An alternative trust calculation method using radial basis function neural networks

S Zacaria

orcid.org/0000-0002-3514-0316

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 JV du Toit

Co-promoter: Prof LM Venter

Graduation May 2023

25380249

ACKNOWLEDGEMENT

First and above all, I praise God, the Almighty, for providing me with this opportunity and granting me the ability to succeed. This research appears in its current form due to the assistance and guidance of several people. I would therefore like to offer my sincere thanks to all of them. In the first place, I would like to thank Prof Lucas Venter and Prof Tiny du Toit for their continuous guidance, support, and inspiration throughout my years at the university. I would not have been able to make it this far without them. Furthermore, I would like to thank my family for their encouragement and support. It is difficult to find the correct words to express my deepest thanks to my parents, brother, husband, and my children. Without their unconditional love and loyal support, this study would not have been possible.

LIST OF ABBREVIATIONS

AI	Artificial intelligence
ANN	Artificial neural network
API	Application program interface
ARDS	Amazon relational database service
ASSS	Amazon Simple Storage Service
CoMSER	Content Modelling for Synthetic E-Health Records
DARPA	Defence Advanced Research Project Agency
DDN PMO	Defence Data Network Program Management Office
GCD	Google Cloud Datastore
GCS	Google Cloud Storage
HMM	Hidden Markov Model
IMP	Interface message processor
IP	Internet Protocol
LCT	Lightweight Cross-domain Trust
MUP	Monthly uptime percentage
NCP	Network control program
P2P	peer-to-peer
PKI	Public Key Infrastructure
PSTDG	Pure synthetic trust data set generation
QoS	Quality of service
RBFNN	Radial basis function neural network
SATNAC	Southern Africa Telecommunication Networks and Applications Conference
SDG	Synthetic data generation
SLA	Service level agreement
SULTAN	Simple Universal Logic-oriented Trust Analysis Notation
TCP	Transmission Control Protocol
TMFM	Trust model based on fuzzy mathematics
TNM	Trust Necessitated through Metrics
UCLA	University of California Los Angeles
UCSB	University of California Santa Barbara
W3C	World Wide Web Consortium

ABSTRACT

One of the most challenging problems in an electronic environment is the trust between electronic entities. Trust is a generic concept and can be effectively used in various contexts. An important question is: “How does an electronic entity trust another electronic entity?” or “How can trust be determined?” The most important challenges identified in trust calculation are increased calculation complexity, data storage and access to large data sets for trust calculation. Most of the trust calculations are in the security context, and each electronic entity has its features and standards.

A radial basis function neural network (RBFNN) is a feed-forward neural network used as a universal function approximator to solve nonlinear problems. Training an RBFNN to model trust values requires accurate, large training data sets. Insufficient trust data sets were found due to privacy and data storage problems. The increased calculation complexity and the data storage problem may be solved using an RBFNN. In addition, a possible solution to trust data scarcity is synthetic data generation.

The primary purpose of this study is to find an alternative trust calculation method using RBFNNs. To achieve this, literature regarding different forms of trust calculation is considered, including identifying the three dimensions of trust that leads to a new definition for the term trust. The three dimensions of trust specified are the trust context, calculations for the quantification of trust, and information sources. Challenges in calculating trust in an electronic environment are investigated. Obstacles in using an RBFNN for the calculation of trust are identified. Literature regarding the creation of synthetic trust data is reviewed, and the challenges in generating synthetic trust data are addressed by a seven-step framework called the PSTDG (Pure synthetic trust data set generation) Framework. Consequently, a new definition for the term validation of generated trust data is developed. In addition, a three-step method is created to validate the data generation model. Finally, a four-step experimental design process is developed to build a model using an RBFNN that can determine trust values between electronic entities.

The four-step experimental design process was demonstrated by performing two experiments. In the first demonstration, a data generation model called the PSTDG-PeerTrust model based on PeerTrust, a purely theoretical trust calculation model, is developed. The PSTDG-PeerTrust model is validated, and a trust data set is generated. An RBFNN model called PeerTrustRBFNN is then built, using the best model hyperparameters found. In the second demonstration, the Amazon Relational Database Service (ARDS) shows that real-life problems can also be solved

using the proposed method. Hence, a data generation model called PSTDG-ARDS is developed. This model is validated, and a trust data set is generated. The ARDSTrustRBFNN trust model is built, using the identified best model hyperparameters. The study shows that the data generation models developed using the PSTDG Framework can produce valid pure synthetic trust data, and an RBFNN can be used to calculate trust in an electronic environment.

Keywords: Data generation, Electronic trust, Neural network, Peer-to-Peer trust, Radial basis function neural network, RBFNN, Synthetic trust data, Trust

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1 INTRODUCTION

1.1 Introduction

The term *trust* is a generic concept, and it can be used in various contexts. The word trust has increasing significance in this digital age where electronic entities, especially software entities, play a major role in many parts of our lives. The use of electronic commerce, internet-based access to information, interpersonal communication through the internet, and peer-to-peer distributed computing are increasing day by day. However, there are concerns regarding the trustworthiness of each of these electronic systems. Trust is an important factor in forecasting the behaviour of any electronic or software system. Interactions will be more reliable if they are based on trust (Yong-sheng & Ying, 2010). Therefore, the trustworthiness of an electronic entity is critical, especially as the internet plays an increasingly large role in all aspects of human existence. It is growing increasingly relevant as we become more reliant on software entities.

Many researchers define trust (Gambetta, 1988; Grandison & Sloman, 2002a; Grandison & Sloman, 2002b; Olmedilla *et al.*, 2006). Each of these definitions is relevant only in the context under which it is defined. All these definitions assume that trust can be quantified in some manner. Hence, trust can be calculated. Unfortunately, no widely accepted definition of trust could be found in the fields of computer science and information technology, hence there is a need for a definition for trust which can be used in general. As the trust relationship depends upon the ability to perform an action within a specific context, trust is not symmetric; in other words, entity A's trust in entity B is not the same as entity B's trust in entity A. By the same token, entity A's trust in entity B in one context is not the same as entity A's trust in entity B in another context. Consequently, trust can be non-transitive.

All of the available definitions state that trust is context-dependent. Information security, service delivery, reliability, and credibility are examples of trust contexts (Gambetta, 1988; Grandison & Sloman, 2002a; Grandison & Sloman, 2002b; Olmedilla *et al.*, 2006). Most of the available trust calculations focus only on the information security context. However, the specific trust value of an entity will be different in diverse contexts. For example, entity A's trust in entity B in the context of service delivery is not the same as entity A's trust in entity B in the context of credibility.

To calculate the trust value, some input information is required. The information required to calculate the trust value can be obtained from a variety of sources, such as recommendations, previous experience, feedback, and the presence of World Wide Web Consortium (W3C) methods (Au *et al.*, 2001; Denning, 1993; Gambetta, 1988; Hang *et al.*, 2012; Mui *et al.*, 2002;

Wang & Vassileva, 2003; Witkowski & Pitt, 2000). Several existing trust models that quantify trust values can be found in the literature (Aberer & Despotovic, 2001; Daskapan *et al.*, 2008; Li & Ling, 2002, 2004; Singal & Kohli, 2016; Tahta *et al.*, 2015). These models use different architectures for accessing information required for calculating and managing the trust values. The complexity of the calculation of trust value increases as the context complexity increases. The complexity of the calculation of trust value also increases when the complexity of the electronic entity networks increases. In a distributed system, some entities may not have direct knowledge or previous transaction experience with other entities. Some trust calculation algorithms use the previous trust value of the entity to calculate the new trust values. All the previous transaction details and trust values of all entities in the network must be stored and accessed each time to calculate the trust value of each entity with regard to other entities. The fact that the trust value of an entity in one context, for example the information security context, can be different in another context, for example the service delivery context, also increases the complexity of the calculation. Consequently, there are several complications regarding the calculation of trust values and the management of trust data. The calculation of trust can furthermore be complex due to information access, data storage, data management, network complexity, and context complexity. In Chapter 2, a more detailed literature review of trust is provided.

There is a need for a broad definition of trust which can be used in general. Moreover, trust is context-dependent and there is also a need to determine a method for quantifying trust values in any context. The identified problems in trust calculation includes the following: the complexity of trust value quantification increases as the context complexity increases or when the complexity of the electronic entity networks increases, and trust value quantification and management are complex due to trust data storage and access. Some of these problems may be solved by an alternative approach to calculate trust values which will be explained in the next section.

1.1.1 Radial basis function neural networks

Artificial neural networks (ANNs) provide a new efficient and powerful modelling tool to represent systems having nonlinear and complex input-output relationships (Abiodun *et al.*, 2018; Baskaran *et al.*, 2008; Delon *et al.*, 2007; Mohanasundaram, 2020; Noor *et al.*, 2010). One of the major features of ANNs is its ability to generalise (McCullagh & Bluff, 1993). An ANN can be considered a black box that has the ability to transform an input vector of *m* dimensional space to an output vector of *n* dimensional space (Vega-Corona *et al.*, 2008). Therefore, an ANN is context-independent. ANNs are capable of handling incomplete data and noise, and can therefore be said to be fault-tolerant. Moreover, there are no assumptions regarding the data

properties or data distributions (Abiodun *et al.*, 2018). A radial basis function neural network (RBFNN) is a feed-forward neural network with a relatively simple structure (Ruslan *et al.*, 2013). RBFNNs have been widely used as a universal function approximator (Chien-Cheng *et al.*, 1999; Lin & Wu, 2011; Ruslan *et al.*, 2013). An RBFNN has several advantages, such as a faster learning algorithm, support of incremental training and its approximation capability that is higher than some other artificial neural networks (Azmi *et al.*, 2011; Chen *et al.*, 1991; Lin & Wu, 2011; Park & Sandberg, 1991; Qasem *et al.*, 2013; Yu *et al.*, 2011). An RBFNN is a feed-forward, three-layered neural network. The three layers are the input layer, hidden layer, and output layer (Khazaei *et al.*, 2017). An RBFNN performs a non-linear transformation over the input vectors before the input vectors are classified. Hence, the RBFNN can convert a non-separable linear problem into a linearly separable problem. The calculation of trust values is a non-separable linear problem. Since an RBFNN can solve complex pattern classification problems, it may be used in trust value classification. A comprehensive review of the RBFNN is provided in Chapter 3. To train an RBFNN to calculate trust values between electronic entities in randomly varying situations requires large trust data sets. Synthetic trust data needs to be generated as a result of the scarcity of real-world trust data due to privacy concerns, time span in managing and accessing the trust data, and cost. Thus, a brief introduction to synthetic data is given in the next section.

1.1.2 Synthetic data

Traditional trust algorithms simulate data to calculate trust as the data sources were not readily available (Aberer & Despotovic, 2001; Li & Ling, 2002, 2004; Wang & Vassileva, 2003). The scarcity of real-world training data can be solved by generating synthetic data programmatically (Anderson *et al.*, 2014; Weston *et al.*, 2015). Synthetic data have been successfully applied across a wide range of scientific fields. Synthetic data can be classified into fully synthetic data, partially synthetic data, hybrid synthetic data, and pure synthetic data (Drechsler *et al.*, 2007; Little & Liu, 2003; McLachlan *et al.*, 2016; Reiter, 2004; Rubin, 1993; Surendra & Mohan, 2017). The selection of the type of synthetic data depends upon the needs or the context (Hawala, 2008). Hence, a synthetic trust data set can be generated successfully.

An algorithm or a program used for the generation of synthetic data is called a data generation model. Using inaccurate synthetic data can lead to the development of inaccurate models. A synthetic data generation model must be able to generate a data set that can represent a real-world data set. The data generation model must be able to generate synthetic data that is sufficiently accurate for its intended use. The quality of the generated synthetic data must be reasonable. All the definitions for the term trust assume that there is some information available regarding the entities where trust can be quantified in some manner. Synthetic trust data can be

generated and can be validated, using the information available regarding the entities for which a trust relationship is calculated. A more detailed description of data generation is given in Chapter 4.

The remainder of the chapter is organised as follows. In Section 1.2, the problem statement will be presented. The proposed solution and objectives of this research will be provided in Section 1.3. The research methodology used in this research will be discussed in Section 1.4. In Section 1.5, an overview of this thesis will be considered. The chapter will be concluded in Section 1.6.

1.2 Problem statement

In this study, the challenges in calculating trust values between electronic entities, the challenges in synthetic data generation and synthetic data validation will be investigated. All these challenges were described in Section 1.1. The calculation of trust is complex. The complexity in calculation of trust can be due to information access, data storage, data management, network complexity and context complexity. The RBFNN has been used in a wide variety of application domains due to its favourable properties mentioned in Section 1.1.1. However, to build a model using an RBFNN, large training data sets need to be obtained. These data sets can be generated synthetically, as indicated in Section 1.1.2. Inaccurate synthetic data can result in the development of an inaccurate RBFNN model. As indicated in Section 1.1.2, the generated synthetic data can be validated, using the available information regarding the entities for which a trust relationship is calculated. The possibility of developing an alternative trust calculation method using an RBFNN will be explored to overcome the challenges in calculating the trust values.

The research question investigated in this study can be stated as:

"How can an RBFNN be used to calculate trust values between electronic entities?"

To address this research question, the proposed solution and the objectives of this study are discussed in the next section.

1.3 Proposed solution and objectives

To answer the research question, a possible alternative trust calculation method using an RBFNN will be developed. To achieve this main aim, two research objectives have been identified for this study. The first objective, which is given below, is derived from the discussion about the scarcity of a trust data set in Sections 1.1.2 and 1.2.

Objective 1: Data generation: Provide a generic framework for valid pure synthetic trust data generation.

Once the first objective is successfully achieved, the problem of not having a large trust data set is solved. The construction of an RBFNN trust model will be done next. Thus, the second objective can be given as:

Objective 2: Construction of an RBFNN trust model: Provide a suitable theoretical experimental design process for developing an RBFNN-based trust calculation model.

To accomplish the above two research objectives, the following steps will be followed:

Step 1: Study different literature regarding trust calculation.
Step 2: Investigate the challenges in calculating trust in an electronic environment.
Step 3: Study different trust calculating algorithms and gather information sources and the features offered by several products as examples.
Step 4: Study the RBFNN.
Step 5: Investigate the challenges in using an RBFNN.
Step 6: Investigate the challenges in generating a valid trust data set.
Step 7: Achieve **Objective 1** and **Objective 2**.

To conclude, an alternative trust calculation method using an RBFNN is proposed and no comparative study with any other trust calculating methods will be given in this study.

1.4 Research methodology

The research methodology used for this research consists of identifying the problems in trust calculation in various contexts of electronic environments, gathering research on the problems identified, designing a framework for generating a valid pure synthetic trust data set, developing a theoretical experimental design process for developing an RBFNN-based trust calculation model and its demonstration, using two experiments. The first demonstration will be using a theoretical trust calculation model and the second demonstration will be with a real-world problem.

1.4.1 Problem identification

The first step is to identify the challenges in trust calculation between electronic entities and challenges in training an RBFNN. Thereafter it must be determined why these problems need to be addressed. This includes determining the objectives and scope of the research.

1.4.2 Literature study

The literature study was done on the following topics for this research.

- Trust in general
- Radial basis function neural network
- Data generation

1.4.3 Design

After an extensive literature study regarding trust, RBFNNs and data generation, the PSTDG (Pure synthetic trust data set generation) Framework and a four-step experimental design process for developing an RBFNN-based trust calculation model are designed.

1.4.4 Experimental design, development, and validation

A generic framework for generating valid pure synthetic trust data (PSTDG Framework) is developed. Then a four-step theoretical experimental design process for developing an RBFNN-based trust calculation model is designed. The demonstration is carried out by using two experiments: the first with a theoretical trust calculation model and the second with a real-world problem. In both experiments, a PSTDG model is developed. The PSTDG model is validated using experiments. A pure synthetic trust data set is generated, using the validated PSTDG model. The proposed RBFNN model for trust calculation is developed, using the generated data set after identifying the best RBFNN model by experiments. Finally, the developed RBFNN model for trust calculation is evaluated. All the relevant flow diagrams regarding the PSTDG model were also created.

1.5 Research overview

In this section, a brief discussion as to what can be expected in each chapter is provided. The signpost diagram used throughout this study is given in Figure 1-1.

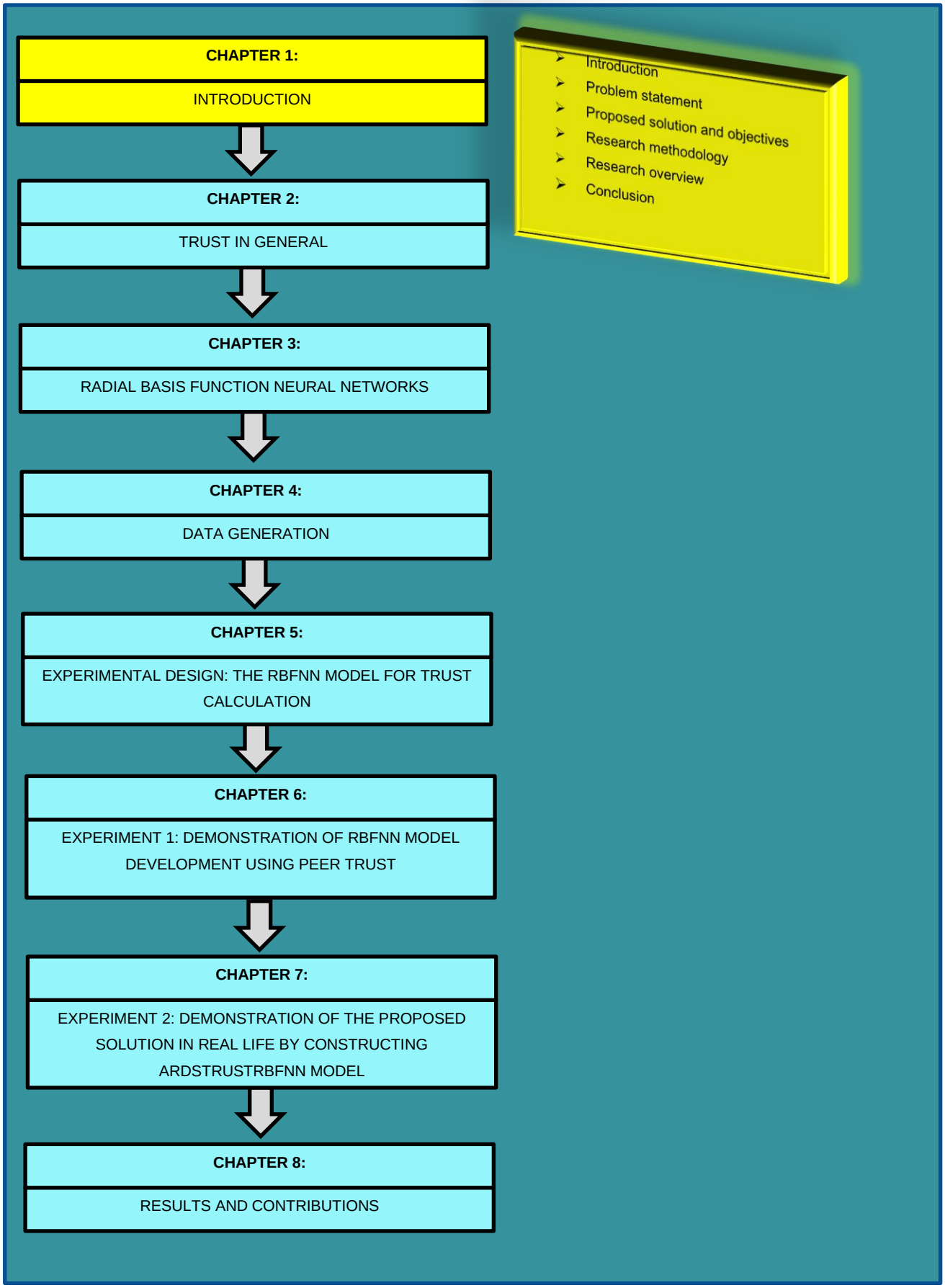


Figure 1-1: Signpost diagram of Chapter 1

This study will be divided up into the chapters described below and follow the sequence as given:

Chapter 1: INTRODUCTION

In this chapter, the research is introduced. An overview of the research, including the problem statement, proposed solution and objectives, research methodology and research overview are presented.

Chapter 2: TRUST IN GENERAL

In this chapter, a literature study on all aspects of trust calculation in electronic environments is undertaken. This will carried out in order to obtain an in-depth understanding of the aspects of trust calculation between electronic entities. A literature study will be done to determine various available definitions of trust, the three dimensions of trust by finding the commonalities and differences between various definitions, a new comprehensive definition of trust, using the three dimensions of trust, examples of trust algorithms, the three dimensions of trust in a fixed context and the three dimensions of trust in a different context.

Chapter 3: RADIAL BASIS FUNCTION NEURAL NETWORKS

In this chapter, a literature study on RBFNN is carried out. The focus will be on the artificial neural network history, the architecture of the RBFNN, the advantages and disadvantages of RBFNNs, non-separable linearity of trust value, the training of the RBFNN and time series prediction, using an RBFNN.

Chapter 4: DATA GENERATION

In this chapter, a literature investigation on problems in trust data collection is covered. A discussion on synthetic data, a generic approach to synthetic data generation, and explanations of synthetic data generation and the validity of the data set for the trust calculation will be provided. In addition, a framework for generating a valid pure synthetic trust data set will be proposed.

Chapter 5: EXPERIMENTAL DESIGN: THE RBFNN MODEL FOR TRUST CALCULATION

A description of the theoretical experimental design to develop an RBFNN model for trust calculation will be given. The implementation details of the PSTDG model development, the validation of the PSTDG model, pure synthetic data set generation, using the validated PSTDG model, and the construction of the RBFNN model, using the generated data, will be presented.

Chapter 6: EXPERIMENT 1: DEMONSTRATION OF RBFNN MODEL DEVELOPMENT USING PEER TRUST

The demonstration of the development of the best RBFNN model called PeerTrustRBFNN for trust calculation, using a purely theoretical trust calculation model (the PeerTrust model by Li and Ling (2004)), will be done. The best RBFNN trust model will be developed, trained, validated, and evaluated (tested).

Chapter 7: EXPERIMENT 2: DEMONSTRATION OF THE PROPOSED SOLUTION IN REAL LIFE BY CONSTRUCTING ARDSTRUSTRBFNN MODEL

The demonstration of the proposed solution will be carried out for a real-world problem by developing the ARDSTrustRBFNN trust model, using the Amazon Relational Database Service (ARDS) (Amazon, 2019).

Chapter 8: RESULTS AND CONTRIBUTIONS

In this chapter, a summary of the research will be given. A discussion on the results obtained and contributions of this study will be provided. The most significant contributions made during this study will be addressed.

1.6 Conclusion

In this chapter, a brief description of what can be expected in this research was provided. A brief introduction to the term trust between electronic entities was presented. The problem statement, proposed solution, objectives, and research methodology were outlined and discussed, as well as the approach that will be followed to complete the research. In Chapter 2, the trust calculation in electronic environments will be addressed.

