Université Libre De Bruxelles.
- Dau HA, Ciesielski V, and Song A, 2014, *Anomaly detection using replicator neural networks trained on examples of one class*, Simulated Evolution and Learning, 10, pg.311-322.
- Decuypere M, 2020, *Visual network analysis: a qualitative method for researching sociomaterial practice*, Qualitative Research, 20(1), pg.73-90.
- Del Gaudio BL, Porzio C, Sampagnaro G, and Verdoliva V, 2020, *How do mobile, internet and ICT diffusion affect the banking industry? An empirical analysis*, European Management Journal, 39(3), pg.327-332.
- Ding K, Li J, Bhanushali R, and Liu H, 2019, *Deep anomaly detection on attributed networks*, In: Proceedings of the 2019 SIAM International Conference on Data Mining, Society of Industrial and Applied Mathematics, pg.594-602.
- Drummond C, and Holte RC, 2003, *C4.5, class imbalance, and cost sensitivity: why under-sampling beats over-sampling*, In: Workshop on learning from imbalanced datasets II, 11, pg.1-8.
- Fakhraei S, Foulds J, Shashanka M, and Getoor L, 2015, *Collective spammer detection evolving multi-relational social networks*, In: Proceedings of the

12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pg.1769-1778.

Gerych W, Agu E, and Rundensteiner E, 2019, *Classifying depression in imbalanced datasets using an autoencoder-based anomaly detection approach*, In: 2019 IEEE 13th International Conference on Semantic Computing (ICSC), pg.124-127.

Goodfellow I, Bengio Y, and Courville A, 2016, *Deep learning*, MIT Press, Cambridge, Massachusetts.

Grando F, Granville LZ, and Lamb LC, 2018, *Machine learning in network centrality measures: tutorials and outlook*, ACM Computing Surveys, 51(5), pg.1-32.

Guo Y, Liao W, Wang Q, Yu L, Ji T, and Li P, 2018, *Multidimensional time series anomaly detection: a GRU-based Gaussian mixture variational autoencoder approach*, In: Asian Conference on Machine Learning, PMLR, pg.97-112.

Hamilton WL, 2020, *Graph representation learning*, Synthesis Lectures on Artificial Intelligence and Machine Learning, 14(3), pg.1-159.

Hand DJ, and Christen P, 2018, *A note on using the F-measure for evaluating record linkage algorithms*, Statistics and Computing, 28(3), pg.539-547.

He K, Lu Y, and Sclaroff S, 2018, *Local descriptors optimized for average precision*, In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pg.596-605.

Henderson R, 2020, *Using graph databases to detect financial fraud*, Computer Fraud and Security, 7, pg.6-10.

Homayouni H, Ghosh S, Ray I, Gondalia S, Duggan J, and Kahn MG, 2020, *An autocorrelation-based LSTM autoencoder for anomaly detection on time-*

series data, In: 2020 IEEE International Conference on Big Data (Big Data), IEEE, pg.5068-5077.

Howell CR, Su W, Nassel AF, Agne AA, and Cherrington AL, 2020, *Area based stratified random sampling using geospatial technology in a community-based survey*, BMC Public Health, 20(1), pg.1-9.

Ji C, Zou X, Liu S, and Li P, 2020, *ADARC: An anomaly detection algorithm based on relative outlier distance and biseries correlation*, Software: Practice and Experience, 50(11), pg.2065-2081.

Johnson JM, and Khoshgoftaar TM, 2019, *Survey on deep learning with class imbalance*, Journal of Big Data, 6(1), pg.1-54.

Khan S, and Taati B, 2017, *Detecting unseen falls from wearable devices using channel-wise ensemble of autoencoders*, Expert Systems with Applications, 87, pg.280-290.

Kivela M, McGee F, Melancon G, Riche NH, and von Landesberger T, 2019, *Visual analytics of multilayer networks across disciplines*, Dagstuhl Reports, 9(2), pg.1-26.

Kriegel H, Kroger P, Schubert E, and Zimek A, 2011, *Interpreting and unifying outlier scores*, In: Proceedings of the 2011 SIAM International Conference on Data Mining, Mesa, Arizona, pg.13-24.

Kucher K, Fatemi M, and Laitinen M, 2021, *Towards visual sociolinguistic network analysis*, In: Proceedings of the 16th International Joint Conference on Computer Vision, Imaging and Computer Graphics, 3, pg.248-255.

Kumari A, Behera RK, Sahoo KS, Nayyar A, Luhach AK, and Sahoo SP, 2020, *Supervised link prediction using structured-based feature extraction in social network*, Concurrency and Computation: Practice and Experience, pg.1-16.