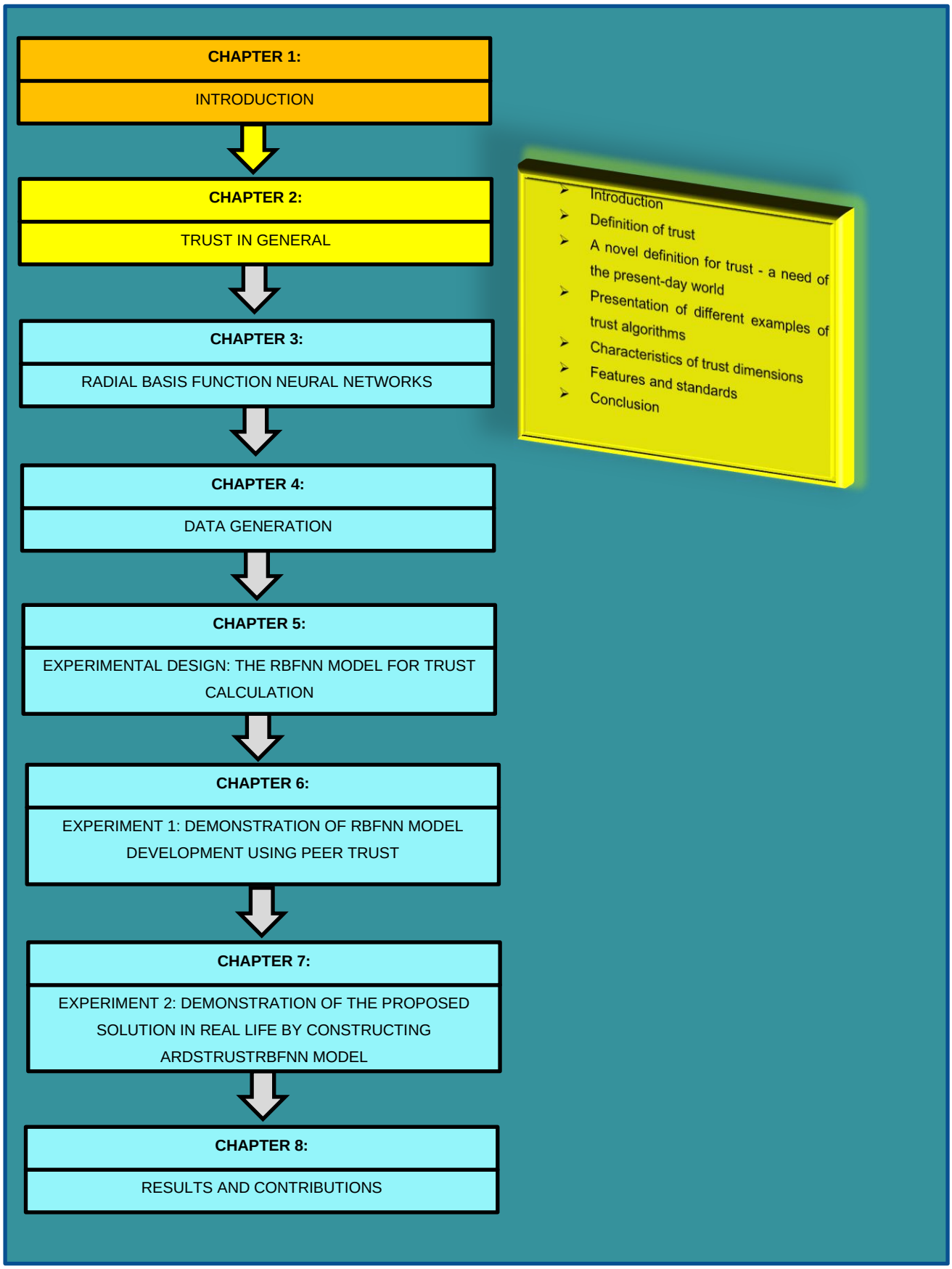


Figure 1-2: Signpost diagram of Chapter 2

CHAPTER 2 TRUST IN GENERAL

2.1 Introduction

Trust plays a major role in all human relationships (Das & Teng, 2004; Govier, 1998). Relationships would not exist without trust. Examples of this can be seen throughout human history. In the Bible, there was a breach in trust between God and humans as depicted in the story of Adam and Eve. Loss of trust has been behind many conflicts around the world. Great wars were fought and won by people who trusted in their leaders. Businesses know the importance of trust and many companies recognise it as one of the main pillars which will aid in the growth of the company.

The word trust has more relevance in this century because the internet plays such a major role in all the aspects of human existence, from physical and emotional well-being to financial aspects. Millions of transactions are entered daily on the internet by people visiting various websites. Mismanagement of data can lead to serious unfavourable issues. An important question to ask will be: “How does one trust in the virtual space?” How does an electronic entity trust another electronic entity?

In this chapter, trust is defined between electronic entities, and definitions and discussions on the three dimensions of trust, examples of trust calculations, characteristics of trust and problems in calculating trust in the electronic environment are provided. A two-step approach was followed in this regard. The first step was to conduct a thorough literature study to determine various available definitions of trust. The second step was to find the commonalities and differences between these definitions. From this literature, three dimensions of trust will emerge, which will lead to a new comprehensive definition of trust.

The remainder of the chapter is organised as follows. In Section 2.2, various definitions of trust are given and in Section 2.3, the three dimensions of trust are considered and a generic definition of trust within the electronic environment is given. Different examples of trust algorithms will be presented in Section 2.4. In Section 2.5, characteristics of trust dimensions will be identified by implementing the three dimensions of trust in different algorithms. In Section 2.6, standards of several products available in the market and the features offered by the Amazon Relational Database Service (ARDS) will be identified. The chapter will be concluded in Section 2.7.

2.2 Definition of trust

The focus point of this chapter is to analyse and define the term trust within the electronic environment, with a specific focus on the trust between software entities. In the fields of information technology and computer science, no widely accepted definition of trust could be found. Despite this lack of a widely accepted definition of trust, many researchers are trying to define trust in the field of computer science (Yong-sheng & Ying, 2010).

The following definitions of trust were found in the literature:

Definition 1: “Trust: a *subjective* expectation an agent has about another’s future behaviour based on the history of their encounters" (Mui *et al.*, 2002:5).

Mui *et al.* (2002:5) have performed a study on trust, reputation and reciprocity, where reciprocity is defined as a mutual exchange of deeds, and reputation is defined as a belief that is created by an agent through its past actions. Based on this study, a computational model of trust and reputation are proposed. The definition in Mui *et al.* (2002:5) suggests that one of the bases of trust is previous interaction with an entity and that trust is subjective (Mui *et al.*, 2002:5).

Definition 2: “Trust (or symmetrically, distrust) as a particular level of *subjective probability* with which an agent *assesses* that another agent or group of agents will perform a particular action, both before he can monitor such action and *in a context* in which it affects his own action" (Gambetta, 1988:217).

Gambetta (1988) summarised a wide variety of trust perspectives. He explained the importance of belief in others and defined trust. Trust is defined as a subjective assessment of the probability that an agent will perform as expected (Gambetta, 1988).

Definition 3: “Trust is a *subjective assessment* of QoS expressed *probabilistically*” (Hang *et al.*, 2012:9).

Hang *et al.* (2012) researched the topic of service selection via trust in composite services and proposed an approach for assigning trust to the constituents of composite services. There will be many services offering the same functions. Customers must be able to select a service by checking the required functionalities and evaluating the quality of service (QoS). According to Hang *et al.* (2012), QoS is subjective in two ways, depending on the services or preferences. Consumers may care about different QoS, and depending on the preferences for different needs, may have different interpretations of QoS performance. QoS information should be exchangeable and comparable; it should be monitored and tracked dynamically and it is necessary to have a service composition model for a better understanding of the QoS. Hang *et*

al. (2012) defined trust in service-oriented systems as an estimation of the quality of service in terms of probability. Thus, trust is defined in the context of QoS.

Definition 4: “Trust is the *assessment* by which one individual, A*, expects* that another individual, B, will perform (or not perform) a given action on which its (A’s) welfare depends, but over which it has limited control” (Witkowski & Pitt, 2000:1).

The approach of Witkowski and Pitt (2000) was to develop a notation of trust for software agents. Trust is defined here as a degree of dependency of A on B in the context of agent selection in a multi-agent trading society.

Definition 5: “Trust is an *assessment* that a person, organisation, or object *can be counted* on to perform according to a *given set of standards* in some domain of action (Au *et al.*, 2001:3; Denning, 1993).

For establishing trust across multiple organisations for external users on extranets, a paradigm is presented by Au *et al.* (2001). Extranets can be defined as the extended intranets in which external users can access internal resources and applications. To achieve this, the authors identified different types of extranet for cross-organisational interactions, the requirements of trust management in virtual enterprise environments and different approaches to infrastructure for a trust distribution. A framework for distributing trust on extranets across multiple organisations is proposed and the protocols of trust establishment on the web of trust are explained. An automated mechanism for deriving and composing trust is also explained in the paper.

Definition 6: “A *peer’s belief* in another peer’s capabilities, honesty and reliability based on its own direct experiences” (Wang & Vassileva, 2003:1).

Wang and Vassileva (2003:1) proposed a Bayesian network-based trust model. A method for building a reputation in peer to peer (P2P) networks based on recommendations was also discussed. To this end, the definitions of trust and reputation and their characteristics were identified. Trust is earned between two entities from direct experiences (Wang & Vassileva, 2003).

Definition 7: "The *firm belief* in the competence of an entity to act dependably, securely, and reliably *within a specified context*" (Grandison & Sloman, 2002a:2).

Grandison and Sloman (2002a) defined trust in the context of internet applications after surveying, to study various definitions of trust. Properties of trust relationships, classes of different types of trust and trust management were also discussed. A trustor and trustee

relationship cannot be declared complete, as the trust relationship depends upon the ability to perform a specific task or service in a certain circumstance or within a specific context. A trustor will not trust a trustee in every context and hence a trust relationship is not absolute. Trust is not symmetric, and some trust relationships should not be transitive. Resource access trust, service provision trust, certification of a trustee, delegation, and infrastructure trust are some of the classes of different types of trust. An entity earns trust by its secure and reliable dealing.

Definition 8: "Trust as *a quantified belief* by a trustor with respect to the competence, honesty, security and dependability of a trustee *within a specified context*" (Grandison & Sloman, 2002b:2).

Grandison and Sloman (2002b) developed a notation called the Simple Universal Logic-oriented Trust Analysis Notation (SULTAN) for internet applications. This notation includes tools for the specification, analysis, and management of trust relationships. The context of a trust relationship is defined as a set of actions with a trust level applying to all the actions, and a set of constraints, which must be evaluated for the trust relationship to apply (Grandison & Sloman, 2002b).

Definition 9: "Trust of a party A to a party B for a service X is the *measurable belief* of A in that B behaves dependably for a specified period *within a specified context* (in relation to service X)" (Olmedilla *et al.*, 2006:5).

Olmedilla *et al.* (2006:5) believe that the meaning of the term "trust" changes in different contexts. Policy-based and reputation-based trust are the two main approaches to trust management. The authors describe the trust and trust management approaches in the context of a virtual organisation's lifecycle and resource access control in the grid, where grid computing is a computer network that allows the sharing of computer resources or services.

Definition 10: "Trust is a level of subjective probability between two entities, a trustor (i.e., source entity) and a trustee (i.e., target entity), which is formed through the direct observation nature and/or recommendation from trusted entities, to fulfilling a particular service within a specific time and context" (Mohsenzadeh & Motameni, 2015).

Mohsenzadeh and Motameni (2015) described a trust model named TMFM (Trust model based on fuzzy mathematics). This model is for the cloud computing environment and is based on fuzzy mathematics. The fuzzy direct trust relation calculation is based on direct experiences between cloud entities. According to Mohsenzadeh and Motameni, the most complex relationship between trustor and trustee is the trust relationship. This is because the trust relationship is subjective, non-symmetric, partial transitive, dynamic, context-dependent,

uncertain, and trust is difficult to evaluate and establish. The capability to fulfil a task in a particular time and context by an entity with a recommendation from trusted entities defines the level of subjective probability of trust (Mohsenzadeh & Motameni, 2015).

2.3 A novel definition of trust - a need of the present-day world

In many definitions of trust, it is defined within a context in which it is applied. However, some definitions state that trust can be applied in a wider context too. Most of the definitions show that trust can be calculated as a probability and some of the above definitions show that the calculation must be done by accessing information from different sources. One of the above definitions shows that some set of standards or guidelines must be used in entity assessment. Each definition is well defined, and it becomes relevant in the situation under which it is defined. Further, the definitions given by Hang *et al.* (2012) and Grandison and Sloman (2002a) clearly show that trust can be used in a wider context and the trust relationship depends upon the ability to perform a specific task within a context. Moreover, the non-symmetric and non-transitive property of trust also gives an open door to a wider context concerning the trust.

There is, therefore, a need for a comprehensive definition of trust which can be used in general. In order to formulate a new definition, the dimensions of trust must be identified from the commonalities and differences between the above listed definitions.

The definitions given in Section 2.2 lead to the following observations:

2.3.1 Dimensions of Trust

In this section, the dimensions of trust will be identified from the commonalities and differences between the trust definitions given in Section 2.2. Trust contains three dimensions, namely the dimension of context, the dimension of calculation or quantification of trust and the dimension of information sources that need to do the calculation. The information needed will depend on the algorithm used in the calculation, as well as on the context. The three dimensions of trust will be discussed in the next three sections.

2.3.1.1 Trust context

Most of the authors defined trust within a context in which the trust is applied. In Definition 1, the context is reputation, in Definition 3, the context is service selection in service-oriented systems, in Definition 4, the context is agent selection in a Multi-agent Trading Society, in Definition 5, the context is distributing trust on extranets across multiple organisations, in Definition 6, the context is a P2P network, in Definition 7 and Definition 8, the context is internet applications and in Definition 9, the context is virtual organisations. Some authors describe trust in a specific

context and others describe trust in general. For this reason, context can be considered as one of the factors which contributes to the trust value in trust assessment. One of the common beliefs regarding trust is that it is context-dependent (Gambetta, 1988; Grandison & Sloman, 2002a; Grandison & Sloman, 2002b; Mohsenzadeh & Motameni, 2015; Olmedilla *et al.*, 2006). Therefore, the term trust is a generic concept, and it can be effectively used in various contexts. No standard list of contexts under the field of information technology and computer science could be found. Security, service delivery, reliability, and credibility are examples of context.

2.3.1.2 Calculations for quantification of trust

Another basic feature of trust is that it can be calculated as a probability (Gambetta, 1988; Hang *et al.*, 2012; Mohsenzadeh & Motameni, 2015). Definition 2, Definition 4, Definition 5 and Definition 10 contain an assessment of an entity by another entity to establish trust. Definition 3 describes trust as a probability assessment of the quality of service. By considering Definition 5, entity assessment can be guided by a set of standards or guidelines. Definition 1 describes trust calculation based on the history of encounters between the entities and Definition 6 describes trust calculation as based on the direct experience of the belief holding entity with the target entity. Definition 2 defines trust as a level of subjective probability and Definition 3 defines trust as a probability of quality of service. In Definition 8, trust is defined as a quantified belief and in Definition 9, trust is defined as a measurable belief. There are several trust calculations for the quantification of trust, especially in the security context. Some of these will be described in more detail in Section 2.4. Trust quantification is used in various trust models and various trust calculation algorithms are used for the quantification process (Au *et al.*, 2001; Wang & Chi, 2006; Wang *et al.*, 2008).

2.3.1.3 Information Sources

Trust establishment can be achieved by *accessing* information from different sources (Au *et al.*, 2001; Denning, 1993; Gambetta, 1988; Hang *et al.*, 2012; Mui *et al.*, 2002; Wang & Vassileva, 2003; Witkowski & Pitt, 2000). Definition 1 describes trust calculation based on the number of history of encounters between the entities (Mui *et al.*, 2002). From Definition 5**,** trust assessment will be done according to a given set of standards (Au *et al.*, 2001; Denning, 1993). Another belief regarding trust is that it is *subjective* (Gambetta, 1988; Hang *et al.*, 2012; Mohsenzadeh & Motameni, 2015; Mui *et al.*, 2002). To quantify trust, information is required. Information can be obtained from different sources, such as customers, other peers in the network, given standards of an entity, etc. To increase the confidence in the information, the number of parameters bounding the information can be increased in trust calculation.

In the case of a P2P network, information, such as feedback regarding the transaction, can be obtained from the other peers in the network (an information source). This can be done using several parameters, such as the trust value of the peer who submits the feedback, amount of satisfaction obtained from the other peer, total number of transactions that the peer has with other peers, and size of the transaction, etc. In the case of websites, trust can be calculated by evaluating the behaviour of each user of a website. In this way, the information (behavioural data) can be obtained from the information source (customer). To increase the confidence in the information, different parameters, such as the average time spent by a user on the website, the average number of web pages visited by a user in a single session, etc. can be obtained. In the case of an entity with given standards, information can be obtained from the given standards and from the customers in terms of satisfaction. For example, the given standards may state that the entity will be available 24 hours a day. Then trust can be calculated using the information obtained from the given standards and the feedback in terms of satisfaction obtained from the customers. Thus, trust can be calculated using different information sources (given standards and customers). Information, such as feedback about past experience, human behaviour, presence of W3C methods, etc. can be obtained from a variety of sources, such as peers in the network or customers and the given standards of an entity.

As described above, trust contains three dimensions, namely the dimension of context, the dimension of calculation or quantification of the trust and the dimension of information sources needed to do the calculation. These three dimensions describe trust.

2.3.2 New Definition of Trust

From the definitions of trust given in Section 2.2 and the observations in Section 2.3.1.1, it is clear that most of the authors defined trust within a context in which the trust is applied. It is shown in Section 2.3.1.1 that trust is a generic concept which can be applied in various contexts. Trust can be calculated as a probability as identified in Section 2.3.1.2. Section 2.3.1.3 reveals that some of the definitions given in Section 2.2 show that the calculation must be done by accessing information from different sources. It is also identified in Section 2.3.1.2, that one of the definitions given in Section 2.2 shows that some set of standards or guidelines must be used in entity assessment. As explained at the beginning of Section 2.3, there is a need for a comprehensive definition of trust which can be used in general. To arrive at a new definition for the term trust, the three dimensions of trust mentioned in Section 2.3.1 are identified from the commonalities and differences between the definitions of trust given in Section 2.2. Based on the discussion in the previous paragraphs, trust contains three important elements, namely the three dimensions of trust, i.e., the dimension of context, the dimension of calculation or

quantification of the trust and the dimension of information sources needed to do the calculation.

Based on the three dimensions identified in Section 2.3.1 and observations identified in Section 2.3.2, *trust* can be defined as follows:

New Definition: *Trust is a quantified belief or a probability of belief of an entity, which can be calculated by accessing or using information from different sources which are based on some set of standards or guidelines, to have some desired property within a specified context.*

Many of the trust definitions identified in Section 2.2 were defined within the context in which it was applied. The new definition of trust can be used in any context. The new definition of trust is defined, using the three dimensions of trust. The three dimensions, namely trust context, calculations for the quantification of trust, and information sources were obtained by identifying the commonalities and differences between several definitions. The new definition of trust is a generalisation of all the trust definitions identified in Section 2.2.

2.4 Presentation of different examples of trust algorithms

In this section, different examples of trust algorithms will be presented. The reasons for the insertion of different trust algorithm examples will be discussed in Section 2.5.

2.4.1 ARPANET version 1

The ARPANET began operation in 1969 with four nodes, namely the University of California Los Angeles (UCLA), the Stanford Research Institute (SRI), the University of California Santa Barbara (UCSB), and the University of Utah (Strawn, 2014). Interface message processors (IMPs) are small computers directly attached to each host computer on the network. The IMP software was called the network control program (NCP). The NCP provides services, such as file transfer, e-mail, and remote login. Packet switching was the technical approach used in ARPANET. In this method, long data messages are broken into small pieces called packets. The address of the destination IMP will be attached to each packet and each packet will also contain a message sequence number. The message can be put back in order even if the packets arrived out of order. The packets can be routed through other IMPs on their way to their final IMP destination, and each IMP can serve as a routing point. This first version of the ARPANET had interconnected computers of different types.

The Defence Advanced Research Project Agency (DARPA) and Defence Data Network Program Management Office (DDN PMO) set broad guidelines for accessing and using the ARPANET. The ARPANET was administered locally by host administrators. The network was

not available to the general public. The users had to use the network only for official business in which their access was authorised. Login names and passwords were used to access the host. If any violations of the policies happened, that terminal or host was terminated from the network (Dennett *et al.*, 1985).

In the ARPANET, trust between the four different computers was implicitly assumed, since they were a small group which trusted each other. In the next section, the second version of the APRANET is discussed.

2.4.2 ARPANET version 2

The first version of ARPANET consisted of interconnected computers of different types, but in the second version of the ARPANET, networks of different types were interconnected (Strawn, 2014). The fathers of the internet, Vint Cerf and Robert Kahn, created a virtual network out of software. Several adjacent networks were connected using small computers called routers. The software in the router would decide where to route each data packet. Each packet contained the destination address which was called the computer's internet protocol number. The first part of the internet protocol number was the network number. Each router had a routing table in its memory which indicated which link should be used to send the packet to reach the correct destination. In the second version of ARPANET, the router software was divided into two parts, namely the Transmission Control Protocol (TCP) and the Internet Protocol (IP). The TCP set up end-to-end virtual connections on the network, converted messages into streams of packets and reassembled them into messages. The IP forwarded packets along with the right network link at the routers.

Packet switching was the technical approach used in the first version of ARPANET. In contrast, the second version of ARPANET used TCP and IP. This shows that as the size of the network increases, the technical approach also gets more complicated, and the importance of trust also started to increase. In the next section, trust management methods in the peer-to-peer (P2P) system are addressed.

2.4.3 Managing trust in a P2P Information System

Aberer and Despotovic (2001) identified several problems in trust management in a decentralised information system. The level of trust of a peer can be accessed by measuring its reputation. The problem of reputation-based trust is described as follows. When a peer, e.g. Alice, interacts with another peer, e.g. Bob, the behavioural data set of Alice contains all the behaviour; that is, the report of transactions that are made about Alice and made by Alice. Formally, the behavioural data set of a peer a can be denoted as

$$B(a) = \{t(a,b) \text{ or } t(b,a) \mid a,b \in P\} \subseteq B, \quad (2\text{-}1)$$

where P denotes the set of all peers, $a, b \in P$, B are the behavioural data, $t(a,b)$ is the observation made when a interacts with b, and $B(a)$ is the behaviour of a based on B. In this case, the reputation of peer a can be obtained from $B(a)$. Trust can then be assessed, based on the reputation. In this model, to calculate trust, the information (behavioural data) is obtained from the information source that is the peers in the network. In order to increase the confidence in the information, Aberer and Despotovic (2001) used parameters, such as total number of complaints received and total number of complaints filed by the peer. To manage the behavioural data in trust management, some centralised database is required because trust access and management can be a problem in completely decentralised environments, especially in P2P networks. Suppose a peer, e.g. Alice, has no access to the global data $B(b)$ and B, but wants to determine the trustworthiness of another peer, e.g. Bob. In this case, peer Alice must rely on the behavioural data that is obtained from direct interactions with peer Bob and peer Alice can also obtain data from other peers who have interaction with peer Bob. The information obtained from other peers is not necessarily correct, as they can be malicious. Aberer and Despotovic (2001) describe a reputation-based trust model. In the next section, details of this model are given.

2.4.3.1 Trust management

When a peer, e.g. Alice, interacts with another peer, e.g. Bob, and if Bob was cheating, then Alice can file a complaint, $c(p,q)$, where c is the complaint and p and q denote the peers, Alice and Bob. In this scenario, complaints are the only behavioural data recorded in B and trust is represented by 1, 0, or -1, where 1 denotes trustworthy, 0 indicates no assessment and -1 denotes untrustworthy.

In this model, a P-Grid as described in Aberer (2001) is used as a decentralised storage structure to store complaints (Aberer & Despotovic, 2001). Every peer Alice can file a complaint about Bob at any time and the identifier corresponds to the search key. The complaints can be stored by sending the messages, e.g. insert (α_1, key(p), $c(p,q)$) and insert (α_2, key(q), $c(p,q)$) to arbitrary peers α_1 and α_2 in order to spread the untrustworthy behaviour of peer Bob widely. The insertion algorithm in the model will forward the complaints to one or more peers, storing complaints about Alice and Bob, respectively. When a peer wants to evaluate the trustworthiness of another peer Bob, it can retrieve the complaint data by submitting a message query (α, key(q)) to an arbitrary peer α. This may be repeated, e.g. s times for getting several referrals and to obtain a result set W, with

$$\mathrm{W} = \{(cr_i(q),\ cf_i(q),\ \alpha_i,\ f_i) \mid i = 1, \dots, w\}, \tag{2-2}$$

where w is the number of different witnesses found, α_i is the identifier of the i^{th} witness, f_i is the frequency with which witness α_i is found, $\mathrm{s} = \sum_{i=1}^{w} f_i$ and $cr_i(q)$ and $cf_i(q)$ are the number of complaints, that according to witness α_i agent q has received and filed respectively.

In this model, the reputation of a peer is defined as a scalar product and can be denoted as

$$T(p) = \left|\{c(p,q) \mid q \in P\}\right| \cdot \left|\{c(q,p) \mid q \in P\}\right|. \tag{2-3}$$

Peer p is not trustworthy if $T(p)$ has a high value. The peer who stores the complaints can itself be malicious, influencing its trustworthiness. To solve this problem, Aberer and Despotovic (2001) assume that peers are only malicious with a certain probability. If a peer receives the same data about a specific peer from enough other peers, the received information about the peer can be true. In the next section, trust in decentralised P2P electronic communities will be considered (Li & Ling, 2002).

2.4.4 Building Trust in Decentralised Peer-to-Peer Electronic Communities

Li and Ling (2002) developed a reputation-based trust mechanism called PeerTrust for quantifying and comparing the trust between peers in a decentralised P2P electronic market. In this model, the information source is the peers in the network. Trust is calculated using the feedback (information) obtained from other peers. In order to increase the confidence in the information, Li and Ling (2002) use three main parameters in trust calculation. The main parameters are satisfaction a peer obtains from other peers, the total number of transactions a peer has with other peers and the credibility of the feedback sources. Different feedback systems differ from each other in terms of the feedback mechanism and feedback measure. In this trust model, the interaction-based complaint system is used. If a peer receives a complaint from another peer after an interaction, it means the satisfaction it gets through that interaction is 0, otherwise it will be 1.

The trust value for peer a up to transaction t, $\mathrm{T}(a,t)$ can then be defined as

$$\mathrm{T}(a,t) = \sum_{b \in P,\ b \neq a} S(a,b,t), \tag{2-4}$$

where P denote the set of N peers in an electronic community, $a, b \in P,$ and $S(a,b,t)$ is the amount of satisfaction peer a has with peer b up to transaction t.

In this trust calculation, the calculation is not fair as the total number of interactions is not considered. For example two peers can get the same trust value in which one peer may be having 10 interactions and the other will be having 100 interactions. This is because the misbehaviours of peers are not considered. The trust value for peer a up to transaction t can be given as

$$\mathrm{T}(a,t) = \frac{\sum_{b \in P,\ b \neq a} S(a,b,t) \cdot Cr(b,t)}{\sum_{b \in P,\ b \neq a} I(a,b,t)}, \tag{2-5}$$

where $I(a,b,t)$ is the number of interactions that peer a has with peer b up to transaction t, $I(a)$ is the total number of transactions peer a has with other peers in the network, and $Cr(b,t)$ is the balance factor of trust that offsets the danger of non-credible feedback from peer b.

A higher value of $\mathrm{T}(a,t)$ shows peer a is more trustworthy. Complaints from a trustworthy peer are more trustable, while complaints from an untrustworthy peer cannot be trusted. The balanced amount of satisfaction $S(a,b,t) \cdot Cr(b,t) = C(a,b,t) \cdot \mathrm{T}(b,t)$, where $C(a,b,t)$ is the number of complaints peer a receives from peer b up to transaction t, and $C(a,b,t) \cdot \mathrm{T}(b,t)$ are the credible complaints filed by peer b. Complaint-based trust can be given as

$$\mathrm{T}(a,t) = 1 - \frac{\sum_{b \in P,\ b \neq a} C(a,b,t) \cdot \mathrm{T}(b,t)}{\sum_{b \in P,\ b \neq a} I(a,b,t)}, \tag{2-6}$$

where $\mathrm{T}(a,t)$ lies between 0 and 1. A P-Grid data structure is used in this model to manage the data. Each peer will maintain a fragment of the data and a routing table to other peers for the data which is not stored locally.

According to Li and Ling (2002), trust can be calculated in two ways, using Equation (2-6). The first method they describe is a dynamic computation that uses the recent trust data of all peers collected at runtime to compute the trust value. The second method is an approximate computation. In this method, each peer will be maintaining a trust cache to keep the trust values the peer has computed for other peers. When a peer a wants to evaluate the trust value of another peer b, peer a will just retrieve the trust value from cache if it is available. If the trust value of peer b is not in cache, peer a will either use a default trust value or will compute the

trust value of peer b and will update the cache. Thus, for an approximate computation, Equation (2-6) becomes

$$\mathrm{T}(a,t) = 1 - \frac{\sum_{b \in P,\ b \neq a} C(a,b,t) \cdot T^1(b,t)}{\sum_{b \in P,\ b \neq a} I(a,b,t)}, \tag{2-7}$$

where, $T^1(b,t)$ is the trust value of peer b and

$$T^1(b,t) = \begin{cases} \mathrm{T}_{Cache}(b,t^1) & cache\ hit \\ \mathrm{T}(b,t)\ or\ \mathrm{T}_{default} & cache\ miss \end{cases},$$

where $\mathrm{T}_{Cache}(b,t^1)$ is the old trust value of peer b up to transaction t^1 in peer $c's$ cache if peer c want to calculate the trust value of peer a and $\mathrm{T}_{default}$ denotes a default trust value. After computing the new trust value of peer a, peer c will update the trust value of peer a with the newly calculated trust value in the cache. Use of cache data will speed up the trust computation process and will reduce the communication cost for the trust calculation.

2.4.5 Hybrid Trust Algorithm

Daskapan *et al.* (2008) developed a hybrid algorithm based on the strengths and weaknesses of algorithms described in Aberer and Despotovic (2001), Abdul-Rahman and Hailes (2000), Agostini and Moro (2004), Almenárez *et al.* (2004), Commerce *et al.* (2002), Jin *et al.* (2005), Kamvar *et al.* (2003), Mengshu *et al.* (2005), Sierra and Debenham (2005), Shi *et al.* (2005), Wang and Vassileva (2003), Li and Ling (2002) and Yu and Singh (2002). This hybrid algorithm is given in Figure 2-1.

In this algorithm, when a peer, Alice, joins the P2P network, the network manager will assign a security level for Alice. This will be done using the security level system which is a user intervention system in the model that allows the network manager to assign a security level for a newly joining peer to the network. The security level can be a number between 1 and 9. The assumption in this algorithm is that the first few peers in a network can be considered as trustworthy because designers and the first users are likely to have less intention to destroy the network. These peers are also called pre-trusted peers.

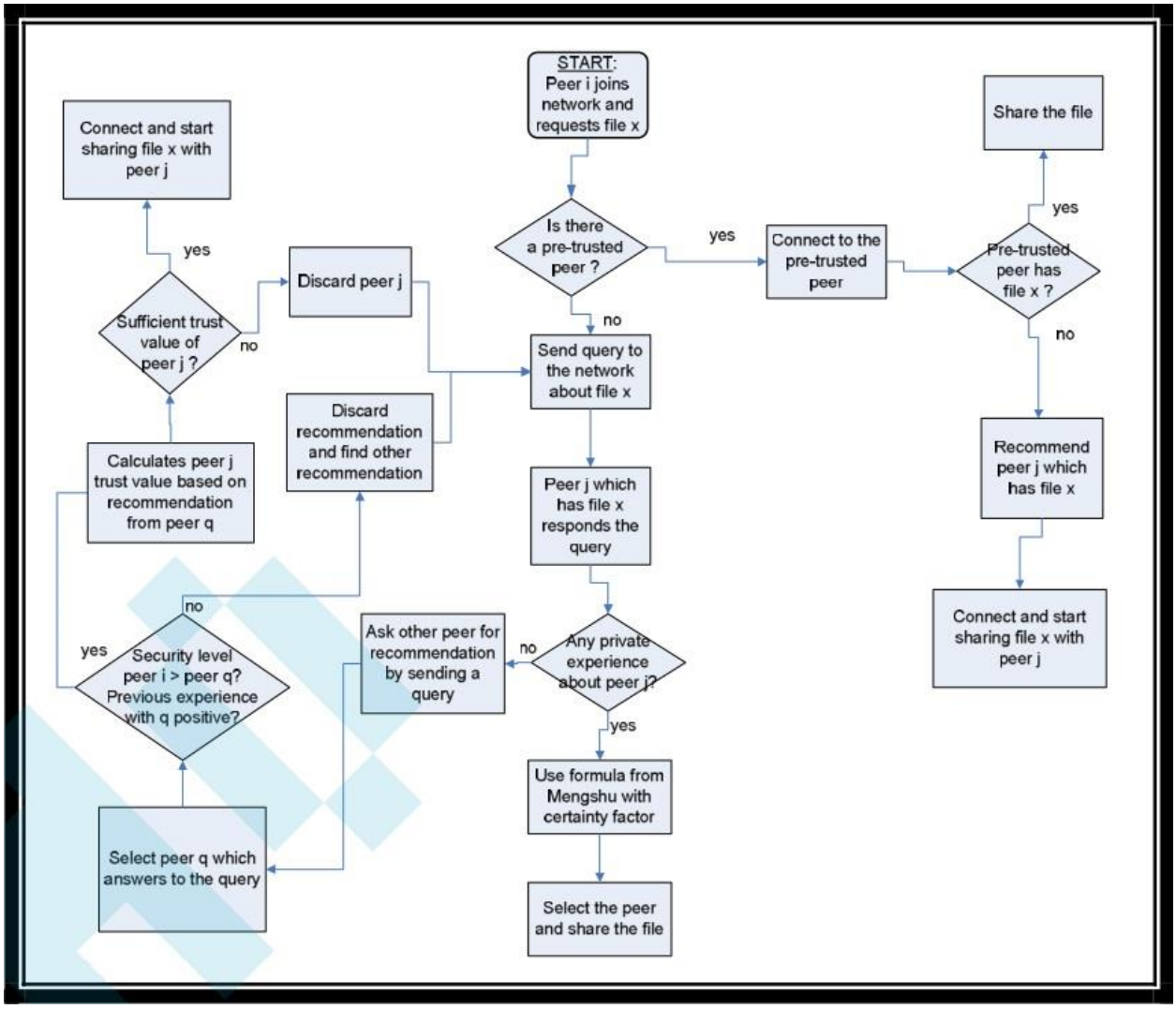


Figure 2-1: Hybrid Algorithm (Daskapan *et al.*, 2008)

Let peer Alice who wants to join the network looking for a specific file and Faythe be pre-trusted peers. If peer Alice does not have any knowledge about any other entities in the network, then Alice will check for the existence of a pre-trusted peer and find Faythe. Then, $T_{AF}^{c} = T_{F}^{c}$, where, T_{AF} denotes the trust value peer Alice has received in peer Faythe, T_F is the trust value of pre-trusted peer Faythe, and the constant *c* denotes a certain context, in this case for a specific file. If Faythe does not have the requested file x, peer Faythe will recommend to peer Alice about another trusted peer Bob who has the requesting file. Then, $T_{AB}^{c} = T_{FB}^{c}$.

If there is no pre-trusted peer in the network, peer Alice will send a query to the network through the nodes close to it. This sending query will contain the identifications for the requested file, identification for the requesting peer, context and the time to live of the query as a timer to return the query.

Suppose peer C called Charlie has the file, which is requested, and Charlie responds to Alice's query. Then peer Alice will check whether the security level of Charlie is greater than or at least equal to the security value of Alice. If this is false, Charlie will be discarded by Alice and Alice will choose another responding peer. Otherwise, Alice will check whether it has private experience with Charlie. This is done by checking the local trust value

$$C_{AC} = \begin{cases} \frac{(S_{AC}^{c} - F_{AC}^{c})}{(S_{AC}^{c} + F_{AC}^{c})} & if\ \beta_C \geq \beta_A \\ discard\ peer\ C & \text{Otherwise} \end{cases} \quad (2\text{-}8)$$

where F_{AC} is the number of unsuccessful transactions between peer Alice and peer Charlie, S_{AC} is the number of successful transactions between peer Alice and peer Charlie, C_{AC} is the local trust value peer Alice has found in peer Charlie and β is the security level. If the local trust value C_{AC} is obtained, Alice needs some recommendations from other peers to calculate the more accurate trust value of Charlie. Thus, Alice will send a query to the network requesting recommendations. Suppose peer D called Dave responds to this query. Before receiving the recommendation, Alice will check whether the security level of Dave is greater than or at least equal to that of Alice.

Therefore, the trust value of Charlie by Alice is given as

$$T_{AC}^{c} = \begin{cases} (1-a)C_{AC}^{c} + aT_{DC}^{c} & if\ \beta_D \geq \beta_A \\ discard\ peer\ D & Otherwise \end{cases} \quad (2\text{-}9)$$

where a is a constant value given by the system that lies between 0 and 1 to balance between the old trust value and the new trust value. After completing the transaction between peer Alice and Charlie, peer Alice will update or add the stored trust value of Charlie. This will be based on the result of the last transaction and will be done using the equation

$$V^{c} = \left(1 - \frac{F_{AC}^{c}}{S_{AC}^{c} + F_{AC}^{c}}\right).W^{\beta} \quad (2\text{-}10)$$

where V is the action value, F_{AC}^{c} is the total number of unsuccessful transactions between peer Alice and Charlie, S_{AC}^{c} is the total number of successful transactions between peer Alice and Charlie, W is the action weight being 1 for a successful transaction and 0 for an unsuccessful

transaction. After committing the transaction, Alice will adjust the trust value of Charlie. The new trust value of Charlie is defined as

$$T_{AC}^{c,New} = V^c a + T_{AC}^c.(1-a). \quad (2\text{-}11)$$

Finally, if the transaction is unsuccessful, Dave is considered malicious, and Alice will send a warning message to the network. This warning message will contain a sender id, malicious id, context id and time to live. Thus, in this model, the information source is the peers in the network. Trust is calculated using the recommendations (information) obtained from other peers in the network (information source). In order to get more confidence in the recommendations (information), Daskapan *et al.* (2008) used several parameters, such as the number of successful transactions between peers, the number of unsuccessful transactions between peers, the security level of the peer, and the action weight of the current transaction to calculate the trust value.

From the above hybrid trust model, it is clear that trust within the P2P security context is well documented. To determine whether this peer will also perform well in other contexts, trust must be calculated in these contexts. Sometimes the size of the file can be very high, which will increase the cost of downloading the file. Hence, to determine the influence of the size of the requested file, trust must be calculated in the context of cost. Trust must be calculated in the context of timeliness to determine whether the trusted peer will deliver the file on time. Furthermore, trust should be calculated in the context of availability to find out whether the trusted peer will be available to provide the file whenever needed. Therefore, there are other, different trust contexts in P2P networks where trust could be calculated. The need for a pre-trusted peer is also a problem in this algorithm. If the pre-trusted peer becomes malicious, the network can fail.

In this method for the calculation of trust, the algorithm used three parameters, namely feedback in terms of the satisfaction a peer obtains from other peers, the total number of transactions a peer has with other peers and the credibility of the feedback sources. Moreover, there should be a pre-trusted peer in the network. In this method, it is observed that the algorithm is more complicated than the other algorithms described in Sections 2.4.1 to 2.4.4 above.

2.4.6 PeerTrust

PeerTrust is a trust model developed by Li and Ling (2004). This PeerTrust model will be used in Chapter 6 for the demonstration of the proposed solution, using the purely theoretical trust

calculation model. This is because PeerTrust is a trust model for P2P transactions, which deals with many problems that can arise during the trust calculations. This model will quantify the trustworthiness of peers in P2P electronic communities. Li and Ling (2004) identified several problems in existing reputation trust calculation systems. This includes the lack of ability of the system to differentiate honest feedback from dishonest feedback, the inability of the system to support and cooperate various contexts, the inability of the system to provide incentives for peers who provide feedback and the inability of the system to identify the strategic dynamic personality of malicious peers to build a reputation. To solve all these identified problems, Li and Ling (2004) identified five important factors for evaluating the trust of the peer in P2P electronic communities to develop the PeerTrust model:

- The first factor is the feedback a peer obtains from another peer. According to Li and Ling (2004), some reputation-based systems use only this feedback factor to compute a peer's trust, and it will be the summation of all the feedback a peer received. They identified a problem in the summation of all the feedback. If the feedback of a peer is rated as values -1, 0, or 1 and if the overall reputation is the sum of all the received ratings, it will be an unfair judgement for some situations, such as a buyer and seller community. If, e.g., a peer Alice does dozens of transactions and cheats in one out of every four transactions and peer Bob completed only a few transactions, but was completely honest, peer Bob will be treated as less reputable.
- The second identified factor is the feedback scope, representing the number of transactions. As described in the first factor, the peer may increase its trust value by increasing the number of transactions to hide the fact that it misbehaves at a certain rate if the simple summation of feedback is used by a trust model. Therefore, a metric should be defined as a ratio of the total amount of satisfaction a peer Alice receives over the total number of transactions peer Alice has.
- The third identified factor is the credibility of the feedback, since a peer can send false feedback about the other peer's service.
- The fourth identified factor is the transaction context factor. For each transaction, the context can be different. In the case of a seller and buyer P2P community, the seller can be honest for all the small transactions and can be dishonest for large transactions to make more profit. Therefore, the value of the transaction (e.g. smaller or higher amount) and functionality of the transaction (context of selling items) are important trust contexts.
- The fifth identified factor is the community context factor. This can include giving rewards for peers who submit feedback, and availability of digital certificates. Adding a reward as a community context for giving feedback to others can reduce incentive problems in the reputation system. However, this can increase the chance of increasing the trust value

by giving feedback to others. The amount of trust that can be gained by giving feedback to others should be controlled by some weight factors. Adding a reward as a community context for those peers having digital certificates, can make the trust metric more robust against malicious peers' manipulations.

By considering all the five factors and associated problems, Li and Ling (2004) formulated a new trust calculation. The trust value of peer Alice is defined as

$$T(a) = \alpha \cdot \sum_{i=1}^{I(a)} S(a,i) \cdot Cr(p(a,i)) \cdot TF(a,i) + \beta \cdot CF(a), \qquad (2\text{-}12)$$

where $T(a)$ is the trust value for peer a, $I(a)$ is the total number of transactions peer a has during the recent time window, $S(a,i)$ is the amount of satisfaction peer a receives for the i^{th} transaction, $Cr\big(p(a,i)\big)$ is the credibility of the interacted peer in the ith transaction, $TF(a,i)$ is the adaptive transaction context factor for peer a's i^{th} transaction, $CF(a)$ is the adaptive community context factor for peer a, $F(a)$ is the total number of feedback peer a give others and α and β are the weight factors for the transaction context factor and the community context factor, depending on the situation.

By turning off the transaction context factor ($TF(a,i) = 1$) and community context ($\alpha = 1$ and $\beta = 0$) in Equation (2-12), the trust value for peer Alice is obtained by the following equation which is known as the basic trust matrix:

$$\mathrm{T(a)} = \sum_{i=1}^{I(a)} S(a,i) \cdot Cr\big(p(a,i)\big) \qquad (2\text{-}13)$$

This is the trust value of a peer Alice calculated by a weighted average of the amount of satisfaction peer Alice receives for each transaction. In Equation (2-13), the number of the transaction $I(a)$ and feedback $S(a,i)$ are quantitative measures and can be collected automatically by a feedback system. The value $S(a,i)$ can be positive, negative, a numeric rating or in a mixed format. Since the credibility factor is a qualitative measure, it should be computed based on the past behaviour of the peers who submit feedback.

The PeerTrust model proposes two credibility measurement approaches. The first method is using a function of the trust value of a peer as its credibility factor:

$$T(a) = \sum_{i=1}^{I(a)} S(a,i) \cdot \frac{T\big(p(a,i)\big)}{\sum_{j=1}^{I(a)} T(p(a,j))}. \quad (2\text{-}14)$$

The second method of credibility uses a personalised similarity measure. Suppose trust is calculated using the personalised similarity measure between the peer w called Walter and any other peer. In this case, peer Walter uses a personalised similarity measure based on Walter's personal experience to rate the credibility of another peer Bob. Then

$$T_{PSM}(a,w) = \sum_{i=1}^{I(a)} S(a,i) \cdot \frac{Sim(p(a,i),w)}{\sum_{j=1}^{I(a)} Sim(p(a,j))}, \quad (2\text{-}15)$$

where
$$Sim(b,w) = 1 - \sqrt{\frac{\sum_{x \,\epsilon\, IJS(b,w)} \left(\frac{\sum_{i=1}^{I(x,b)} S(x,i)}{I(x,b)} - \frac{\sum_{i=1}^{I(x,w)} S(x,i)}{I(x,w)} \right)^2}{\big| IJS(b,w) \big|}}, \quad (2\text{-}16)$$

where $T_{PSM}(a,w)$ is the trust of peer a using a personalised similarity measure of peer w as the credibility measure, $Sim(p(a,i),w)$ is the personalised similarity measure of peer w to rate the credibility of other peers, $IJS(b,w)$ is the common set of peers that has interacted with both peer b and w, $I(x,b)$ denotes the total number of transactions performed by peer x with peer b, and $S(x,i)$ is the amount of satisfaction peer x receives for the i[th] transaction. When the transaction context factor is incorporated into the basic trust matrix, Equation (2-13) becomes

$$T(a) = \sum_{i=1}^{I(a)} S(a,i) \cdot Cr\big(p(a,i)\big) \cdot TF(a,i), \quad (2\text{-}17)$$

where $TF(a,i)$ is the transaction context factor.

The size, category or time stamp of the transaction can be applied as a transaction context. Therefore, compared to other transactions, more weight can be assigned for the feedback received for larger, more important, or more recent transactions.

Trust can also be calculated by incorporating both the transaction context and community context factors into the basic trust matrix. The community context factor will be calculated as the ratio of the number of feedback peer Alice gives to other peers over the total number of transactions that peer Alice has during the period. The weight factors α and β can also be incorporated into Equation (2-17). Thus Equation (2-17) becomes

$$T(a) = \alpha \sum_{i=1}^{I(a)} S(a,i) \,.\, Cr\big(p(a,i)\big).TF(a,i) + \beta \frac{F(a)}{I(a)}, \tag{2-18}$$

where $F(a)$ is the total number of feedback peer Alice gives to other peers and $I(a)$ is the total number of transactions that peer Alice has during the period. After incorporating the credibility as a function of the trust value of a peer, Equation (2-18) becomes

$$T(a) = \alpha \cdot \sum_{i=1}^{I(a)} S(a,i) \cdot \frac{\mathrm{T}\big(\mathrm{p}(a,\mathrm{i})\big)}{\sum_{j=1}^{I(a)} T(p(a,j))} \cdot TF(a,i) + \beta \cdot \frac{\mathrm{F}(a)}{\mathrm{I}(a)}. \tag{2-19}$$

It is the user's choice to decide how much the community context factor must contribute to trust value.

From the above PeerTrust trust model, it is again clear that trust for P2P electronic communities in a security context is well documented. There are other different trust contexts in P2P electronic communities where trust should be calculated as identified in Section 2.4.5. In this model, the information source is the peers in the network. Trust is calculated using the feedback (information) obtained from other peers. In order to increase the confidence in the information, Li and Ling (2004) use five main parameters in trust calculation. This includes the amount of satisfaction a peer obtains from other peers, the total number of transactions that a peer has with another peer, the credibility of the feedback, the transaction context factor, and the community context factor. It can be observed that the algorithm also becomes complex when the number of parameters or the size of the network increases. Genetic programming is a relatively new method and another trust calculation method that is entirely different from those explained above will be considered in the next section.

2.4.7 GenTrust

GenTrust is a trust model developed by Tahta *et al.* (2015) and this model is based on genetic programming. Using this trust model, every peer in a P2P network can calculate the trust value

of other peers in the network. In the GenTrust model, the trust calculation is based on the P2P interaction (any P2P application-specific activity, such as file sharing, CPU sharing and storage sharing), based on features and recommendation-based features. If a peer does not have information regarding another peer, it can use recommendations from neighbouring peers.

In this model, the information source is the peers in the network. Trust is calculated using the past interactions (information) and recommendations (information) obtained from other peers. In order to increase the confidence in the information, Tahta *et al.* (2015) use several features (parameters) regarding the past interactions and recommendations. More details regarding features (parameters) will be given below.

GenTrust in the case of a file-sharing application will be explained next. In this case, GenTrust interaction-based features (the peer's direct experience) include the number of interactions, the number of successful interactions, the average size of downloaded files, the average time difference between the last two interactions, the average weight and the average satisfaction.

The recommendation-based features include the number of recommendations, an average of a neighbour's number of successful interactions, an average of the neighbours' average satisfaction values, an average of the neighbours' average weight values, and an average of trust values. The weight parameter addresses the importance of interaction.