- Kurka DB, Godoy A, and Von Zuben FJ, 2015, *Online social network analysis: a survey of research applications in computer science*, arXiv preprint arXiv:1504.05655.
- Leite RA, Gschwandtner T, Miksch S, Gstrein E, and Kuntner J, 2018, *Visual analytics for event detection: focusing on fraud*, Visual Informatics, 2(4), pg.198-212.
- Lesouple J, Baudoin C, Spigai M, and Tourneret J, 2021, *Generalized isolation forest for anomaly detection*, Pattern Recognition Letters, 149, pg.109-119.
- Li L, Yan J, Wang H, and Jin Y, 2020, *Anomaly detection of time series with smoothness-inducing sequential variational auto-encoder*, IEEE Transactions on Neural Networks and Learning Systems, 32(3), pg.1177-1191.
- Liang X, Duan F, Bennett I, and Mba D, 2020, *A sparse autoencoder-based unsupervised scheme for pump fault detection and isolation*, Applied Sciences, 10(19), pg.6789.
- Liashchynskyi P, and Liashchynskyi P, 2019, *Grid search, random search, genetic algorithms: a big comparison for NAS*, arXiv preprint arXiv:1912.06059.
- Lind PD, Gonzalez MC, and Herrmann HJ, 2005, *Cycles and clustering in bipartite networks*, Physical Review E, 72(5), pg.056127.
- Lindemann B, Maschler B, Sahlab N, and Weyrich M, 2021, *A survey on anomaly detection for technical systems using LSTM networks*, Computers in Industry, 131, pg.103498.
- Liu F, Ting KM, and Zhou Z, 2012, *Isolation-based anomaly detection*, ACM Transactions on Knowledge Discovery from Data, 6(1), pg.1-39.

- Lookman S, and Nurcan S, 2015, *A framework for occupational fraud detection by social network analysis*, In: CAISE 2015 Forum, 1367.
- Lu W, Cheng Y, Xiao C, Chang S, Huang S, Liang B, and Huang T, 2017, *Unsupervised sequential outlier detection with deep architectures*, IEEE Transactions on Image Processing, 26(9), pg.4321-4330.
- Lundberg SM and Lee S, 2017, *A unified approach to interpreting model predictions*, Advances in Neural Information Processing Systems, 30, pg.4765-474.
- Makki S, Assaghir Z, Taher Y, Haque R, Hacid M, and Zeineddine H, 2019, *An experimental study with imbalanced classification approaches for credit card fraud detection*, IEEE Access: Advanced Software and Data Engineering for Secure Societies, 7, pg.93010-93022.
- Schutze H, Manning CD, and Raghavan P, 2008, *Introduction to information retrieval*, Cambridge: Cambridge University Press, 39, pg.234-265.
- Mehrotra KG, Mohan CK, and Huang H, 2017, *Anomaly detection principles and algorithms*, Springer International Publishing, New York.
- Meidianingsih Q, Erfiani, and Sartono B, 2017, *The study of safe-level SMOTE method in unbalanced data classification*, International Journal of Scientific & Engineering Research, 8(5), pg.1167-1171.
- Mienye ID, Sun Y, and Wang Z, 2020, *Improved sparse autoencoder based artificial neural network approach for prediction of heart disease*, Informatics in Medicine Unlocked, 18, pg.100307.
- Misra S, Thakur S, Ghosh M, and Saha SK, 2020, *An autoencoder based model for detecting fraudulent credit card transaction*, Procedia Computer Science, 167, pg.254-262.

- Mohamed S, Ejballi R, and Zaied M, 2019, *Denoising autoencoder with dropout-based network anomaly detection*, In: The 14th International Conference on Software Engineering Advances, pg.98-103.
- Molnar C, 2019, *Interpretable machine learning*, <https://christophm.github.io/interpretable-ml-book/>, 20 December 2021.
- Narayana K, Venkata K, and Prasad PVGD, 2021, *A hybrid intrusion detection system based on sparse autoencoder and deep neural network*, Computer Communications, 180, pg.77-88.
- Needham M, and Hodler AE, 2020, *Graph algorithms*, 3, O'Reilly Media, Inc, United States of America.
- Newman MEJ, 2010, *Networks: an introduction*, 2, Oxford University Press, Oxford, United Kingdom.
- Nguyen HD, Tran KP, Thomassey S, and Hamad M, 2021, *Forecasting and anomaly detection approaches using LSTM and LSTM autoencoder techniques with the applications in supply chain management*, International Journal of Information Management, 57, pg.102282.
- Nikkel B, 2020, *Fintech forensics: Criminal investigation and digital evidence in financial technologies*, Forensic Science International: Digital Investigation, 33, pg.200908.
- Nonnenmacher J, Kruse F, Schumann G, and Gomez JM, 2021, *Using autoencoders for data-driven analysis in internal auditing*, In: Proceedings of the 54th Hawaii International Conference on System Sciences, pg.5748.
- O'Malley AJ, and Onnela J, 2014, *Topics in social network analysis and network science*, arXiv preprint arXiv:1404.0067.
- Oskarsdottir M, Ahmed W, Antonio K, Baesens B, Dendievel R, Donas T, and Reynkens T, 2021, *Social network analysis for supervised fraud detection*