As the GenTrust model is based on genetic programming, a genetic programming tree is developed by using the operators and extracted features from past interactions and recommendations. Summation, subtraction, division, multiplication, inverse, log, square root, and square are the operators used in the GenTrust model for the calculation of trust of a peer. Each tree will generate a mathematical function to evaluate trust values. GenTrust uses a fitness function to select the best solution. The fitness function is defined as

$$Fitness = \frac{R_{trust}}{R_{noTrust}}, \tag{2-20}$$

where R_{trust} is the number of attacks with the trust model and $R_{noTrust}$ is the number of attacks without a trust model. The individual (a member of the set of possible solutions) that minimises the fitness function will be selected as the solution. According to Tahta *et al.* (2015), the GenTrust model is very effective against different types of attack, such as individual attackers, and collaborative and pseudospoofing attackers. In a security context, GenTrust for P2P networks is well understood. In this model, trust is also not well defined in some other contexts, such as timeliness, cost and availability. The number of parameters used by the GenTrust model is greater than the number of parameters used in the methods discussed above. All

these models are calculating trust within the security context. In the next section, the calculation of trust in another context will be considered.

2.4.8 The TNM trust model

Trust Necessitated through Metrics (TNM) is a trust model developed by Singal and Kohli (2016) to determine the trust of websites which mainly occur in the credibility context. In this model, trust is calculated by evaluating the behaviour of each user of a website. This model is based on human behaviour as websites are created and used by humans.

In the TNM trust model, the information source is the customers of the websites. Trust calculation is based on human behaviour (information). In order to increase the confidence in the information, Singal and Kohli (2016) uses metrics which include several parameters in trust calculation which will be explained below.

Trust is calculated by analysing the metrics of the average time spent by a user on the website (*Average time on website)*, the average number of web pages visited by a user in a single session (*Pages/Visit)*, the amount of traffic or number of users who have used the website (*Average Daily Visits)* and the number of users who have left the website within 10 seconds of their arrival (*Bounce rate)*.

The TNM tool uses Google's Custom Search Application Program Interface (API) to fetch web results of a search query. Google's Custom Search API enables the embedding of search functionality in a web application. The search engine can be configured according to a user's needs. Sometimes the result may contain different web pages of the same website. Therefore, unique websites will be identified with their frequency.

To collect values for all the metrics for all the identified unique websites, Similarweb.com APIs are used. Hence, the values for the above metrics will be obtained through the interfaces provided by Similarweb.com. To use the APIs in Similarweb.com, one must sign up and must get a unique user key. The API's input will contain the Domain (the target domain name top-level and second level), Start Month, End Month, Main Domain (*True/False*), User Key (unique user key) and Granularity (Daily, Weekly, Monthly) as parameters (SimilarWeb, 2016). When Main Domain is *True*, it returns data from the main domain and when it is *False,* it returns data from subdomains as well.

The data collected by the interfaces are pre-processed and normalised using R which is a free software environment for statistical computing and graphics (Team, 2016). Moreover, it has an API interface. As a result of the different analysis techniques, the metrics will get weights or

values to form an equation to calculate trust. The trust calculated in this model is in the credibility context.

In this model, the trust is calculated only in the credibility context and consequently, trust is not well defined in some other contexts. To illustrate this, suppose a client wants to buy something through a website. When doing a search for a suitable website, many websites will be listed. How does the client trust that the website will deliver the item on time? Available security trust models will not be able to calculate trust in the context of timeliness. Consider another set of examples, namely an online system for the government medical department to capture the information of the patients, a government electronic voting system, and a home affairs website. All of these systems should be accessible whenever needed. Currently, available security models are not enough to calculate trust in the context of availability. In the case of an electronic voting system, no valid vote is rejected or an invalid vote accepted, which means the system should be reliable. Trust should therefore be calculated in the context of reliability.

Consider the following situations: to access a web site there may be some service charges to access the service, some additional hardware or software may need to be set up, and sometimes the uploading and downloading data sizes will be huge. All these situations involve cost. For these reasons, trust must be calculated in the context of cost. Moreover, trust models in the context of security are not sufficient to calculate trust in the context of usability or ease of use of a web site. Another context is the response time. It can be the response time of a web site or the response time of a network. The reputation of a web site or of a cloud service provider will also affect trust value. Reputation is consequently also a context in the calculation of trust values.

The web sites on medical information and educational purpose data are additional example situations. These web sites must be useful and credible. The question remains, how can trust be calculated for these web sites in the context of credibility and usefulness? The value of credibility and usefulness will also affect trust value. However, the TNM model is a stand-alone solution for trust in the context of credibility of a web site, as the model is not able to calculate trust in other contexts related to a web site.

Web services is a special type of P2P network. Security is one of the major problems in using web services, but as pointed out above, it is well understood. In web services networks, there are consolidated ways of solving the security problems, using the work of the World Wide Web consortium (W3C). According to the W3C, W3C XML encryption, W3C XML signature, WS security tokens, WS-Secure conversation and WS Policy are some of the consolidated methods to solve web service security problems (W3C, 2002). All these W3C methods can be considered

as sources of information for calculating trust. As a specific example, there is a trust model for E-Government web services which was developed by Al-Shargabi (2016). This model will be explained next.

2.4.9 Al-Shargab's Security Trust Model for E-Government Web Services

The Security Trust Model for E-government Web Services is a trust model developed by Al-Shargabi (2016) to secure the communications and interactions between governmental web services. In this model, a trusted third party who is under the control of a government agency is responsible for providing a way for all governmental web services to communicate securely. The third-party controller will provide an identity for both parties (web service consumer and provider). The identity provided by the trusted third party can be used when both parties need to communicate or interact. The third party will encrypt the identity using XML encryption before sending it through SOAP messages. Using Public Key Infrastructure (PKI) techniques the third party will also control the communication between the web services and the third party (Al-Shargabi, 2016). However, this trust model describes trust only in the security context.

In the security trust model developed by Al-Shargabi (2016), there is no trust quantification. It is only a model which describes how to implement secure communication. The information source is the W3C methods. Even though there is no trust quantification, trust can be determined by checking the presence of W3C methods (information). In order to increase the confidence in the information, several parameters, such as the presence of XML encryption, the presence of a W3C XML signature, the presence of WS security tokens, and WS-Secure conversation and WS Policy can be used. Compared to all other examples of trust calculation algorithms presented in Section 2.4, in this trust model, the information source is not peers in the network or other interacting entities. In this trust model, the information source is the W3C methods as mentioned above. Moreover, the information used to calculate trust is not in any form of recommendation or feedback. The information is just the presence of W3C methods.

Consider a user in South Africa who plans to travel to the USA to attend a conference. It consists of flight booking from South Africa to the USA, booking of the hotel near to the conference venue, and transportation during the conference period. In this case, the user selected service may fail to function successfully according to the user's expectation. For example, the user selected plane may be delayed or that service may not protect user details. Therefore, the user should select a trustworthy service; trust should be taken into account for the selection of the trustworthy service (Hang *et al.*, 2012). According to Wang *et al.* (2015), the requirement of trustworthiness may be conflicting with user preferences, such that a user may want to select a particular flight option that can be unsecure. Therefore to solve this problem,

Wang *et al.* (2015) integrated trust into a multi-objective optimisation model together with qualitative and quantitative preferences. However, how will the user know whether the flight will land on time? In addition, how will the user know whether the hotel rooms will be as good as those offered on the web site? Trust should be calculated by considering all these factors. In the next section, the calculation of trust in a different context will be considered, namely in the context of quality of service web.

2.4.10 A Fuzzy Trust Management Framework for Service Web

Nepal *et al.* (2010) propose a fuzzy reputation model to predict the trust of a web service. In this model, the information source is the customers that use the service web. Trust is calculated using the feedback (information) obtained from customers. In order to increase the confidence in the information, Nepal *et al.* (2010) use quality of service parameters in trust calculation. In this model, the quality of a web service is taken as the reputation. The quality is measured, using a set of parameters, including performance, availability, reliability, and response time. Consumers can rate the service after consuming the service. The information used is a fuzzy set of quality of service parameters and evaluation of trust is executed by using a set of fuzzy functions. Another trust model in the context of reputation will be explained in the next section.

2.4.11 Trust prediction model for service Web using the Hidden Markov Model (HMM).

Sherchan *et al.* (2011) propose a trust prediction model for the service web. In this model, the context is quality of web service. They use reputation as a means to establish trust on the service web. The reputation of a web service can be considered as a reflection of its quality. Reputation is based on past experiences. In this model, the quality of service is calculated using a set of Quality of Web Service parameters, such as performance, availability, reliability, security and response time. Here trust is calculated using the consumer's feedback on these parameters after using the service. In this model, the information source is the customers that use the service web and the information used in trust calculation is the feedback obtained from customers. Sherchan *et al.* (2011) use quality of service parameters as parameters to increase the confidence in the information. The three dimensions of trust in the security context will be discussed in the next section.

2.5 Characteristics of trust dimensions

In this section, characteristics of the three dimensions of trust will be identified by analysing the different trust calculation examples presented in Section 2.4. Two analyses will be done, one with a fixed context in Section 2.5.1 and another with different contexts in Section 2.5.2. In Section 2.5.1, the dimension of context will be fixed to the security context, as most of the

examples given in Section 2.4 are in the security context. The changes to the dimension of information sources and in the dimension of calculation will be analysed when there is a change in the complexity of the environment. In Section 2.5.2, variation in the dimension of context will be applied, and the effect of variation in the complexity of the environment to the dimension of information sources and on the dimension of calculation will be analysed.

2.5.1 Information sources and Trust calculation in Fixed Context: Security

In this section, trust in the security context will be considered. The discussion will start with the information sources in a simple security network (ARPANET) and then move to a complex network (internet). The discussion regarding trust calculation in the security context will also be discussed.

In the ARPANET, the only information needed to calculate trust was from which node the message came, since all the nodes were trusted. On the internet, algorithms, such as Managing trust in a P2P Information System, Building Trust in Decentralised Peer-to-Peer Electronic Communities, the Hybrid Algorithm, the PeerTrust model, the GenTrust model, and Al-Shargab's Security Trust Model for E-Government web services are some of the available trust algorithms for trust calculation in the context of security. The information sources for most of these trust models include other peers in the network. For the Al-Shargabi (2016) trust model the information source is the W3C methods.

From Section 2.4 it is clear that the information used for trust calculation in most of these trust models include behavioural data, feedback, recommendations and past interactions, i.e., some information based on past interactions between peers. In the case of the Al-Shargabi (2016) trust model, the mere presence of W3C methods is sufficient information.

To increase the confidence in the information for trust calculation it is noticed that authors are increasing the number of parameters bounding the information. For instance, compared to the first version of ARPANET, the second version is more complex. More unknown nodes were added to the second version of the ARPANET, which lead to the establishment of the internet, a very complex network of networks. When the number of nodes increases, the complexity of the network will also increase, and the nodes are no longer necessarily trusted. As the complexity of the network increases, the complexity of the trust problem also increases. To increase the confidence in the information in trust calculation, the number of parameters bounding the information was also increased which led to the calculation of trust becoming more complex.

In the case of a large P2P network or a web site, information must be collected from a large number of unknown peers. The number of parameters needed to give confidence in the

information increases. For instance, the PeerTrust trust model used five main parameters, such as the amount of satisfaction a peer obtains from other peers, the total number of transactions that a peer has with the other peers, the credibility of the feedback, the transaction context factor, and the community context factor as described in Section 2.4.6. Therefore, the calculation complexity will increase automatically. As stated in Section 2.4, the above-mentioned trust algorithms used on the internet for trust calculation are more complex compared to the second version of the ARPANET.

In summary, in the security context, as the complexity of the network increases, the number of parameters to provide confidence in the information increases. This leads to an increase in calculation complexity. When the complexity of the network increased, in other words when the environmental complexity increased, the calculation or computational complexity also increased.

2.5.2 Information sources and Trust calculation in different contexts

In this section, trust in a context different from the security context will be considered, i.e., the dimension of context will be varied. The discussion will start with the information and then move on to trust calculation in a different context.

As mentioned in Section 2.5.1, the only information needed to calculate trust with the ARPANET was from which node the message came, as the nodes were trusted. This will be the same for calculating trust in any context for the ARPANET.

In the internet, algorithms, such as the TNM trust model in the context of credibility, the Fuzzy trust Management Framework for Service Web model, and the trust prediction model for service web, using the HMM in the context of quality are some of the available trust algorithms. In all these algorithms, the information source is the customers of the web sites or the service web. From Section 2.4, it is clear that the information used for trust calculation in most of these trust models includes human behaviour, and the feedback from the customers.

To increase the confidence in the information for trust calculation, it is noticed that authors are increasing the number of parameters bounding the information. For instance, compared to the first version of ARPANET, the second version is more complex as mentioned in Section 2.5.1. As the complexity of the network increases, the complexity of the context also increases. To increase the confidence in the information in trust calculation, the number of parameters bounding the information was also increased in different contexts. For this reason, the calculation of trust became more complex.

In the case of a large web site, behaviours of a large number of unknown customers must be quantified to calculate trust. In the case of a large service web, feedback must be collected from a large number of unknown users. In both cases, a number of parameters are needed to give credence to the information. Therefore, the calculation complexity will definitely increase in any context as the number of parameters increase. As stated in Section 2.4, the above-mentioned trust algorithms used on the internet for trust calculation are more complex compared to the second version of the ARPANET.

In summary, as in the security context, as the complexity of the network increases, the number of parameters to provide confidence in the information increases in a different context. This leads to an increase in calculation complexity. When the complexity of the network increased, in other words when the environmental complexity increased, the calculation or computational complexity increased.

2.5.3 Findings and Conclusion

From Section 2.4 to Section 2.5.2, it is clear that very little work could be found about trust calculation in contexts, such as the context of cost, the context of availability, the context of reliability, the context of the response time, and the context of credibility and usefulness. One of the important dimensions of trust is context. All the trust calculation models described in Section 2.4 are described in a specific context. Consequently, there is a need for a method that enables the calculation of trust in any context. Sections 2.4 and 2.5 can be summarised as follows:

- When calculating trust, the information can be obtained from different sources, such as other peers, customers or W3C methods (standards).
- Trust in the information security context is well documented.
- To increase the confidence in the information, the number of parameters bounding the information needs to be increased when the complexity of the network grows.
- The trust calculation becomes more complex when the complexity of the network grows.
- As the context complexity increases, the calculation complexity also increases.
- When the number of parameters used for the quantification of trust increases, the trust calculation algorithm also gets more complex.
- Each trust algorithm uses different models to store and manage data while calculating trust values.
- A trust calculation method needs to be identified to calculate trust in any context.

An alternative trust calculation method which can be used in any context is necessary. The environmental complexity should not increase the complexity of trust calculation. This can be

possible, as the information sources do not change significantly when the context complexity increases. A trust calculation method, using neural networks, may be beneficial, as the neural networks use only known information.

2.6 Features and standards

From Sections 2.2, 2.3, 2.4, and 2.5, it is clear that in order to quantify trust, information can be obtained from different sources. Most of the trust algorithms, except the trust model developed by Al-Shargabi (2016), used recommendations, feedback, and human behaviour as information. Even though there was no trust quantification in the trust model developed by Al-Shargabi (2016), trust can be determined by confirming the presence of W3C methods.

In this section, another kind of situation will be introduced where the service level agreement (SLA) of an entity can be used in the trust quantification process. This will be demonstrated in Chapter 7 using the experimental design developed in Chapter 5. Many examples can be found where the SLA can be used in trust calculation. The SLA of several products, including the Amazon Relational Database Service (ARDS), will be gathered to identify what kind of information is available in the SLA. The ARDS of Amazon (2019) will be used in Chapter 7 for the above-mentioned demonstration. Table 2-1 shows the SLA of several products.

Company	**Product**	**MUP**	**Service Credit Percentage**				
			10 %	25 %	30 %	50 %	100 %
Google	Google Cloud Storage (GCS) (Multi-Regional Storage class) (GoogleCloud, 2018a)	>= 99.95%	< 99.95%	< 99.0%	N/A	< 95.0%	
Google	GCS (Regional Storage class) (GoogleCloud, 2018a)	>= 99.9%	< 99.9%	< 99.0%	N/A	< 95.0%	
Google	GCS (Nearline, Coldline and Durable Reduced Availability Storage classes) (GoogleCloud, 2018a)	>= 99.0%	< 99.0%	< 98.0	N/A	< 95.0%	

Table 2-1: Monthly uptime percentage of different entities

Company	Product	MUP	Service Credit Percentage				
			10 %	25 %	30 %	50 %	100 %
Google	Google Cloud Datastore (GCD) (Multi-Region) (GoogleCloud, 2018b)	>= 99.95%	< 99.95%	< 99.0%	N/A	< 95.0%	
Google	GCD (Regional) (GoogleCloud, 2018b)	>= 99.9%	< 99.9%	< 98.0%	N/A	< 95.0%	
Microsoft Azure	Cloud Services (Azure, 2008)	>=99,95%	< 99.95%	<99.0%			
Microsoft Azure	Load Balancer (Azure, 2008)	>=99,95%	< 99.95%	<99.9%			
Microsoft Azure	Virtual Machines (Azure, 2008)	>=99,95%	< 99.95%	<99.0%			
Amazon	ARDS (Amazon, 2019)	>=99.95%	< 99.95%	<99.0%	N/A	N/A	<95.0%
Amazon	ARDS (Amazon, 2018a)	>=99.95%	< 99.95%	<99.0%	N/A		
Amazon	ASSS (Amazon, 2018b)	>=99.9%	<99.9%	<99.0%	N/A		

Table 2-1: Monthly uptime percentage of different entities (Continued)

From Table 2-1, it is clear that most of the electronic entities provide a Monthly Uptime Percentage (MUP) and service credit percentage to a customer as an SLA. In the event of the electronic entity not meeting the SLA, the customers will be eligible to receive a service credit

calculated in percentage as compensation. The SLA can also be considered as a source to collect information for trust quantification. If the electronic entity again failed to give the required compensation (service credit calculated), the trust value must be reduced. In trust calculation of these kinds of situation, one information source can be the success or failure of the SLA. To increase the confidence in the information for trust calculation, the number of parameters bounding the information can be increased.

In addition to the SLA, each electronic entity has its features and standards in the specific context. To identify the parameters bounding the information (the success or failure of the SLA) for the trust calculation of the ARDS, the features offered by the ARDS will be gathered, as the ARDS of Amazon (2019) will be used in Chapter 7 for the demonstration.

The ARDS has different features in the specific context, such as lower administrative burden, performance, scalability, availability and durability, security, manageability, and cost-effectiveness. In the context of availability and durability, the ARDS offers features like automated backups, database snapshots, Multi-AZ deployments, and automatic host replacement. In the trust calculation of the ARDS, information other than the success or failure of the SLA can also be included. In addition to the success or failure of the SLA (information), the feedback in terms of satisfaction regarding the MUP of features (information) can also be included in the trust calculation. The information source will be the SLA and the users of the ARDS. For the demonstration in Chapter 7, the context will be restricted to the context of availability and durability. To increase the confidence in the information for trust calculation, the following parameters will be used:

- The number of features offered in the context of availability and durability
- Percentage availability or percentage success of automated backups in terms of satisfaction in MUP
- Percentage availability or percentage success of database snapshots in terms of satisfaction in MUP
- Percentage availability or percentage success of Multi-AZ deployments in terms of satisfaction in MUP
- Percentage availability or percentage success of automatic host replacement in terms of satisfaction in MUP
- The number of transactions that took place so far
- Credit return success or failure of the transaction
- The number of credit returns failed so far
- Trust values that are achieved due to previous transactions or performance

In summary, the ARDS offers an MUP of not less than 99.95% in its latest SLA (Amazon, 2019). In the case of the ARDS, the standards are defined in terms of the MUP. The SLA provides an MUP to a customer. When there is a failure to meet the offered MUP, the customers should receive the agreed service credit calculated in percentage. The standards are described in terms of the MUP of the features. Therefore, the trust value must decrease when the percentage availability or the satisfaction received from customers goes down. There must also be a reduction in trust value if there is a failure in the offered credit return. The amount of the reduction of trust value must increase with every credit return failure. The trust value must go down when the percentage availability goes down. More details regarding the trust calculation for the ARDS (Amazon, 2019) will be explained in Chapter 7.

2.7 Conclusion

In this chapter, a literature study to determine various definitions available on trust was presented. The three dimensions of trust were identified by finding the commonalities and differences between various definitions. A new comprehensive definition of trust was given, using the three dimensions of trust. In addition, the examples of trust algorithms, the characteristics of trust dimensions, and the features and standards offered by a couple of products were addressed. In Chapter 3, the radial basis function neural network (RBFNN) model which is proposed as an alternative trust calculation method in this study will be considered.

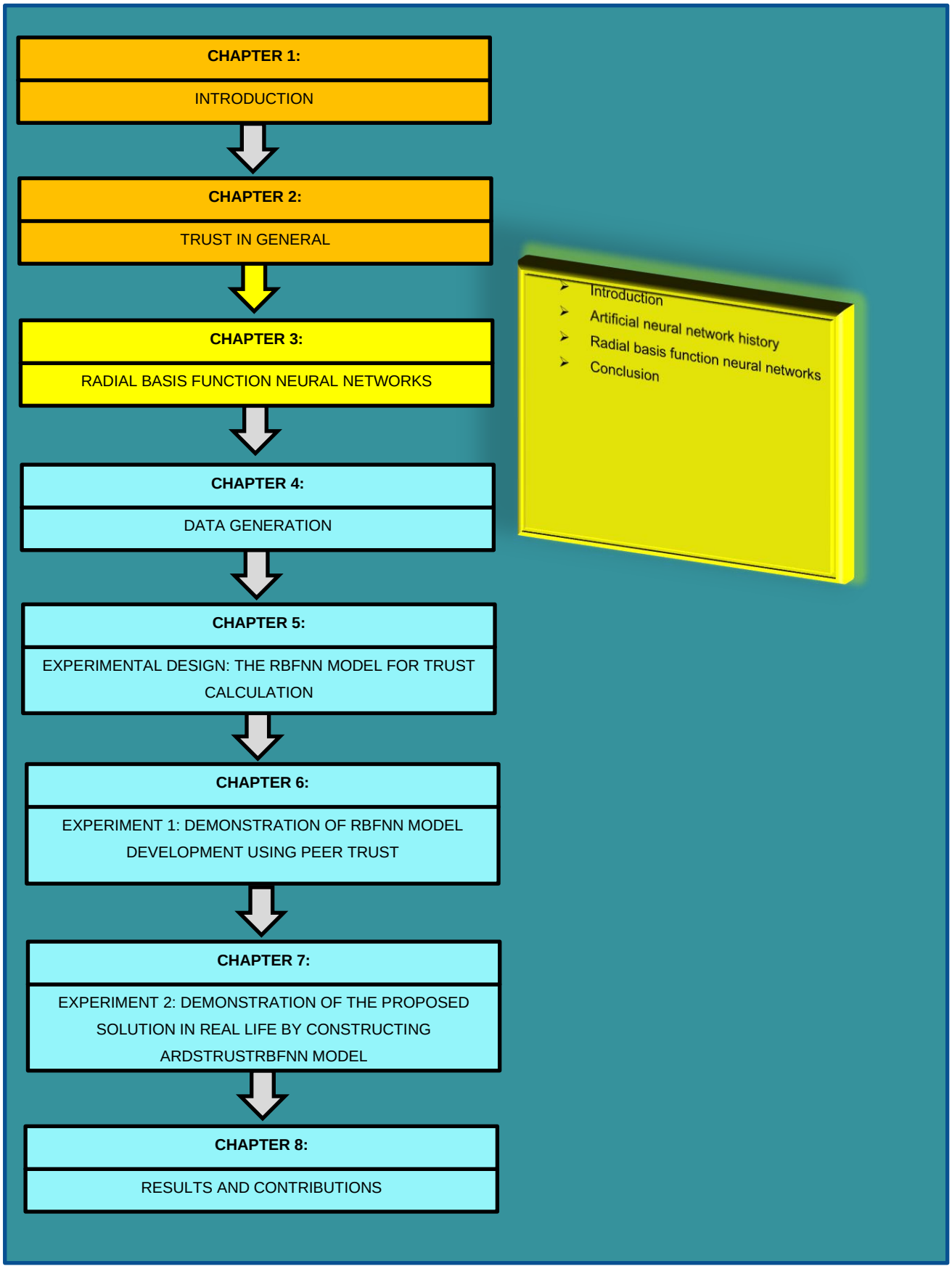


Figure 2-2: Signpost diagram of Chapter 3

CHAPTER 3 RADIAL BASIS FUNCTION NEURAL NETWORKS

3.1 Introduction

Artificial intelligence (AI) is a relatively young field with many successful real-world applications, and research in the field has increased dramatically since 2012. In the early years, AI was used to solve problems that were intellectually difficult for human beings, but which could be described by mathematical rules. There are, however, some problems that are relatively easy to solve for a human while difficult to describe by mathematical rules. Examples include recognising faces in images and recognising words. These kinds of problem may be solved if computers are able to learn from experience. To solve such problems, the computer must understand the world in terms of a hierarchy of concepts. Each concept must be defined in terms of its relation to simpler concepts, and this will allow the computer to learn complicated concepts by building on simpler ones. Getting informal knowledge into a computer is one of the key challenges in AI. The capability of AI systems to acquire their knowledge by extracting patterns from the raw data is called machine learning (Goodfellow *et al.*, 2016).

Artificial neural networks (ANNs), a subfield of machine learning, commonly referred to as neural networks, have been motivated by the findings that human brains work entirely differently from the digital computer (Haykin, 1994). A neural network is a massively parallel distributed processor made up of simple processing units which can acquire knowledge from the environment through a learning process and can store the knowledge using synaptic weights (Haykin, 1994). Non-linearity, input-output mapping, adaptability, evidential response, contextual information, fault tolerance, uniformity of analysis and design, and neurobiological analogy are some of the properties and capabilities of neural networks. ANNs are very powerful modelling techniques. The generalisation property of the neural network ensures reasonable output for the given input. A perceptron, feed-forward neural network, radial basis function neural network (RBFNN), deep feed-forward network, and recurrent neural network are some of the different types of neural network. ANNs have been successfully applied in application areas, such as aerospace, automotive, banking, defence, electronics, entertainment, financial, insurance, manufacturing, medical, oil and gas, robotics, speech, securities, telecommunications, and transportations industries (Hagan *et al.*, 2014).

The remainder of this chapter is organised as follows. In Section 3.2, the ANN history will be considered. This will include a discussion on its biological inspiration, biological neurons and artificial neurons, artificial neural network architecture and a discussion on the linearly separable and non-separable problems. The general concept of an RBFNN will be explained in Section 3.3. This will include the architecture of an RBFNN, a general discussion on the advantages and

disadvantages of an RBFNN, a discussion on trust calculation as a non-separable linear problem as a motivation for using an RBFNN to model trust and the training of an RBFNN. Time series prediction using an RBFNN will be explained at the end of Section 3.3. The chapter will be concluded in Section 3.4.

3.2 Artificial neural network history

In 1943, McCulloch and Pitts presented the first mathematical model of a neuron (McCulloch & Pitts, 1943). In this first model, a neuron is an element with several inputs and a single output (Ling & Bo, 1999). The neuron is called the fundamental processor of the neural network. It has three basic elements: a set of connecting links, a summation of the links and an activation function. In the late 1950s, Rosenblatt developed a perceptron neural network which is the simplest form of a neural network (Rosenblatt, 1961; Tappert, 2019). It consists of a single neuron and several adjustable weights. A perceptron neural network can classify data into two classes. In 1969, Minsky and Papert pointed out some of the limitations of the perceptron and showed that a two-layer feed-forward network could overcome many of these restrictions. Rumelhart developed the backpropagation algorithm that is most often used to train a feed-forward network (E. Rumelhart *et al.*, 1986; Rumelhart *et al.*, 1995; Sukhan, 1988).

3.2.1 Biological inspiration

A nervous system can be defined as a network of cells that receives information from the internal and external environment, integrates the information and transmits the information to other neurons and effector organs (Eluyode & Akomolafe, 2013). The neuron (Figure 3-1) is the fundamental unit of a nervous system (Eluyode & Akomolafe, 2013; Jadid & Fairbairn, 1996). A neuron consists of dendrites, the axon and the cell body which is also called the soma. The cell body includes the oval-shaped nucleus, which contains information about hereditary traits (Agatonovic-Kustrin & Beresford, 2000; Jain *et al.*, 1996).

The dendrite structure of the neuron acts as the primary receptor of the neuron as a neuron receives signals from other neurons (Basheer & Hajmeer, 2000; Hagan *et al.*, 2014; Jain *et al.*, 1996). The axon is the transmitter which is longer than the dendrites, and it receives signals from the cell body and then transmits them to the next neuron (Basheer & Hajmeer, 2000; Hagan *et al.*, 2014; Jain *et al.*, 1996). The Myelin sheath is a protective fatty coating that covers the axon and acts as an insulator to keep the electrical signal inside the cell. This insulation will cause the signal to move more quickly. The connector between the axon terminals of the transmitter neuron and the dendrites of the receiver neuron is called a synapse (Basheer & Hajmeer, 2000; Jain *et al.*, 1996).

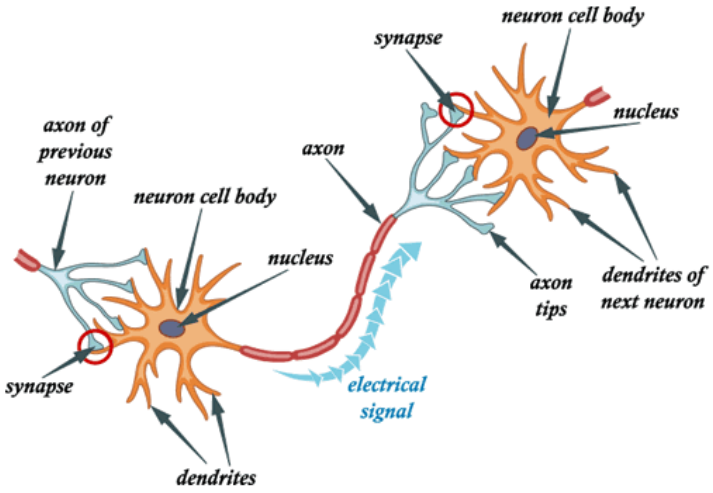


Figure 3-1: Connection between two biological neurons (Preetham, 2016)

The detailed structure of the biological neuron is shown in Figure 3-1. The connection between a biological neuron and an artificial neuron will be discussed in the next section.

3.2.2 Biological neuron and artificial neuron

The inspiration for the artificial neural network came from the biological neural network (Haykin, 1994; Jadid & Fairbairn, 1996). Dendrites and the axon can be considered as the connections between the nodes. One can consider the inputs of the neural network as the dendrites of the biological neural network, and the outputs of the neural network can be considered as the axon. The connection weights of the network represent the synapses. The neuron itself represents the cell body soma. An artificial neuron can be considered as the building component of an ANN which is designed to simulate the biological neuron's function. ANNs can be linked to the working of the human brain, and they can be loosely considered as the digitised model of a human brain (Agatonovic-Kustrin & Beresford, 2000). ANNs learn through experience, and they will collect knowledge by detecting the patterns and the relationships in the data. Compared to an ANN, the human brain is more complex and contains many more neurons and is much faster in performing many tasks (Agatonovic-Kustrin & Beresford, 2000; Hagan *et al.*, 2014). The human brain can also generate new ideas which were previously unknown (Agatonovic-Kustrin & Beresford, 2000; Kushiro *et al.*, 2013). The first artificial neuron by McCulloch and Pitts was based on the properties of a biological neuron (Maan *et al.*, 2017; McCulloch & Pitts, 1943). A summarised comparison between a biological neural network and an ANN is given in Table 3-1.

Biological neural network	Artificial neural network
Cell body	Node
Dendrites	Inputs
Axon	Outputs
Synapse	Weights

Table 3-1: Comparison between biological and artificial neural networks

The artificial neural network architecture which forms the basis of the proposed RBFNN model that calculates trust values will be discussed next.

3.2.3 Artificial neural network architecture

In ANNs, the artificial neurons form the nodes and weighted directed edges form the connections between the input and output neurons (Jain *et al.*, 1996). An ANN can be divided into a feed-forward network and a recurrent or feedback network, based on the direction of the flow of information.

In a feed-forward network, data enters at the input layer. Then it passes to the next layers until it reaches the output layer of the network. It is called a feed-forward network because there will be no feedback between layers. A feed-forward network does not have a connection back from the output to the input (Agatonovic-Kustrin & Beresford, 2000). It will never keep a record of the previous output. Feed-forward networks will give only one set of output values; hence the network is static (Jain *et al.*, 1996). The network's response to an input is independent of the previous network state; hence the networks are memoryless.

In a feedback neural network, the output of one layer will be given as the input to the previous layer or the same layer (Agatonovic-Kustrin & Beresford, 2000). A feedback network will keep a record of the previous output. The output will depend upon the current input and the previous state of the network.

There are many types of ANN architecture, such as the perceptron, single-layer perceptron, multi-layer perceptron and networks with feedback loops. A multi-layer feed-forward network has one or more hidden layers. In a multi-layer perceptron, the output of one layer will be the input of the following layer. The output of the last layer of the network is the network output.

The development of the backpropagation learning algorithm which is the process of adjusting the weights in a multi-layer perceptron to reduce the difference between the network's actual

output vector and the desired output made these networks more popular (Rumelhart *et al.*, 1986). The evaluation between the network's actual output vector and the desired output vector is done by a cost function which can be the mean squared error. The error which is computed at the output layer will be transferred back to the hidden layer and finally to the input layer to adjust the weights of the network. The backpropagation learning algorithm will specify the cost function, and it will aim to minimise the cost function by adjusting the weights and biases (Rumelhart *et al.*, 1995).

In Figure 3-2, an example of an ANN that consists of the input layer, hidden layer and the output layer is shown.

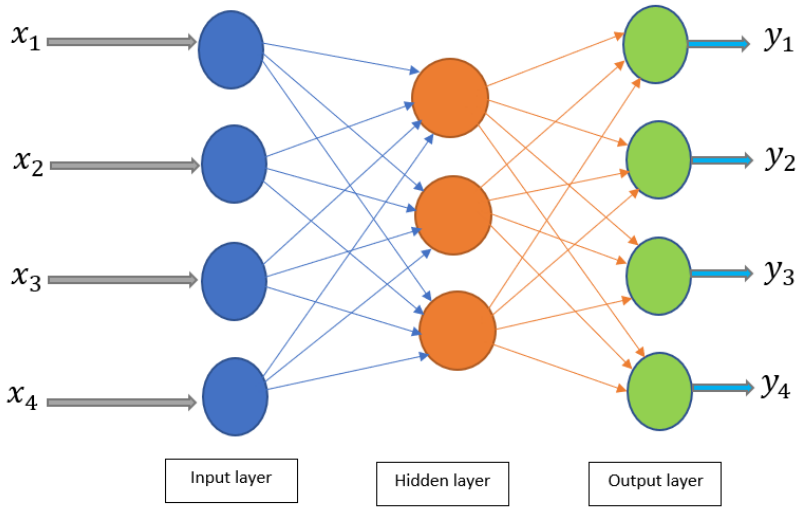


Figure 3-2: An artificial neural network

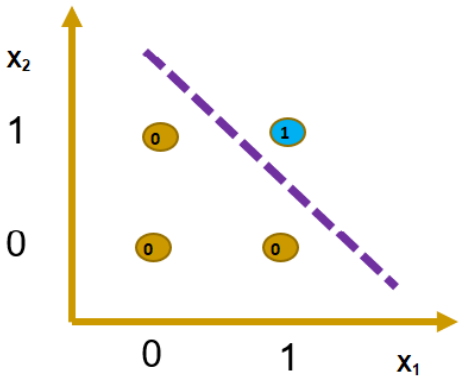
Since a multi-layer perceptron may be used for non-separable linear problems, linearly separable and non-separable problems will be discussed in the next section.

3.2.4 Linearly separable and non-separable problems

If two classes of values can be separated by a straight line or, in general, by a hyperplane, then it can be defined as a linearly separable problem (Roychowdhury *et al.*, 1995). Examples of linear separable problems are the AND and OR functions (Figure 3-3). With these problems, a single line can separate the different classes of output values.

AND Function

X_1	X_2	Output
0	0	0
0	1	0
1	0	0
1	1	1



OR Function

X_1	X_2	Output
0	0	0
0	1	1
1	0	1
1	1	1

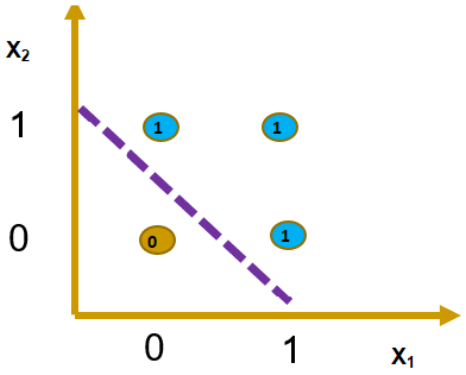


Figure 3-3: The AND function and OR function

In the case of the XOR function (Figure 3-4), two lines are needed to separate the output values. The function is therefore not linearly separable and cannot be solved by a single-layer perceptron. A feed-forward neural network with at least one hidden layer would be able to solve it.

XOR Function

X_1	X_2	Output
0	0	0
0	1	1
1	0	1
1	1	0

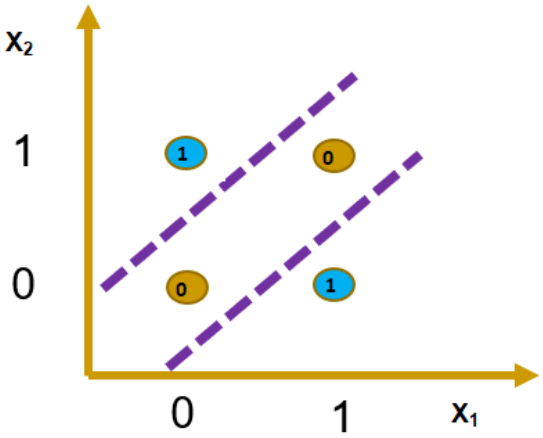


Figure 3-4: The XOR function

In the next section, a discussion on the radial basis function neural network (RBFNN) will be presented.

3.3 Radial basis function neural networks

There are many types of feed-forward neural networks, such as the multi-layer perceptron and the RBFNN. Compared to the multi-layer perceptron network, RBFNNs are simpler. The training process in RBFNNs is faster than the multi-layer perceptron network. Compared to traditional neural networks, the RBFNN is more robust and have better tolerance to input noises. RBFNNs can respond well to patterns that are not used for training (Yu *et al.*, 2011). The RBFNN has gained widespread appeal in a wide variety of application domains. This model was proposed in 1973 by Duda and Hart (Duda *et al.*, 2000). Due to the fast-learning capability compared with other feed-forward networks, the RBFNN is considered a good model for approximation problems (Chien-Cheng *et al.*, 1999; Ruslan *et al.*, 2013). RBFNNs have been widely used as a universal function approximator and to solve non-linear problems because of their simple topology structure and the ability to reveal how learning proceeds in an explicit manner (Lin & Wu, 2011). The RBFNN can convert a non-separable linear problem into a linearly separable problem. Broomhead and Lowe (1988) explained the procedure for the design of the layered feed-forward networks using a radial basis function. RBFNNs are used in many applications, e.g. non-linear system identification and time-series predictions (Broomhead & Lowe, 1988; Moody & Darken, 1989). Based on these advantages, the RBFNN is chosen as the alternative trust calculating technique in this research.

3.3.1 The architecture of the RBFNN

An RBFNN is a feed-forward type of artificial neural network. The RBFNN consists of three layers, namely one input layer, one hidden layer and one output layer (Aziz & Abdullah, 2009; Billings & Zheng, 1995; Dubey, 2015; Haykin, 1994; Neruda & Kudová, 2005; Shakya *et al.*, 2011; Xie *et al.*, 2011; Yu *et al.*, 2011). An RBFNN can solve complex pattern classification problems, and this property can be used in trust value classification. This solution is obtained by transforming the problem into a high dimensional space by applying a non-linear transfer function.

The structure of the RBFNN is shown in Figure 3-5 where P denotes the number of the input features, M the number of nodes in the hidden layer and C the number of nodes in the output layer. M can also be considered as the increased dimensionality to convert a non-separable linear problem into a linearly separable problem.

There can be more than one predictor variable $x_1, x_2, \ldots, x_P$ in the input layer, as shown in Figure 3-5. Each predictor variable will be associated with an independent neuron (Dash *et al.*, 2016). The output of each input layer neuron is fed forward to each neuron in the hidden layer. Each neuron in the hidden layer consists of a radial basis function which is a non-linear activation function, and it will be centred at a point (Dash *et al.*, 2016; Khazaei *et al.*, 2017; Shakya *et al.*, 2011). The Gaussian function is commonly used as the radial basis function (Haviluddin & Tahyudin, 2015; Rivas *et al.*, 2004; Rojas *et al.*, 2000). A change in dimensions can vary the spread or radius for the radial basis function.

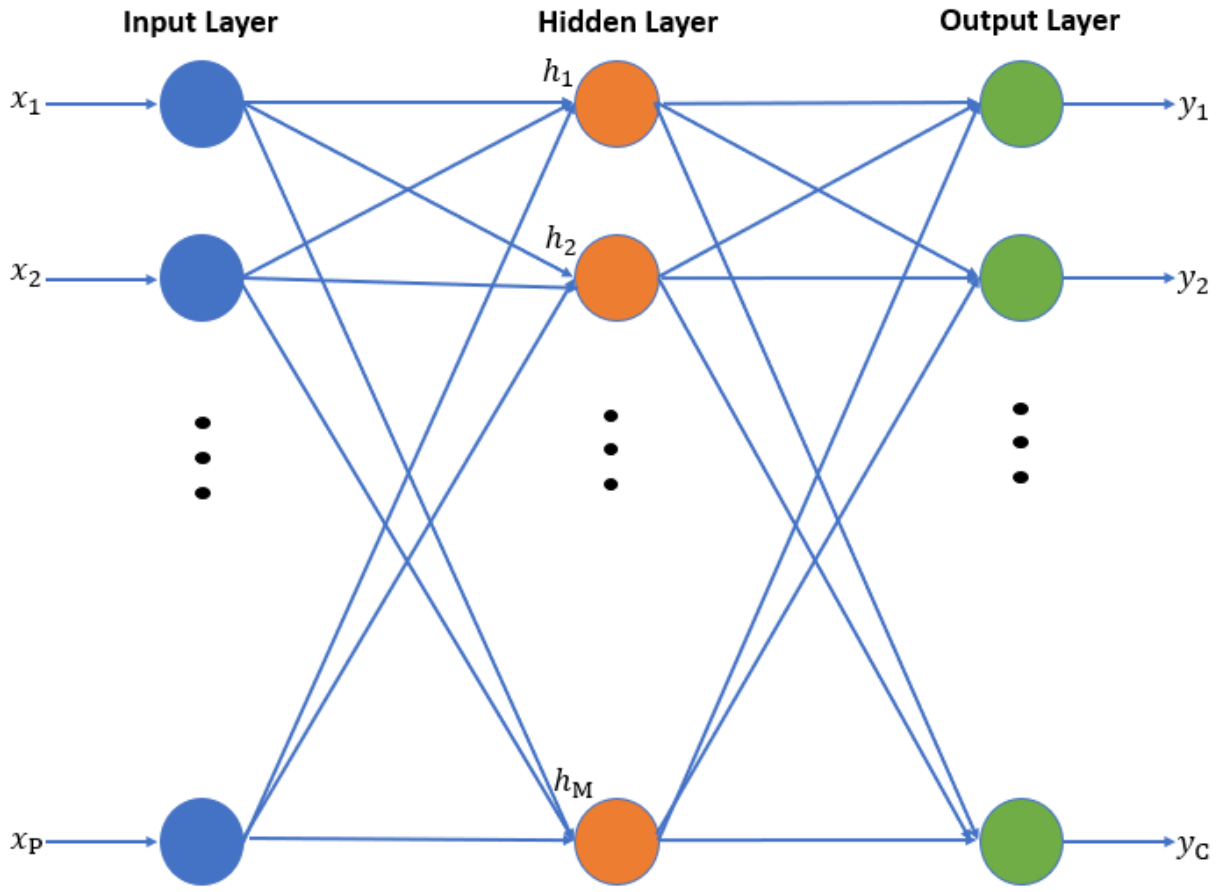


Figure 3-5: RBFNN structure

The single hidden layer performs the non-linear transformation from the P-dimensional input space to the M-dimensional space by increasing the dimension using the M radial basis functions (Bezerianos *et al.*, 1999). The radius or the spread of the radial basis function will depend upon the dimension (Dash *et al.*, 2016). The centres and the spreads are determined during the training process of the network. The output layer of the RBFNN will always be linear

(Bezerianos *et al.*, 1999; Rojas *et al.*, 2000; Sossa *et al.*, 2014). The number of nodes in the output layer will be the same as the number of classes. Each neuron in the output layer of the network will be associated with a weight. The values of the hidden layer neurons will be multiplied by the weight associated with the output layer neuron, and this product will be given to the output layer. The nodes in the output layer perform the linear combinations of the outputs of the hidden layer. Finally, classification is only done in the output layer. The advantages and disadvantages of the RBFNN will be described in the next section.

3.3.2 Advantages and disadvantages of RBFNN

The simple topological structure is one of the advantages of using an RBFNN. A faster learning capability of the RBFNN and better approximation capability compared to traditional neural networks are other advantages (Aziz & Abdullah, 2009; Chien-Cheng *et al.*, 1999; Erol *et al.*, 2008; Jayawardena *et al.*, 2006; Qasem *et al.*, 2013; Shakya *et al.*, 2011; Sossa *et al.*, 2014; Xie *et al.*, 2011). An RBFNN is considered as a universal approximator (Haykin, 1994). Some other advantages include a good generalisation property, pattern recognition, strong tolerance to input noise and online learning ability (Dubey, 2015; Er *et al.*, 2002; Yu *et al.*, 2011). Due to the three-layer structure, an RBFNN is much easier to design (Yu *et al.*, 2011). An RBFNN's strong tolerance to the input noise increases the designed system's stability. The curse of dimensionality is one of the disadvantages of using an RBFNN which is referred to as the number of basis functions required that will increase as the dimension of the input space increases (Bezerianos *et al.*, 1999; Hagan *et al.*, 2014). Another disadvantage is that the runtime speed of an RBFNN can be low due to a large number of hidden units in many problems (Jain *et al.*, 1996).

3.3.3 Trust calculation using an RBFNN

Trust calculation is done by accessing information from different sources, and the complexity of trust calculation increases as the network complexity and the context complexity increase. Trust values and the number of parameters used for the calculation of trust will depend upon the context and the trust algorithm. The feedback a peer obtains from another peer, the total number of transactions that a peer has with another peer, the credibility of the feedback source, the transaction context factor and the community context factor are the information used by the PeerTrust algorithm to calculate trust (Li & Ling, 2004). The hybrid algorithm of Daskapan *et al.* (2008) used three parameters, namely the feedback in terms of satisfaction a peer obtains from other peers, the total number of transactions a peer has with other peers and the credibility of the feedback sources. Moreover, the non-symmetric and non-transitive property of trust gives a wider context possibility for trust calculation, and this leads to the use of any number of

parameters in the calculation of trust. Trust can therefore be calculated by using any number of variables, depending upon the context. The trust value will never be constantly increasing or decreasing, as it can depend upon the level of past and latest interactions. This is a non-monotonicity property of the trust value. The wider context possibility for trust and the different levels of expectation of each entity to trust gives the subjectivity property for the trust value. The trust value is not static, since a new trust value will be calculated when a new transaction occurs. A trust value is dynamic. A depreciation of the previously calculated trust value may occur if no interactions are happening for a long time. Moreover, the future value of trust is difficult to predict. The trust value has the properties of context-dependency, dynamicity, non-symmetry, non-transitivity, non-monotonicity, subjectivity, uncertainty and temporal decay (Gambetta, 1988; Grandison & Sloman, 2002a; Grandison & Sloman, 2002b; Hang *et al.*, 2012; Manna *et al.*, 2016; Mohsenzadeh & Motameni, 2015; Olmedilla *et al.*, 2006; Tran *et al.*, 2005). As stated previously, trust can be calculated using any number of variables depending upon the context. The non-monotonicity property of the trust values makes it difficult to fit a hyperplane between any two sets of trust values. There is a need for more than one hyperplane to separate each class of trust values. The calculation of trust value can be considered as a non-separable linear problem. The ability of an RBFNN to convert a non-separable linear problem to a linearly separable problem can be used for trust calculation. Time series prediction, using an RBFNN, will solve the uncertainty regarding future trust values. The training of the RBFNN will be discussed in the next section.