in insurance, Risk Analysis: an Official Publication of the Society for Risk Analysis.

Pidhorskyi S, Adjero DA, and Doretto G, 2020, *Adversarial latent autoencoders*, In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pg.14104-14113.

Piraveenan M, Prokopenko M, and Zomaya AY, 2008, *Local assortativeness in scale-free networks*, Europhysics Letters, 84(2), pg.28002.

Pourhabibi T, Ong KL, Kam BH, and Boo YL, 2020, *Fraud detection: a systematic literature review of graph-based anomaly detection approaches*, Decision Support Systems, 133, pg.113303.

Preethi D, and Khare N, 2021, *Sparse autoencoder drive support vector regression based deep learning model for predicting network intrusions*, Peer-to-Peer Networking and Applications, 14, pg.2419-2429.

Principi E, Rossetti D, Squartini S, and Piazza F, *Unsupervised electric motor fault detection by using deep autoencoders*, IEEE/CAA Journal of Automatica Sinica, 6(2), pg.441-451.

Ren H, Ye Z, and Li Z, 2017, *Anomaly detection based on a dynamic Markov model*, Information Sciences, 411, pg.52-65.

Riquelme F, Gonzalez-Cantergiani P, Molinero X, and Serna M, 2018, *Centrality measures in social networks based on linear threshold model*, Knowledge-Based Systems, 140, pg.92-102.

Sabetti L, and Heijmans R, 2021, *Shallow or deep? Training an autoencoder to detect anomalous flows in a retail payment system*, Latin American Journal of Central Banking, 2(2), pg.100031.

Saia R, Carta S, Recupero DR, and Fenu G, 2019, *Fraud detection for e-commerce transactions by employing a prudential multiple consensus model*, Journal of Information Security and Applications, 46, pg.13-22.

- Said Elsayed M, Le-Khac N, Dev S, and Jurcut AD, *Network anomaly detection using LSTM based autoencoder*, In: Proceedings of the 16th ACM Symposium on QoS and Security for Wireless and Mobile Networks, pg. 37-45.
- Saito T, and Rehmsmeier M, 2015, *The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets*, PLoS ONE, 10(3), pg.118432.
- Salahuddin M, Bari MDF, Alameddine H, Pourahmadi V, and Boutaba R, 2020, *Time-based anomaly detection using autoencoder*, In: 2020 16th International Conference on Network and Service Management (CNSM), IEEE, pg.1-9.
- Sammut C, and Webb G, 2011, *Encyclopedia of machine learning*, Springer Science & Business Media, New York.
- Sanchez OR, Repello M, Carrega A, and Bolla R, 2021, *Evaluating ML-based DDoS detection with grid search hyperparameter optimization*, In: 2021 IEEE 7th International Conference on Network Softwarization (NetSoft), pg.402-408.
- Sanni L and Nurcan S, 2015, *A framework for occupational fraud detection by social network analysis*, CAISE Forum, CEUR, 1367(29), pg.1-8.
- Sanvido PHM, Kurtz GB, Teixeira CRG, Wagner PP, Leuck LP, Silveira MS, Tietzmann R, and Manssour IH, 2021, *PeakVis: a visual analysis tool for social data and video broadcasts*, In: 2021 IEEE 45th Annual Computers, Software, and Applications Conference (COMPSAC), pg.418-427.
- Schreyer M, Sattarov T, Borth D, Dengel A, and Reimer B, 2018, *Detection of anomalies in large-scale accounting data using deep autoencoder networks*, arXiv preprint arXiv:1709.05254.

Skare M and Soriano DB, 2021, *How globalization is changing digital technology adoption: An international perspective*, Journal of Innovation and Knowledge, 6(4), pg.222-233.

Tun MT, Nyaung DE, and Phyu MP, 2020, *Network anomaly detection using threshold-based sparse*, In: Proceedings of the 11th International Conference on Advances in Information Technology, 21, pg.1-8.