3.3.4 Training of the RBFNN

An RBFNN needs two levels of training: hidden layer training and output layer training (Awchi, 2008; Aziz & Abdullah, 2009; Hagan *et al.*, 2014; Shakya *et al.*, 2011). The first level involves determining the number of radial basis functions and the corresponding centres, and the second level establishes the output layer weight matrix (Raitoharju *et al.*, 2016; Yao *et al.*, 2006). Before explaining the training of hidden layer nodes and training of weight vectors, a brief description will be given regarding the structure of radial basis functions and the *K*-Means clustering algorithm.

Radial basis functions

Let the original dimension of input feature vectors be P. To convert a non-separable linear problem to a linearly separable problem, increase the dimension of the feature vectors to dimension M.

Let $\boldsymbol{x}$ be a given feature vector. Then

$$\Phi(\boldsymbol{x}) = [\Phi_1(x), \Phi_2(x), \Phi_3(x), \Phi_4(x), \dots, \Phi_M(x)]^t \quad (3\text{-}1)$$

Each of the Φ functions will produce a real value. The Φ functions $\Phi_1, \Phi_2, .., \Phi_M$ are defined as radial basis functions. Every radial basis function has a receptor $\boldsymbol{t}$ (Mongillo, 2011). While moving radially away from the receptor, with the radius defined as $\|\boldsymbol{x} - \boldsymbol{t}\|$, the value of the function will go on increasing or decreasing. The value of the function will be either maximum or minimum at the point $\boldsymbol{t}$. The value of Φ in each of these concentric circles will be constant. There are different types of radial basis functions (Billings & Zheng, 1995; Haykin, 1994; Mongillo, 2011; Wettschereck & Dietterich, 1992). Some examples are the following:

- Multi-quadratic radial basis function: $\Phi(\mathrm{r}) = (r^2 + c^2)^{\frac{1}{2}}$, $\mathrm{c} > 0$, where r is the radius and c is a constant. In a multi-quadratic radial basis function at $r = 0$ where $\boldsymbol{x}$ and $\boldsymbol{t}$ coincide, the value of the radial basis function c is a minimum. When the value of r increases, the value of the radial basis function also increases.
- Inverse multi-quadratic radial basis function: $\Phi(\mathrm{r}) = \frac{1}{(r^2 + c^2)^{\frac{1}{2}}}, c > 0,$ where r is the radius and c is a constant. In an inverse multi-quadratic radial basis function at $r = 0$, the value of the radial basis function is $1/c$ which is the maximum. When the value of r increases, the value of the radial basis function decreases.
- Gaussian function radial basis function: $\Phi(\mathrm{r}) = \exp\left[-\frac{r^2}{2\sigma^2}\right], \sigma > 0,$ where σ indicates the spread of the radial basis function. In a Gaussian radial basis function at the receptor $\boldsymbol{t}$, the value of the radial basis function will be a maximum. A Gaussian radial basis function will be used in this study, as it is the most popular type of RBF.

The *K*-Means clustering algorithm can be used to find the number of neurons in the hidden layer and the cluster centres (Ocampo-Vega *et al.*, 2016; Pislaru & Shebani, 2014; Raitoharju *et al.*, 2016).

The *K*-Means clustering algorithm

MacQueen proposed the *K*-Means clustering algorithm (Li & Wu, 2012; MacQueen, 1967) to find a feasible method of computing an optimal partition of the given training sample. The following steps can be used to produce K clusters with centres:

1. Divide the given training sample randomly into K samples.

2. Randomly select the cluster centre for each of the clusters which will be representative of that cluster.

3. For each of the sample data points determine which cluster centre is nearest to that sample.

4. Move the data point to the cluster whose centre is nearest to it. This step must be done for every data point until all the data points are moved to their nearest cluster centre.

5. Select the new cluster centre for each of the clusters by taking the average of all data points assigned to each cluster centre.

6. If no cluster was updated, exit the algorithm, else go to Step 3.

In the next section, the training of the hidden layer nodes of the RBFNN is described.

3.3.4.1 Training of hidden layer nodes

As each node in the hidden layer implements a radial basis function, the training of the hidden layer includes determining the cluster centres or RBF centre values and the spread of the radial basis function which is the value for σ in each of the nodes in the hidden layer. This value is calculated by taking the average of the distance between the cluster centre and the training instances in the specific cluster (Awchi, 2008).

The number of nodes in the hidden layer must also be determined. Too few hidden layer neurons will lead to a poor generalisation capability and too many hidden layer neurons will lead to an overly complex network structure and can also cause a poor generalisation capability (Yan *et al.*, 2005). The process can start from a relatively small number of hidden layer neurons. Then the number of neurons in the hidden layer is gradually increased until achieving the precision requirement (Yan *et al.*, 2005). Once the number of hidden nodes in the hidden layer has been decided, the cluster centres must be identified (Yan *et al.*, 2005). A random method, grid method and clustering methods are some of the techniques used for finding the RBF centres (Hagan *et al.*, 2014; Yan *et al.*, 2005). The *K*-means clustering algorithm, which is explained in Section 3.3.4, can be used to identify the RBF centres (Dubey, 2015; Yan *et al.*, 2005)*.*

Let N be the number of training vectors $\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3, \ldots, \boldsymbol{x}_N$ and P the dimension of each of the training vectors. To transform these vectors to a vector of the dimension of M, each of the hidden layer nodes must have a receptor $\boldsymbol{t}$ and spread σ. That means for the j^{th} node the

receptor will be $\boldsymbol{t_j}$ which can be determined by *K*-means clustering, and the spread of the radial basis function will be σ_j where, $j = 1, 2, \ldots, M$. The spread is defined as follows:

$$\sigma_j = \sqrt{\frac{1}{\mathrm{P}} \sum_{i=1}^{P} \left(t_j - t_i\right)^2} \tag{3-2}$$

where P is the dimension of the input vector.

Each node in the hidden layer will have a receptor t_i which is be determined by *K*-means clustering and spread σ_i. These values will enable the calculation of the radial basis function $\Phi_i(\boldsymbol{X})$ for each node in the hidden layer. If there are M hidden nodes in the hidden layer, the radial basis function $\Phi(\boldsymbol{X})$ for each node $\Phi_1(\boldsymbol{X})$, $\Phi_2(\boldsymbol{X})$, $\Phi_3(\boldsymbol{X})$, … $\Phi_M(\boldsymbol{X})$ can be calculated using t_i and σ_i where $i = 1, 2, \ldots, M$. The input feature vector $\boldsymbol{X}$ of P dimension is converted to an M dimensional vector. Training of the weight vectors will be explained in the next section.

3.3.4.2 Training of the weight vectors

The second level of training in an RBFNN is the training of the weight vectors. The weight vectors connect the output of the hidden layer and the output layer. The i^{th} node in the hidden layer will be connected to the j^{th} node in the output layer with a connection weight w_{ij}. With these weights, every node in the output layer computes a linear combination of the outputs of the hidden layer. Based on the value of linear combinations, the output layer nodes determine to which class the input vector should be classified. If the input feature vector belongs to class 1 for example, then only the output of the first node in the output layer y_1 in Figure 3-5 will be 1 and all other output nodes' output will be 0. The output of the *j*th output neuron can be given as

$$\mathrm{C}_j = \sum_{i=1}^{M} \mathrm{w}_{ij} \Phi_i(\mathbf{X}) = \sum_{i=1}^{M} \mathrm{w}_{ij} \exp\left[-\frac{||\,\mathbf{x} - \mathrm{t}_i||^2}{2{\sigma_i}^2}\right] \tag{3-3}$$

where M is the number of neurons in the hidden layer and w_{ij} is the weight between the i^{th} neuron in the hidden layer and the j^{th} neuron in the output layer which can be determined by the Pseudo-Inverse method, $\boldsymbol{X}$ is an input vector and t_i and σ_i are the centre and the spread of the i^{th} neuron in the hidden layer, respectively (Raitoharju *et al.*, 2016).

The output can be represented in matrix format as follows:

$$\begin{bmatrix} \Phi_{11} & \Phi_{21} & \dots & \Phi_{M1} \\ \Phi_{12} & \Phi_{22} & \dots & \Phi_{M2} \\ \vdots & \vdots & \dots & \vdots \\ \Phi_{1N} & \Phi_{2N} & \dots & \Phi_{MN} \end{bmatrix} \begin{bmatrix} w_{1j} \\ w_{2j} \\ . \\ . \\ w_{Mj} \end{bmatrix} = \begin{bmatrix} b_{1j} \\ b_{2j} \\ . \\ . \\ b_{Nj} \end{bmatrix}, \quad (3\text{-}4)$$

where $b_{ij} = 1$ if X_i is an element of class $\mathrm{W_j}$ and $b_{ij} = 0$ if X_i is not an element of class $\mathrm{W_j}$ where $\mathrm{W_j}$ is the output feature vector class. In addition,

$$\Phi W_j = b_j. \quad (3\text{-}5)$$

If Φ^+ is the pseudo-inverse of the matrix Φ, one can obtain the weights by using the formula

$$W_j = \Phi + b_j \text{ (Haykin, 1994).}$$

The goal of determining the weights is to minimise the error function between the obtained output and the target output (Neruda & Kudová, 2005; Raitoharju *et al.*, 2016).

3.3.5 Time series prediction using an RBFNN

Time series prediction estimates the next value or future value in a series of values. As discussed in Section 3.3.3, trust properties, such as non-monotonicity, subjectivity and being non-static made it challenging to predict the future value of trust. Time series prediction, using an RBFNN, can solve the uncertainty regarding future trust values. Feed-forward neural networks have been widely used as time series forecasters by implementing a sliding window over the input sequence (Frank *et al.*, 2001). Time series can be defined as a set of observations x_t where each observation is recorded at a specific time t (Brockwell & Davis, 2002). Time series prediction using a sliding window can be represented by Figure 3-6 (Frank *et al.*, 2001), where $\boldsymbol{x}(t)$ is the value of the input vector at time t.

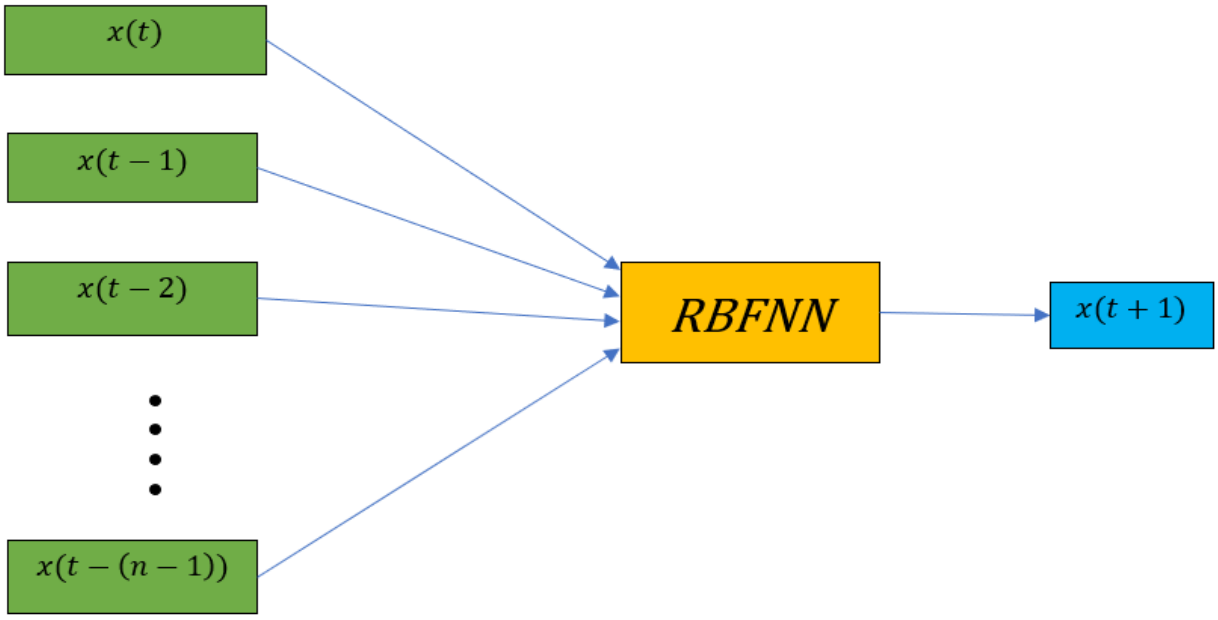


Figure 3-6: Time series prediction using a sliding window

Here, the network uses n-tuples as the inputs and produces one target output. This method can be defined as the sliding window technique because this n-tuple input slides over the complete training set.

The oldest transition must be removed from the window, and a new transition must be added to the window when the sliding window moves forward (Nori *et al.*, 2011). As an example, let T_1, T_2,…, T_8 be consecutive transactions and the sliding window size be 3. Figure 3-7 shows how a sliding window moves (Hota *et al.*, 2017).

Figure 3-7: Sliding window size 3

The sliding window technique can be used as the input to an RBFNN and to generate an input vector of x values. However, too small or large window sizes are unsuitable for time series prediction (Frank *et al.*, 2000). Therefore, in this study, a sliding window will be used to construct the input vectors of an RBFNN, as it can increase the prediction accuracy (Cui *et al.*, 2021).

3.4 Conclusion

In this chapter, artificial neural network history, the architecture of the RBFNN, the advantages and disadvantages of RBFNNs, the non-separable linearity of trust values, the training of the RBFNN and time series prediction using an RBFNN have been discussed. In Chapter 4, the focus will be on data generation. A generic approach for synthetic data generation and synthetic data generation for trust calculation specifically will be discussed.

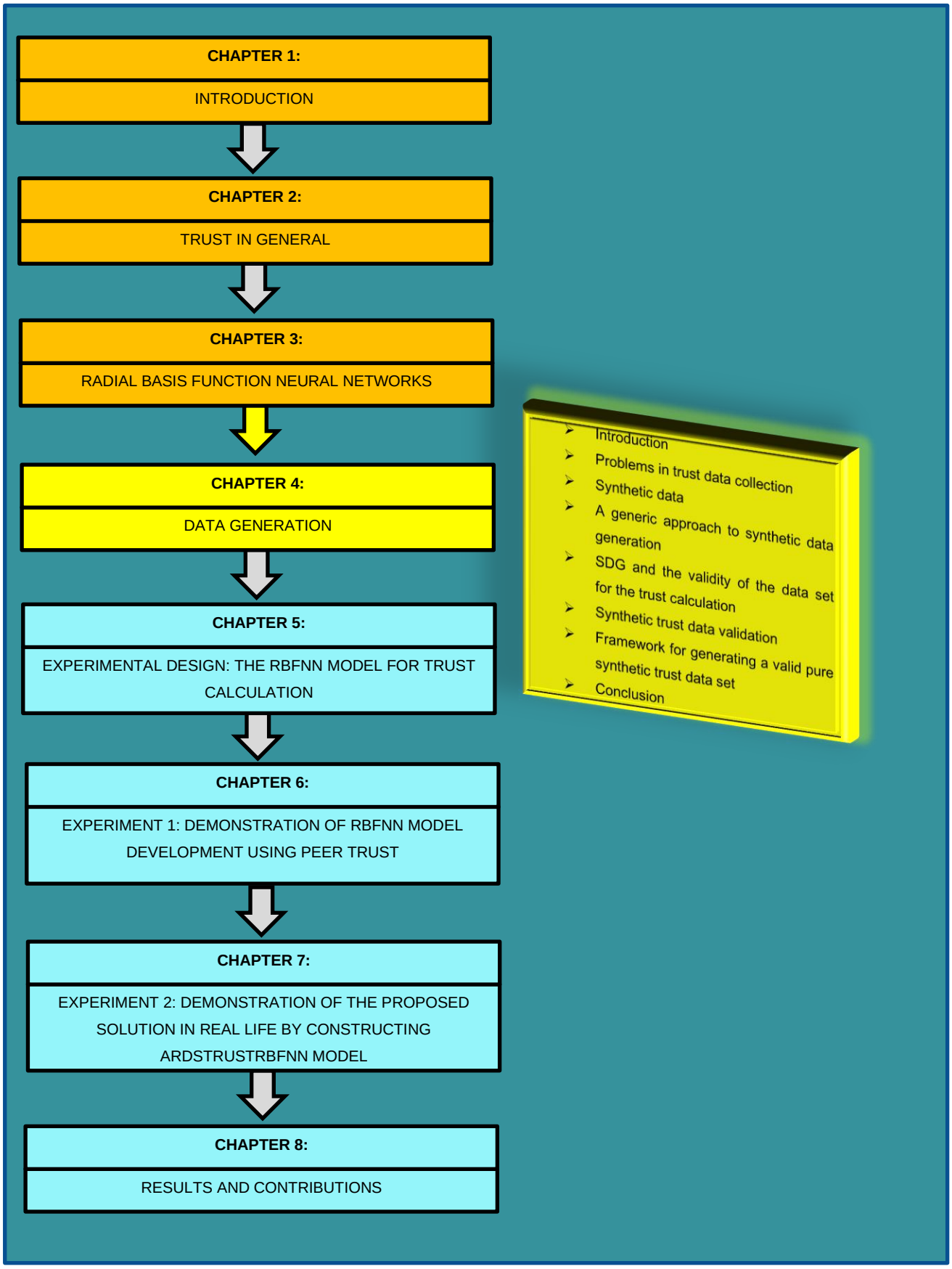


Figure 3-8: Signpost diagram of Chapter 4

CHAPTER 4 DATA GENERATION

4.1 Introduction

In this study, insufficient trust data sets could be found to build a model, using a radial basis function neural network (RBFNN) that is able to determine trust values between electronic entities. This can be due to privacy issues, data storage problems due to the need for a centralised system to manage trust data, and network problems due to the storage and access of large data sets for trust calculation. Training of an RBFNN to calculate trust between electronic entities in randomly varying situations may require large data sets. Building a trust model requires accurate training data. Therefore, obtaining training data is a significant first step in the RBFNN trust model building process. A possible solution to the scarcity of training data is to generate authentic synthetic data. In this chapter, a framework to create synthetic data which can be used to train the RBFNN model will be discussed.

The remainder of the chapter is organised as follows. In Section 4.2, problems encountered in data collection for trust calculation will be considered. The general concept of synthetic data will be explained in Section 4.3. In Section 4.4, a generic approach to synthetic data generation will be provided. Synthetic data generation for trust calculation and its validity will be discussed in Section 4.5. In Section 4.6, synthetic trust data validation will be addressed. The generic method will be extended to enable synthetic data generation for trust calculation, and this extended framework for generating a trust data set will be discussed in Section 4.7. The chapter will be concluded in Section 4.8.

4.2 Problems in trust data collection

Training of an RBFNN to calculate trust values between electronic entities in randomly varying situations requires large data sets. Traditional algorithms use data, such as recommendations, previous experience, feedback, human behaviour and presence of the World Wide Web Consortium (W3C) methods to calculate trust (Al-Shargabi, 2016; Li & Ling, 2004; Singal & Kohli, 2016; Tahta *et al.*, 2015; W3C, 2002). For this data to be used, there must be an architecture for submitting and collecting feedback, calculating trust values and managing the trust values. Each system may use different methods for storing the data and accessing the data of previous transactions (Li & Ling, 2002). There are several complications regarding this data. In general, trading data on trust is kept confidential by customers and suppliers. Data for the traditional algorithms and recommendations are difficult to obtain. The previous interaction experience is not available if someone is using a software entity for the first time. One can receive inaccurate feedback, which can affect the trust values. Access to stored data can affect

the speed of the trusted network and the trust calculation time. Besides, collecting data to calculate trust values may be time-consuming and costly. Privacy concerns may limit the disclosure of trust data. Obtaining a real-world data set for calculating trust values would enable the training of the RBFNN model. Traditional algorithms solve this problem by simulating real-world data to calculate trust (Aberer & Despotovic, 2001; Li & Ling, 2002, 2004; Wang & Vassileva, 2003). Due to the scarcity of real-world data, the data will also be generated synthetically in this study.

4.3 Synthetic data

One solution to the scarcity of real-world training data is to generate synthetic data programmatically (Anderson *et al.*, 2014; Weston *et al.*, 2015). This solution contrasts with data collected through user surveys or performing experiments. Synthetic data have been extensively studied and successfully applied across a wide range of scientific fields, which includes:

- Time-resolved quantification of causal brain-heart interplay measurement (Catrambone *et al.*, 2019)
- Synthetic traffic generation and performance evaluation of Internet Protocol Television over Ethernet Passive Optical Networks (Bhaumik *et al.*, 2015)
- Creation of an online P2P classifier (Zarei *et al.*, 2015)
- Simulation of real-time network workload (Botta *et al.*, 2012)
- Workload generation for cloud computing (Bahga & Madisetti, 2011)
- Training and testing of a fraud detection system (Barse *et al.*, 2003)
- Building deterioration models for a sewer system (Scheidegger & Maurer, 2012)
- The study of the performance of renewable energy technologies (Pillai *et al.*, 2014)
- Simulation of the effect of climate change on buildings (Van Paassen & Luo, 2002)
- Building and testing an information discovery system (Lin *et al.*, 2006)
- Application to a transportation system (Rich & Mulalic, 2012)
- Realistic workload generation for YouTube (Abhari & Soraya, 2010)
- For learning algorithms or for learning analytics infrastructures (Liu *et al.*, 2016; M. Berg *et al.*, 2016)
- Use in synthetic patients and synthetic health care records (Walonoski *et al.*, 2018)

Synthetic data can be classified as follows:

1. Fully synthetic data

An early formal process to generate fully synthetic data was published by Rubin (1993). Real-world data or aggregate data are used as input data to the synthetic data generation method, but the output data do not contain real-world data. A fully synthetic data set can protect confidentiality, as disclosing sensitive information is nearly impossible (Drechsler *et al.*, 2007).

2. Partially synthetic data

Little published a formal process to generate a partially synthetic data set (Drechsler *et al.*, 2007; Little & Liu, 2003; Reiter, 2004). Unaltered real-world data are intermixed with some form of simulated data. Only identifying variables will be synthesised to protect highly sensitive data from public disclosure, and its data utility is higher (Drechsler *et al.*, 2007; Hawala, 2008). Another way of producing partially synthetic data is to mask sensitive or private data fields as anonymous (Ghinita *et al.*, 2008; Jian-min *et al.*, 2008; Narayanan & Shmatikov, 2008), keeping the other fields of the data set unchanged. Correlation aware anonymisation of high-dimensional data is an example of synthetic data using anonymisation (Ghinita *et al.*, 2008). The first two kinds of synthetic data type use real-world data either in the input phase of data generation or in the output data set.

3. Hybrid synthetic data

Hybrid synthetic data is generated by using original and synthetic data. Records from the original data set are combined with records from the synthetic data set to create hybrid synthetic data. Hybrid synthetic data have the advantages of both partially synthetic data and fully synthetic data (Surendra & Mohan, 2017).

4. Pure synthetic data

In situations where real-world data is not available, pure synthetic data, also called artificial data or true synthetic data, in which the data will be generated without accessing the real-world data, can be produced. An example of a pure synthetic data generation method is CoMSER (Content Modelling for Synthetic E-Health Records) (McLachlan *et al.*, 2016). This method uses publicly available health information statistics, knowledge collected from experienced clinicians and published clinical practice guidelines. There has been limited research on a method for the automatic generation of data for trust modelling with or without using any real-world data.

The type of synthetic data must be chosen according to the needs or the context (Hawala, 2008). Due to the scarcity of real-world data and since large volumes of data with well-defined characteristics can be generated using a synthetic data generator (Aboulnaga *et al.*, 2001), pure synthetic data will be generated in this study to mitigate the scarcity of the trust data. This generated data can then be used to train the RBFNN model for the calculation of trust values. User-defined rules and constraints will generate synthetic data if the data is generated without original data (Surendra & Mohan, 2017). Next, a generic approach for creating synthetic data will be considered.

4.4 A generic approach to synthetic data generation

McLachlan *et al.* (2018) developed a generic framework for synthetic data generation (SDG), as shown in Figure 4-1:

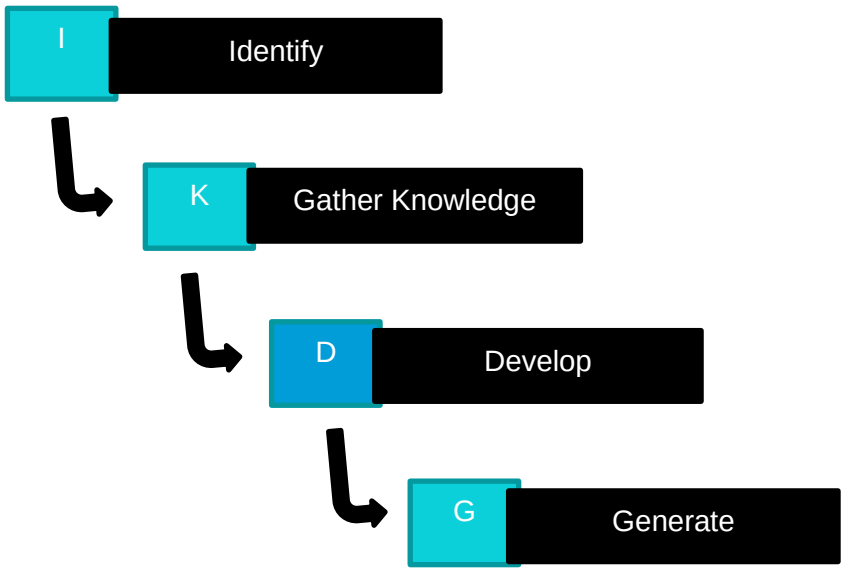


Figure 4-1: Four-step generic SDG approach

The generic framework for SDG follows a pure waterfall method in which the stages never overlap (Alshamrani & Bahattab, 2015). The four basic steps of the generic SDG approach are as follows:

1. Identify the need for synthetic data

In this step, the justification behind synthetic data generation instead of using real-life data is determined. Researchers or practitioners who require highly confidential data can use synthetic data sets (United States General Accounting, 2001). Synthetic data will reduce privacy concerns and will overcome real-life data usage restrictions. One of the benefits of synthetic data is that it can fulfil particular characteristics or specific needs that cannot be done by real-world data sets

(Ayala-Rivera *et al.*, 2016). The lack of enough real-world data can be considered another reason for synthetic data sets.

2. Knowledge gathering

All the knowledge regarding the data required to generate the synthetic data is collected in this step. It can include the characteristics of the required data, the analysis of the data to be generated, the identification of the fields to be generated, and the scope and restrictions or rules that need to be used to create the data set. As synthetic data should represent real-world data, it is necessary to define the properties of the real-world data (Kennedy *et al.*, 2011; McLachlan *et al.*, 2018).

3. Development of the data generation system

In this step, the algorithm or program used for the data generation process (McLachlan *et al.*, 2018) is developed. The data generating algorithm or program must be carefully designed so as not to affect the quality of the generated data. Non-realistic data will result in an invalid trust calculation model.

4. Generate the synthetic data set

The generation of the synthetic data set is the last step. This will be the seed data for the training of the trust model using an RBFNN. The constraints and restrictions or rules identified in Step 2 of the framework will guide the generated synthetic data.

4.5 SDG and the validity of the data set for the trust calculation

Realism is a major concern regarding the quality of a synthetic data set (Tsvetovat & Carley, 2005). Synthetic data must be representative of real-life data sets. Therefore, the quality of the data sets needs to be reasonable. There is a necessity to define the required properties of real-world data. Inaccurate data will lead to the development of an inaccurate RBFNN model to predict trust. The data should be based on standards or guidelines to have some desired property within a specified context to quantify trust. Each electronic entity has its features and standards in a specific context.

One of the key requirements of this study is to generate data using the given standards of an entity and to use this data to train the RBFNN to provide satisfactory results in trust calculations. In the next section, synthetic trust data validation will be explained.

4.6 Synthetic trust data validation

Due to the scarcity of real-world trust data, pure synthetic trust data must be generated to train the RBFNN. As inaccurate data will lead to an inaccurate RBFNN model to predict trust, the validation of the synthetic trust data generation model must be done. This can be done by validating the model output, which is the generated trust data set.

4.6.1 Definition of validation

The focus point of this section is to analyse the definition for the term validation in general, especially in the context of a simulation model and to define the term validation within the context of generated trust data using the Pure Synthetic Trust Data Generation Framework. The Pure Synthetic Trust Data Generation Framework will be explained in Section 4.7.

The following definitions of validation were found in the literature regarding models:

Definition 1. "Validation means that a model is acceptable for its intended use because it meets specified performance requirements" (Rykiel, 1996:229).

Rykiel (1996) defined the term *validation* within the context of testing ecological models. If the model meets the performance requirements, the model is acceptable for its intended use.

Definition 2. "The term validation will be used to refer to various processes and techniques for addressing the comparability between the simulated world of the computational model and the ''real'' world" (Carley, 1996:2).

Carley (1996) defined the term validation within the context of computational modelling as the processes and techniques used for ensuring and assessing the comparability between the synthetic or simulated data and real data. The author describes the real data as the information collected through the experimental, field, archival, or survey analyses.

Definition 3. "Operational validity is defined as determining that the model's output behaviour has sufficient accuracy for its intended purpose or use over the domain of the model's intended application" (Sargent, 1984:115).

The author describes validation as determining whether the model's results are correct for its intended use.

Definition 4. "Verification and validation are processes that collect evidence of a model's correctness or accuracy for a specific scenario" (Thacker *et al.*, 2004:2).

Thacker *et al.* (2004) describe validation as a process of collecting evidence to show the model is sufficiently accurate for its intended use.

Definition 5. "Validation is the process of determining the degree to which a calculation method is an accurate representation of the real world from the perspective of the intended uses of the calculation method" (Jones *et al.*, 2004:1).

Jones *et al.* (2004) describe validation as a process to identify how accurate the calculation method is for its intended use in the real world.

Definition 6. Kleijnen (1998:2) defines "validation as determining whether the conceptual simulation model is an accurate representation of the real system".

The author describes validation as a process to determine whether the simulation model reflects the real system.

Definition 7. "Validation is the process of determining whether a simulation model is an accurate representation of the system, for the particular objectives of the study" (Law, 2019:1402).

Law (2019) describes validation as a process to determine whether the model represents the real world for its intended purpose or the objectives of the study.

Definition 8. "Confirmation by examination and provisions of objective evidence that the particular requirements for a specific intended use are fulfilled" (Society, 1998:71).

Validation provides confirmation and evidence to show validity.

The above definitions lead to the following observations. Some of the above definitions (Carley, 1996; Jones *et al.*, 2004; Kleijnen, 1998) refer to comparing the simulated data or simulated model and the real-world data or real-world system. Most of the authors defined validation as a process to determine the performance of a model so that the user can decide whether to accept or reject the model for its intended purpose (Jones *et al.*, 2004; Sargent, 1984). Validation is defined, based on the needs of a user or a researcher. Validation is a process to determine whether the model represents the real world (Carley, 1996; Jones *et al.*, 2004; Kleijnen, 1998; Law, 2019). Validation provides pieces of evidence to show that the model is sufficiently accurate for its intended use (Society, 1998; Thacker *et al.*, 2004). In this study, the focus is on training a radial basis function neural network using generated trust data sets. The trust data set will be generated using the Pure Synthetic Trust Data Generation Framework proposed in Section 4.7. Since there is a scarcity of real-world trust data, it is not possible to do the

validation by comparing the generated data set with the real-world trust data. Therefore, the focus will be on the intended use of the data set. Hence the validation of the data generation model will be done by determining if the intended use is satisfied.

Based on the discussion in the previous paragraph, validation can be defined in the context of generated trust data as follows:

Definition 9. New Definition: *Validation is a process or method to provide evidence that can be used to determine whether the synthetic trust data generation model is generating a valid trust data set under its specific scenarios for its intended use*.

This new definition will be used in the remainder of the study.

In the next section, methods will be described to determine whether the synthetic trust data generation model generates a valid trust data set when there is a scarcity of real-world trust data to do the validation. What-if analysis and sensitivity analysis can be used to achieve this validation process. Expert knowledge regarding the trust calculations will be used to compile the What-if Scenarios. More details regarding expert knowledge will be given in the next section. Trust data sets will be generated with scatter plots for each scenario. These plotted graphs will then be analysed to determine if the visual representation provides evidence that can be used to determine whether the synthetic trust data generation model is generating a valid trust data set.

4.6.2 Sensitivity and What-if analyses

The comparability between the synthetic or simulated data and real-world data must be done to ensure the validity of the synthetic or simulated data sets if there is enough real-world data. Three situations can arrive regarding the availability of the real-world trust data: no real-world trust data is available, only output trust data (trust values obtained from the real-life trust calculator or system) is available, and both input (input parameter values provided to the real-life trust calculator or system to calculate trust values) and output trust data (trust values obtained from the real-life trust calculator or system) are available. Many types of validation, such as statistical validations, graphical plots, goodness-of-fit tests, structured walk-through of the assumptions document, Schruben-Turing tests, trace-driven simulation, and sensitivity analyses are used in practice, depending upon the availability of real data (Kleijnen, 1998; Law, 2019). Most of the methods mentioned above require real-world data to do the validation.

In this study, the pure synthetic trust data generation model will be validated using Definition 9. As mentioned in Section 4.2, real-world trust data are scarce. The methods discussed above

cannot be used in this study. However, qualitative expert knowledge regarding trust data can be obtained. Expert knowledge indicates how certain input variables affect the output behaviour of the synthetic trust data generation model. Expert knowledge can validate the simulation model by checking whether the input and output behaviour of the simulation model violates the expert knowledge (Kleijnen, 1998). In cases where no real-world data are available, a What-if analysis or a sensitivity analysis can be done to validate the model.

What-if analysis and a sensitivity analysis examine the variation of the output in a model based on the changes in the values of input variables to understand the relationship between the input variables and the output (Chan *et al.*, 2010; Kleijnen, 1997). More specifically, a What-if analysis determines what happens when a parameter or a variable change. On the other hand, a sensitivity analysis establishes what happens when a parameter or a variable change to extreme values. The simulation model analysis requires simulation runs since factors do change from run to run according to the scenarios. Values of one or more input factors will be varied while others will be kept constant.

In this study, both techniques will be used as methods mentioned in Definition 9 to validate the simulation model that produced the trust data set. The analyses can be done in various ways, including mathematical methods, statistical methods, and graphical methods (Christopher Frey & Patil, 2002). The latter will be used to give a visual representation of the change in the generated trust data when the input is changed.

The aim in this study is to develop an alternative way of calculating trust, using an RBFNN that gives satisfactory results. There will be no comparative study with other trust calculating methods in this research. To obtain such a valid RBFNN trust model, the RBFNN must be trained using a valid data set. There is no need for a precise analysis, as this is an alternative model which gives satisfactory results. Graphical methods will be sufficient to determine the validity of the data set.

Scatter plots are simple, effective, and useful to understand the relationship between two variables (Chan *et al.*, 2010; Kleijnen, 1997). A scatter plot will be used with model output on the y-axis and values of one input variable on the x-axis and two different input variables on the x- and y-axes and the output on the z-axis.

The steps below will be followed to do the sensitivity and What-if analyses of the generated trust data set.

1) A set of What-if Scenarios will be compiled, using the available expert knowledge regarding the trust calculations.

2) The trust data set will be generated with scatter plots for each scenario.
3) The plotted graphs will be analysed to determine if the visual representation of the change in the generated trust data matches the expert knowledge.

If the plotted graphs affirm the expectation of the expert, the data generation model can be said to be valid for its intended use. This process can be used as a validation method to show that the data generation model is valid according to Definition 9.

In the next section, a framework for generating a valid pure synthetic trust data set is proposed.

4.7 Framework for generating a valid pure synthetic trust data set

The four steps of the generic framework for SDG (Figure 4-1), which McLachlan *et al.* (2018) developed, will be enhanced to enable the generation of pure synthetic data sets for trust calculation. Three new steps will be included in the generic framework (Figure 4-1) to make it more appropriate for valid data generation in trust calculation. This modified seven-step framework called the PSTDG Framework (Figure 4-2) will be used to generate pure synthetic data sets for trust calculation in this study.

As in the generic framework for SDG (Figure 4-1), the justification behind pure synthetic trust data generation instead of using a real-life trust data set will be determined as a first step in the enhanced framework. All the knowledge, including expert knowledge regarding trust calculation, will be gathered as a second step. The details of the expert knowledge regarding trust calculation are already given in Section 4.6.2.

A set of rules or guidelines can be implemented as constraints in the trust data generation system. Consequently, these rules or guidelines must be compiled before developing the trust data generation model or system. A new step, Compile constraints, will be added to the generic framework for SDG (Figure 4-1) as Step 3 to enable the generation of pure synthetic data sets for trust calculation. In the Compile constraints step, it must be determined how the expert knowledge identified in Step 2 can be used to calculate the trust value. After compiling the constraints, it must be decided on how to calculate trust value or which constraints must be implemented in the trust data generation model, depending upon the needs or situations.

The next step is the Develop step where the development of the trust data generation model or system should happen. As in the generic framework for SDG (Figure 4-1), the algorithm or program that can be used for the trust data generation process must be developed in this step.

The trust data generation model must generate valid trust data. To determine whether the trust data generation model generates a valid trust data set, a step is added to validate the data

generation model between the Develop step and the Feed Input step. The details of how to validate a trust data generation model is given in Section 4.6.2.

The trust data generation model or system must be able to generate data according to specific situations or parameter values. A step called Feed Input where the input values are determined is added between the Validation step and the Generation of the trust data step. In this step, the input parameter values that need to be used for the trust data set generation must be determined.

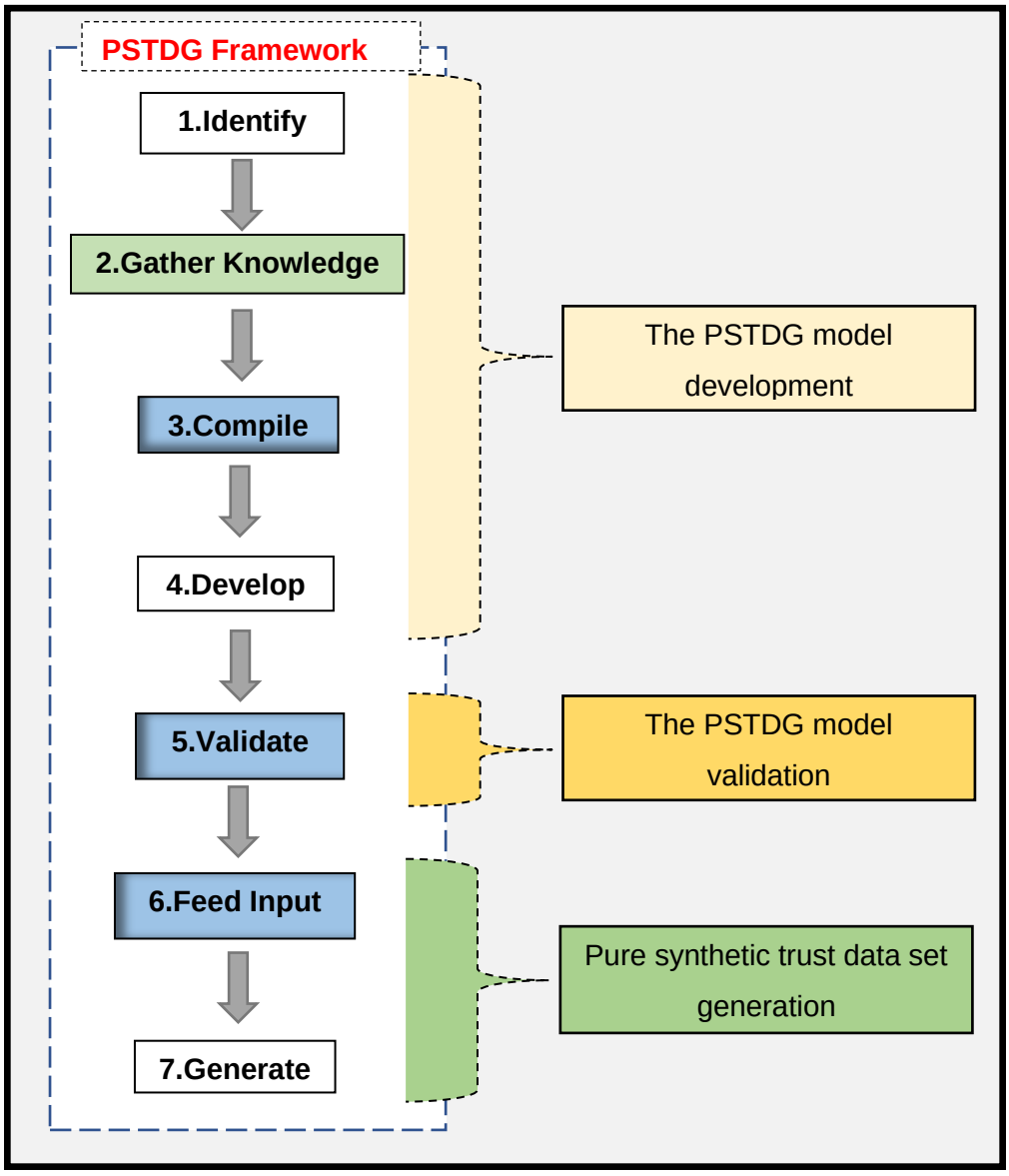


Figure 4-2: PSTDG Framework

The last and final step in the enhanced framework is the Generate step where a trust data set will be generated. The constraints and restrictions or rules selected in Step 3 of the framework and the input parameter values given in Step 6, will guide the generation of the synthetic trust data set. In summary, the enhanced framework obtained three new steps (Compile constraints, Validate the model, and Feed Input) compared to the four-step generic framework for SDG given in Figure 4-1.

The PSTDG Framework is an SDG lifecycle that provides a structured approach to generate valid pure synthetic trust data sets. The PSTDG Framework includes the PSTDG model development, PSTDG model validation, and the pure synthetic trust data generation using the validated PSTDG model.

4.8 Conclusion

In this chapter, problems in trust data collection, a discussion on synthetic data, a generic approach to synthetic data generation, explanations of synthetic data generation and the validity of the data set for the trust calculation were provided. A framework called the PSTDG Framework was proposed for generating a valid pure synthetic trust data set. In Chapter 5, the experimental design to achieve the main aim of this study will be described.

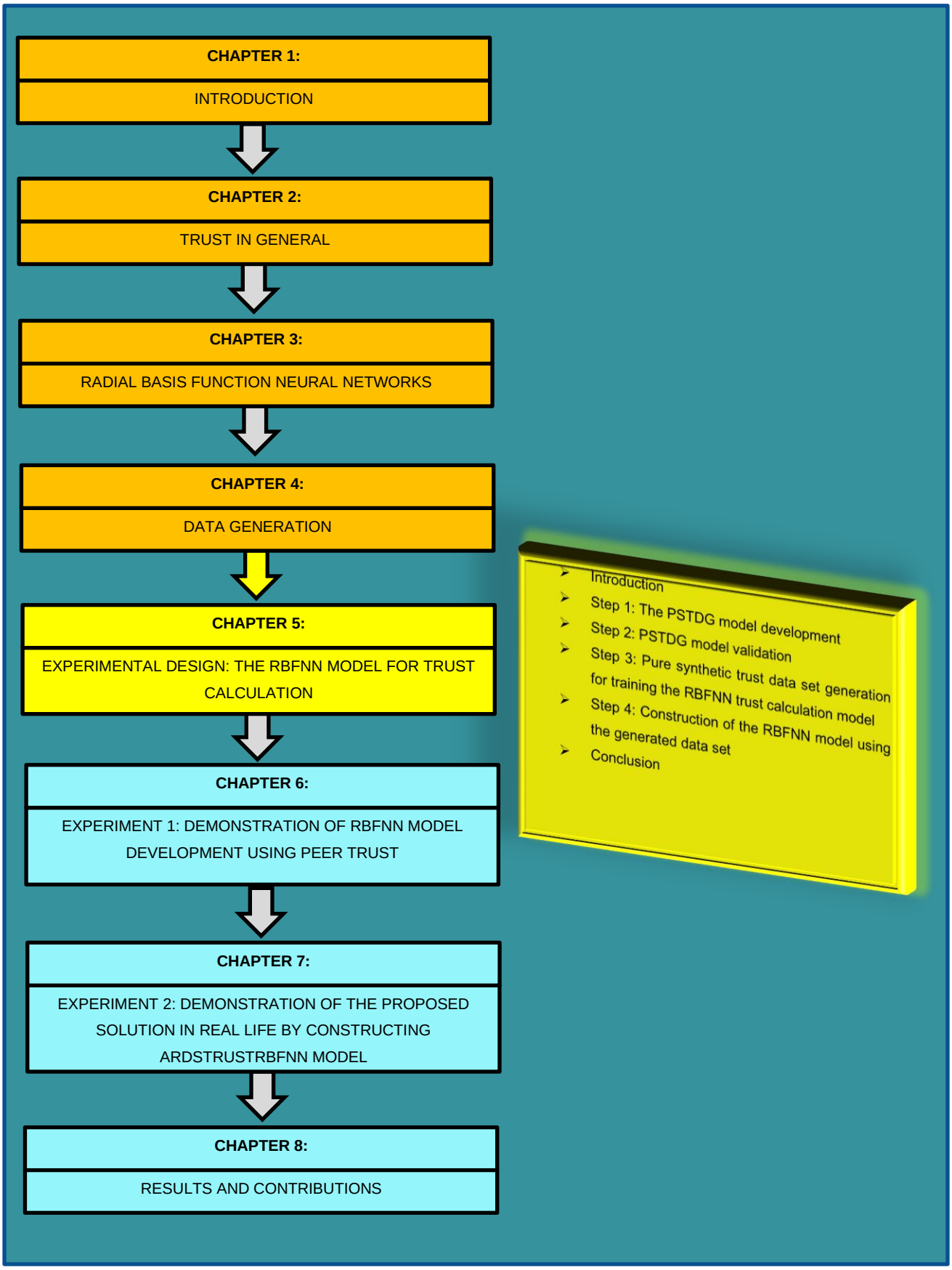


Figure 4-3: Signpost diagram of Chapter 5

CHAPTER 5 EXPERIMENTAL DESIGN: THE RBFNN MODEL FOR TRUST CALCULATION

5.1 Introduction

The main aim of this study is to find an alternative trust calculation method, using radial basis function neural networks (RBFNN). To achieve this goal, a general description is presented in this chapter on how an RBFNN model to calculate trust for a specific problem can be built. To build an RBFNN model that can determine trust values between electronic entities, sufficient instances of trust data sets are required in randomly varying situations. To obtain a valid RBFNN trust model, the RBFNN must be trained, using a valid data set. Inaccurate data will lead to an inaccurate RBFNN model to predict trust.

In Section 4.2, it was identified that building a trust model, using an RBFNN that can determine trust values between electronic entities, requires large trust data sets. It was also identified that there is a scarcity of a real-world trust data set, and the trust data set needs to be generated synthetically in this study. Training an RBFNN model using an inaccurate trust data set can lead to the development of an inaccurate RBFNN trust model. The Pure Synthetic Trust Data Generation (PSTDG) framework developed in Section 4.7 provides a structured approach to generate valid pure synthetic trust data sets. The PSTDG Framework developed in Section 4.7 has seven steps to generate a synthetic trust data set.

To build a model using an RBFNN that can determine trust values between electronic entities, a four-step experimental design process will be followed as shown in Figure 5-1. Figure 5-1 also shows how the PSTDG Framework developed in Section 4.7 is linked to the four-step experimental design process to build a model using an RBFNN that can determine trust values between electronic entities. The four steps in the experimental design process are given below:

Step 1: A PSTDG model will be developed, using the first four steps of the PSTDG Framework discussed in Section 4.7. This will be explained in Section 5.2.
Step 2: The PSTDG model will be validated, using Definition 9 described in Section 4.6.1. This will make use of the fifth process of the PSTDG Framework discussed in Section 4.7 and will be detailed in Section 5.3.
Step 3: In Section 5.4, how a pure synthetic trust data set can be generated using the validated PSTDG model will be described. Synthetic trust data set generation will be done using the sixth and seventh process of the PSTDG Framework discussed in Section 4.7.
Step 4: The construction of the proposed RBFNN model for trust calculation, using the generated data set will be explained in Section 5.5.