Valverde-Albacete FJ, Carrillo de Albornoz J, and Pelaez-Moreno C, 2013, *A proposal for new evaluation metrics and result visualization technique for sentiment analysis tasks*, In: International Conference of the Cross-Language Evaluation Forum for European Languages, Springer, Berlin, Heidelberg, pg.41-52.

Van Vlasselaer V, Bravo C, Caelen O, Eliassi-Rad T, Akoglu L, Snoeck M and Baesens B, 2015, *APATE: A novel approach for automated credit card transaction fraud detection using network-based extensions*, Decision Support Systems, 75, pg.38-48.

Wang S, Wang X, Zhang L, and Zhong Y, 2021, *Auto-AD: Autonomous hyperspectral anomaly detection network based on fully convolutional autoencoder*, IEEE Transactions on Geoscience and Remote Sensing, 60, pg.1-14.

Wu P, Liu J, and Shen F, 2019, *A deep one-class neural network for anomalous event detection in complex scenes*, IEEE Transactions on Neural Networks and Learning Systems, 31(7), pg.2609-2622.

Xie L, Pi D, Zhang X, Chen J, Luo Y, and Yu W, 2021, *Graph neural network approach for anomaly detection*, Measurement, 180, pg.109546.

Yin H, Benson AR, and Leskovec J, 2019, *The local closure coefficient: a new perspective on network clustering*, In: Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining, 12, pg.303-311.

Zandian ZK, and Keyvanpour MR, 2019, *Feature extraction method based on social network analysis*, Applied Artificial Intelligence, 33(8), pg.669-688.

Zavrak S and Iskefiyeli M, 2020, *Anomaly-based intrusion detection from network flow features using variational autoencoder*, IEEE Access, 8, pg.108346-108358.

Zhang J and Luo Y, 2017, *Degree centrality, betweenness centrality, and closeness centrality in social network*, In: Proceedings of the 2017 2nd International Conference on Modelling, Simulation and Applied Mathematics, 132, pg.300-303.

Zhou Y, Ren H, Li Z, and Pedrycz W, 2021, *An anomaly detection framework for time series data: An interval-based approach*, Knowledge-Based Systems, 228, pg.107153.

Appendix E

Research output 4

Appendix E contains the conference abstract titled "Self-organising maps for feature interpretation in anomaly detection", which was based on Chapter 7, and presented at the *51st Annual Conference of the Operations Research Society of South Africa*, Cape Town, South Africa, 12-14 September 2022.

Self-organising maps for feature interpretation in anomaly detection

Richard Ball, Hennie Kruger, Lynette Drevin

Computer Sciences and Information Systems, North-West University

Anomaly detection is a machine learning approach for identifying observations that differ significantly from the majority of the observations within a data set. An important application of anomaly detection is fraud detection, where various techniques are applied in order to detect financial fraud. One of these approaches, autoencoders, are powerful neural networks that excel in detecting anomalies in unlabelled data. In this paper, an autoencoder was developed to detect anomalies in a transactional system. The model was trained using a real-world transactional data set sourced from an organisation in the financial services industry. The autoencoder was extended by incorporating social network analysis metrics as training features, including metrics such as PageRank and degree centrality. Applying SHAP to the output of the autoencoder showed that both transactional and network metrics features contribute significantly to the detection of anomalies. Self-organising maps are then introduced in order to understand the nature of the anomalies in more detail. Various visual analysis techniques are employed, including node activation frequency, class distribution, and class proportion methods, as well as u-matrix and component plane analysis. Component plane analysis proved to be a valuable technique for determining feature importance as well as for detecting anomalous subgroupings in the data. The findings from the component plane analysis confirmed the results of applying SHAP on the autoencoder output for determining feature importance for the anomalies detected. The self-organising map produced feature contributions that overlapped significantly with those determined by the SHAP approach, for a fraction of the computational effort. Future work will entail combining the autoencoder and self-organising maps into a consolidated decision-support system for detecting anomalies in a transactional setting.

Keywords: Anomaly detection, Autoencoder, Self-organising maps and neural networks. *Status:* Complete results

Appendix F

Language editing declaration

This serves to confirm that I, Isabella Johanna Swart, registered with and accredited as professional translator by the South African Translators' Institute, registration number 1001128, language edited the following thesis:

A machine learning-based framework for anomaly detection

by

RS Ball

A handwritten signature in black ink, appearing to read 'ISwart', with a large, stylized initial 'I'.

Dr Isabel J Swart

Date: 7 July 2022

23 Poinsettia Close
Van der Stel Park
Dormehlsdrift
GEORGE
6529
Tel: (044) 873 0111
Cell: 082 718 4210
e-mail: isaswart@telkomsa.net