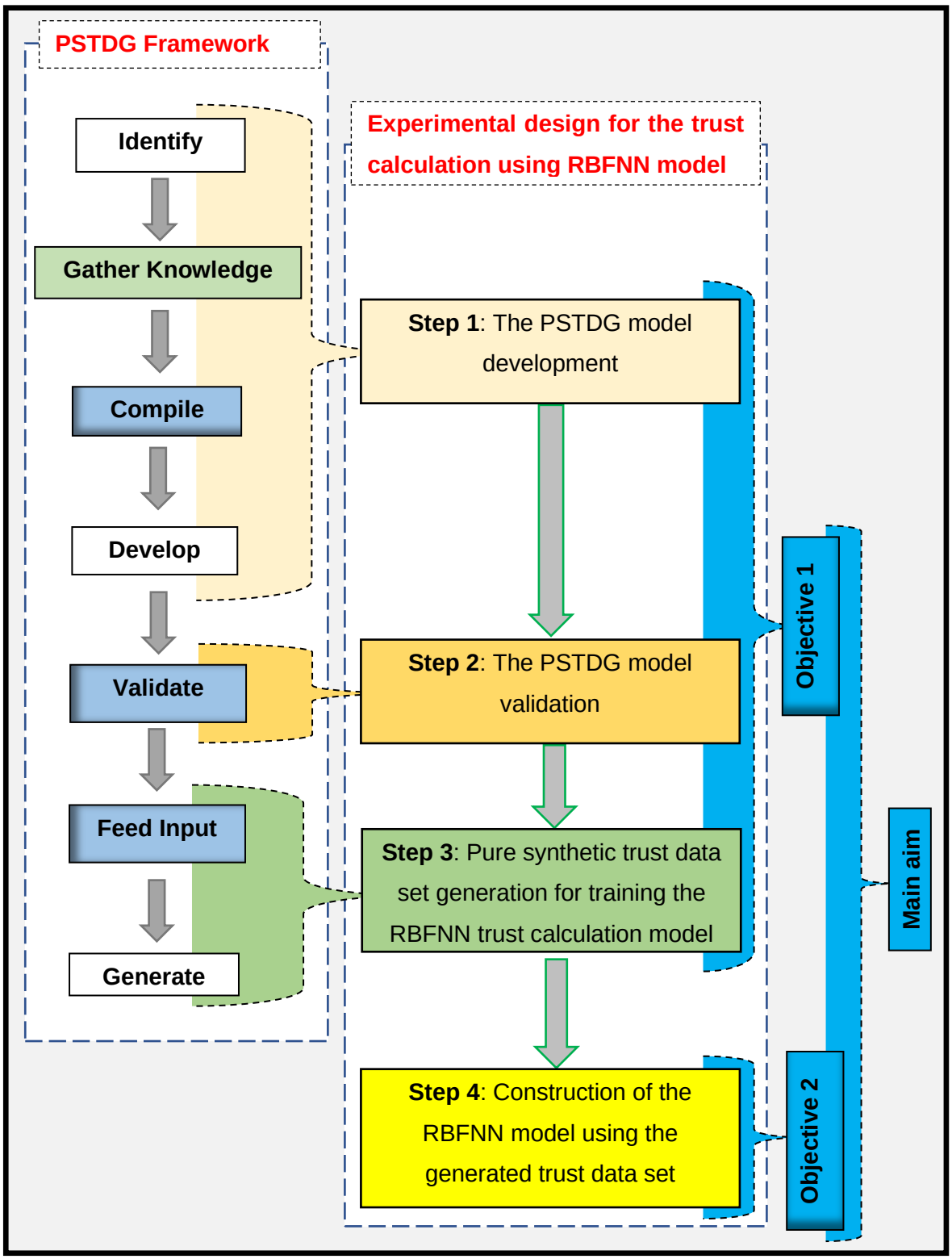


Figure 5-1: Experimental design of the RBFNN model for trust calculation

In this chapter, the experimental design to build a model using an RBFNN that can determine trust values between electronic entities will be provided. The remainder of this chapter is organised as follows. In Section 5.2, the implementation issues regarding the first four steps of the PSTDG Framework will be explained in connection with the development of the PSTDG model which is the first step to achieve research **Objective 1**. The implementation details of the

PSTDG model validation will be detailed in Section 5.3. In Section 5.4, the trust data set generation, which is the last step to achieve research **Objective 1**, will be addressed. This will solve the problem of not having a large, validated trust data set to build a model using an RBFNN that can determine trust values between electronic entities. In Section 5.5, the final step of the four-step experimental design process to build a model using an RBFNN that can determine trust values between electronic entities will be implemented. This will explain how an RBFNN model for trust calculation can be constructed using the generated trust data set and this will lead to the achievement of research **Objective 2**. The chapter will be concluded in Section 5.6.

In this chapter, the details of the four-step experimental design process to achieve the main aim of this study, namely to develop an alternative trust calculation method, using an RBFNN, as discussed in Section 1.3 will be provided.

5.2 Step 1: The PSTDG model development

The first step in the four-step experimental design process given in Figure 5-1 is the development of the PSTDG model. The development of the PSTDG model will make use of the first four parts of the PSTDG Framework which is described in Chapter 4 as shown in Figure 5-1.

5.2.1 Identify the need for SDG

Training an RBFNN trust model requires a large trust data set and in Section 4.2, the need for trust data set generation was identified, where data generation is research **Objective 1**.

5.2.2 Gather knowledge

All knowledge, including qualitative expert knowledge regarding trust data, must be obtained in this process. As described in Section 4.6.2, the expert knowledge must include the characteristics of the required data, the fields to be generated, and the scope and the restrictions or the rules which must be used to generate the synthetic trust data so that the synthetic trust data set can represent the real-world trust data set.

In this study, *"Trust is a quantified belief or a probability of belief of an entity, which can be calculated by accessing or using information from different sources which are based on some set of standards or guidelines, to have some desired property within a specified context"* as defined in Section 2.2.

This definition states that trust can be calculated using information from sources, based on some standards or guidelines, resulting in the desired properties for the trust values. The properties of trust values will depend on the data, or alternatively the standards or guidelines. However, the trust data is not available as given in Section 4.2. Therefore, the problem needs to be analysed and must determine the following:

- All the input sources, features offered by an entity and the critical factors that can be used for evaluating the trust value
- Any standards that can be used to evaluate trust value
- The relationship between critical factors or input sources and the output trust value

The input sources can be obtained from a variety of sources, such as recommendations, previous experience, and feedback. All the critical factors for evaluating the trust value must be listed by doing research on an electronic entity as mentioned above. The feedback a peer obtains from another peer in terms of the level of satisfaction, the feedback scope in terms of the number of transactions and percentage availability or percentage success of Automated Backups are examples of critical factors. These critical factors are entirely dependent on the entity and context. Features offered by the entities must be identified. Any available offered standards for the entity that can be used to evaluate the trust value must be identified. For instance, the monthly uptime percentage (MUP) availability of the offered features can be considered as an example of standards offered by an entity (see Chapter 7 for a discussion of such an example). One would need to identify how certain input variables (critical factors) must affect the trust value with respect to the offered standards. From all the obtained information, expert knowledge must be listed according to each critical factor that affects the trust value. A suitable choice must be made by the user to decide on how the trust value must be linked to the critical factors of an entity because different choices of the critical factors will deliver different trust values. The following can be considered as examples for expert knowledge (see Section 6.2.2 and 7.2.2):

“The new trust value of the interacting peer must depend upon the satisfaction received from other peers after an interaction. Thus, the new trust value should be directly proportional to the satisfaction received”.

“The new trust value of the ARDS in the context of availability and durability must depend upon the success of MUP. Thus, the new trust value should be directly proportional to the MUP. The trust value must decrease when the percentage availability decreases.”

All the knowledge, including expert knowledge regarding the data required to generate the synthetic trust data, will be collected in this process. After obtaining the expert knowledge, the next task is compiling constraints using the expert knowledge which will be detailed in the next section.

5.2.3 Compile constraints

As shown in Figure 5-1, compiling the constraints is the third task that needs to be performed in the development of the PSTDG model. All the expert knowledge will be identified and listed in the previous task. To develop a PSTDG model, some calculations, algorithm or equations must be developed to calculate trust values using this expert knowledge. A trust value can be calculated differently in different contexts and the user can decide how to calculate a trust value, depending upon the needs, situations or contexts. It must be identified which expert knowledge is applicable to calculate the trust value, depending upon the needs. For example, the trust value can be calculated with or without using transaction size and more weight can be assigned to the feedback received for larger transactions (see Section 6.2.3 for a discussion of such an example). Expert knowledge linked to the transaction size must also be considered and included when developing the equation to calculate the trust value if the transaction size must also be considered when calculating trust. These types of decision on what to include and what to exclude when calculating the trust value can be considered as rules or guidelines to calculate trust. The developed PSTDG system or PSTDG model must generate a trust data set that satisfies the expectation of the expert knowledge. The validity of the developed PSTDG system or PSTDG model will be carried out in Step 2. If the developed PSTDG model is not valid, one must return to this task and revise the calculations, algorithm or equations until a valid PSTDG model is obtained.

A set of rules or guidelines to calculate trust values, using the expert knowledge identified in the previous task, must be listed as constraints for the PSTDG system or PSTDG model as indicated in Section 4.7. Trust calculating equations in connection with each constraint must be developed using the expert knowledge. After compiling the constraints, the user must decide how to calculate the trust value or which constraint or equation must be implemented in the trust data generation system or a PSTDG model, depending upon the needs or situations. After selecting a particular constraint or equation, the next task is the development of the PSTDG model using the selected constraints. This task will be explained in the next section.

5.2.4 Develop PSTDG model

After the selection of a particular constraint, calculations or an equation that needs to be implemented in the data generation system or a PSTDG model, the input values and the output

values must be decided. *PeerCount* (the total number of peers in the network) and *InteractionCount* (the total number of transactions that need to be done) are examples of input values. The trust value of the interacting peer can be an example of output values. Thereafter, a computer program will be designed to implement the selected constraint, calculations or equation that was selected in the previous task into the trust data generation system or a PSTDG model, using a data flow diagram. After finalising the design (data flow diagram), an implementation must be done using any programming language. MS SQL code will be used in this study to develop a trust data generation system or a PSTDG model, as it allows the creation, storage, retrieval and manipulation of large amounts of data (Ben-Gan, 2012).

After completing Step 1 of the four-step experimental design process shown in Figure 5-1, the problem of not having a large trust data set for the training of the RBFNN model, which was identified in Section 4.2, will obtain a solution, as the developed PSTDG model can be used to generate a large trust data set. However, the quality of the generated trust data set needs to be reasonable as described in Section 4.5. The PSTDG model validation will be the next step in the experimental design of the RBFNN model for trust calculation as shown in Figure 5-1 which will be explained in the next section.

5.3 Step 2: PSTDG model validation

In Step 1, a PSTDG model was developed for trust data set generation. The validity of this data generation model must now be tested. As mentioned in Section 5.2.2 in this study, *"Trust is a quantified belief or a probability of belief of an entity, which can be calculated by accessing or using information from different sources which are based on some set of standards or guidelines, to have some desired property within a specified context".* This definition states that trust values will have some properties, as it is calculated using information from sources, based on some standards or guidelines.

In Chapter 4, Definition 9 defines validation as a process or method to provide evidence that can be used to determine whether the synthetic trust data generation model is generating a valid trust data set under its specific scenarios for its intended use. A data set is said to be valid if it satisfies the expectation of the expert knowledge for its intended use. A valid PSTDG model should be able to generate a trust data set that can satisfy the expert knowledge identified and implemented as constraints during PSTDG model development. The data set's quality needs to be validated, using Definition 9 described in Section 4.6.1 to develop an accurate RBFNN model to predict trust. The validation of the developed PSTDG model will be done by completing the three steps for validation as described in Section 4.6.2.

5.3.1 Compile

The first step in the validation process is the compilation of What-if Scenarios regarding the calculation of trust values as explained in Section 4.6.2. The What-if Scenarios will be compiled, using the expert knowledge identified during Step 1 and which is used for the constraints that are implemented in the PSTDG model. As explained in Section 5.2.2, expert knowledge is obtained after identifying how a particular input variable (critical factor) should affect the trust value. Expert knowledge will describe how a particular input variable will affect the trust value. It must be determined what will happen to the trust value when a particular input variable or critical factor changes, keeping all other input variables or critical factors constant. The following can be considered as an example of a What-if Scenario for the expert knowledge obtained in Section 5.2.2:

- "What is the effect on the trust value of Peer1 if the satisfaction level received from all other peers interacting with Peer1 is randomly decreased? The satisfaction level starts at a maximum value of 1 and declines with a random value that lies between 0 and 0.001 in a series of transactions".
- "What is the effect on the trust value of the ARDS if the satisfaction (feedback regarding the percentage availability of features) decreases? The satisfaction for percentage availability of features starts at a maximum value of 1 and declines with a random value lying between 0 and 0.001 in a series of feedback submissions".

All the What-if Scenarios that can be compiled using the expert knowledge which was implemented in the PSTDG model as constraints in Step 1 must be listed as a first task. After listing all the What-if Scenarios, a set of experiments must be compiled to test whether the expert knowledge is satisfied. The next task in the validation process is the generation of the data set and the scatter plots, as described in Section 4.6.2. This will be explained in the next section.

5.3.2 Generate and plot

Trust data with scatter plots will be generated, using the developed PSTDG model according to the What-if Scenarios compiled in the previous step. The trust data set will be generated by keeping the input variable or critical factor which is linked to the expert knowledge variable and keeping all other input variables or critical factors fixed with a constant value. After generating the scatter plots, they must be analysed in the third validation step which will be explained below.

5.3.3 Analysis

The PSTDG model developed in Section 5.2.4 by implementing the selected constraint from Section 5.2.3 uses particular expert knowledge identified in Section 5.2.2. This expert knowledge will be describing a particular input variable (critical factor) that affects the trust value. If the visual representation of the plotted graphs matches the expert knowledge in the constraint which is implemented in the PSTDG model, the PSTDG model can be said to be valid. According to Section 4.6.2, if the plotted graphs affirm the expectation of the identified expert knowledge, the data generation model can be said to be valid for its intended use. In this step, it will simply answer whether the graphical representation of the generated data set corresponding to each What-if Scenario confirms the corresponding expert knowledge. If the PSTDG model is said to be not valid, one must go back to the third task (Section 5.2.3) of Step 1 of the four-step experimental design process until the model is valid.

The problem of not having a validated trust data set can be solved using Step 2 in the four-step experimental design process shown in Figure 5-1. A pure synthetic trust data set can be generated, using the validated PSTDG model as the next step of the four-step experimental design process.

5.4 Step 3: Pure synthetic trust data set generation for training the RBFNN trust calculation model

The PSTDG model, which is developed as explained in Section 5.2, will be validated as explained in Section 5.3. In this step, the pure synthetic trust data set will be generated, using this validated PSTDG model. The pure synthetic trust data set will be generated, using the last two parts of the PSTDG Framework as shown in Figure 5-1. The following are the two tasks that need to be carried out according to Figure 5-1: feed input and generate the trust data set. In Figure 5-1, the Feed Input task in the PSTDG Framework just above the Generate task is to decide the input parameters' value that needs to be given as input to the developed PSTDG model. The developed PSTDG model or trust data generation model will be able to generate data according to specific situations or input parameter values. The input parameter values will be decided for the PSTDG model. *PeerCount* (the total number of peers in the network), the *InteractionCount* (the total number of transactions that need to be done), and *MonthCount* (the total number of months taking part in the trust calculation) are examples of input parameters or user constraints (See Chapters 6 and 7). Before developing the data set, a set of input parameter values that needs to be used for the trust data set generation should be selected, as described in Section 4.7. Then the validated PSTDG model will generate the pure synthetic trust data set using the given parameter values.

The problem of not having validated a large data set will be solved using Step 3 of the four-step experimental design process shown in Figure 5-1. This will achieve research **Objective 1**. Construction of the RBFNN model, using the generated trust data set, will be considered in the next section.

5.5 Step 4: Construction of the RBFNN model using the generated data set

This study aims to develop an alternative trust calculation method using an RBFNN, as discussed in Section 1.3. To build an accurate RBFNN model that can determine trust values between electronic entities, suitable model hyperparameters must be chosen before training the model. This can be done by training several candidate RBFNN models using uniform randomly sampled hyperparameters and selecting the best model. Random search is a widely used and efficient method for hyperparameter optimisation (Bergstra & Bengio, 2012; Li *et al.*, 2021; Wu *et al.*, 2019).

The proposed RBFNN model for trust calculation will be built on the generated trust data set. This process will include the following three processes: data pre-processing, identifying the best RBFNN model, and evaluating the best model.

5.5.1 Data pre-processing

Data pre-processing can improve the accuracy of the ANN model, decrease the computational cost, and accelerate the learning process (Koval, 2018; Kuźniar & Zając, 2017; Mohd Nawi *et al.*, 2013). In addition, the pre-processing of the input variables helps to better match the predicted output (Koval, 2018; Kuźniar & Zając, 2017). The data set obtained by Step 3 of the four-step experimental design process given in Figure 5-1 needs to be pre-processed before constructing the RBFNN model if the generated inputs vary across different ranges (Koval, 2018). After completing the data pre-processing, the best model hyperparameters need to be identified, as explained in the next section.

5.5.2 Identifying the best RBFNN model for trust calculation

In this process, experiments will be done to identify the best model hyperparameters for the RBFNN model. Before the search for the best hyperparameters is performed, a hyperparameter search space needs to be defined. The program utilised for this purpose will be written in the Python programming language with the TensorFlow library and Keras application program interface (API) used for RBFNN training. TensorFlow is Google's open-source deep learning software library, based on computational graphs for defining, training, and deploying machine learning models (Abadi *et al.*, 2016; Shukla & Fricklas, 2018). Keras which is written in Python,

serves as a high-level API for the TensorFlow library (Chollet, 2020; Nagisetty & Gupta, 2019; Shanmugamani, 2018).

5.5.3 Evaluation of the best RBFNN model for trust calculation

As a final task, the evaluation of the RBFNN model will be done. An MSE loss value closer to zero shows better model performance (Elzwayie *et al.*, 2017).

The experimental design process to build a trust model using an RBFNN given in Figure 5-1 has four steps. A PSTDG model will be developed in the first step, the developed PSTDG model will be validated in the second step and a trust data set will be generated using the validated PSTDG model in the third step. After the successful completion of the first three steps of the four-step experimental design process, research **Objective 1** will be achieved. This will solve the problem of not having a large valid trust data set to build a model using an RBFNN that can determine trust values between electronic entities. An RBFNN model will be developed for trust calculation in the fourth step using the generated trust data set. Thus research **Objective 2** will be achieved after the completion of the fourth step of the experimental design process given in Figure 5-1. The four-step experimental design process given in Figure 5-1 leads to achieving the main aim of this study, namely to develop an alternative trust calculation method using an RBFNN.

5.6 Conclusion

In this chapter, a four-step experimental design process to achieve the main aim of this study to build a model using an RBFNN that can determine trust values between electronic entities was provided. In Chapter 6, the first experiment regarding the PSTDG model development, the validation of the PSTDG model, the pure synthetic trust data set generation for training the RBFNN trust calculation model, and the construction of the RBFNN model called PeerTrustRBFNN, using a purely theoretical trust calculation model (the PeerTrust model by Li and Ling (2004)) will be discussed.

The second experimentation to show that the same four-step experimental design process can be used to generate a trust data set and to build a trust model for a real-world problem will be discussed in Chapter 7. To achieve this aim, the construction of the RBFNN model, called ARDSTrustRBFNN, will be created for the Amazon Relational Database Service (ARDS) which was studied in Section 2.6.

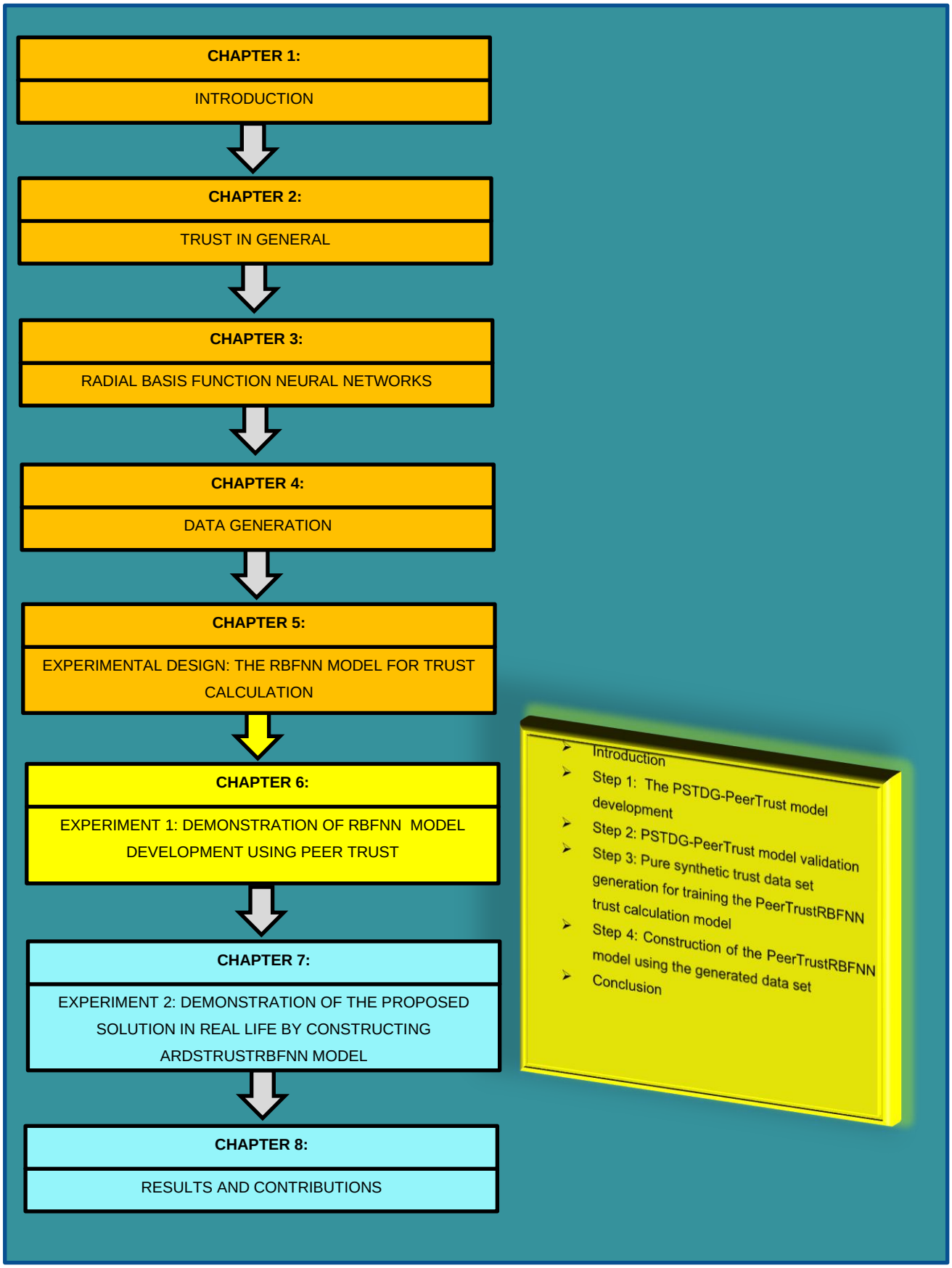


Figure 5-2: Signpost diagram of Chapter 6

CHAPTER 6 EXPERIMENT 1: DEMONSTRATION OF RBFNN MODEL DEVELOPMENT USING PEERTRUST

6.1 Introduction

In order to achieve the main aim of this study to find an alternative trust calculation method using radial basis function neural networks (RBFNN), a four-step experimental design process was described in Chapter 5. There, it was described in general how an RBFNN model to calculate trust for a specific problem can be built. Figure 5-1 shows how the PSTDG Framework developed in Section 4.7 is linked to the four-step experimental design process to build a model using an RBFNN that can determine trust values between electronic entities. Using the four-step experimental design process given in Chapter 5, two experiments will be done in the following chapters.

The PeerTrust model by Li and Ling (2004) which was studied in Section 2.4.6 is purely a theoretical trust calculation model. In this chapter, an RBFNN trust model called the PeerTrustRBFNN model using the PeerTrust model by Li and Ling (2004) will be developed using the four-step experimental design process described in Chapter 5. As a first step, a pure synthetic trust data generation (PSTDG) model called the PSTDG-PeerTrust model will be developed, it will be validated, a large trust data set will be generated and the RBFNN trust model called the PeerTrustRBFNN will be trained using this data set. In this experiment, the focus during the development of the RBFNN trust model called the PeerTrustRBFNN will be mainly on the validation of the trust data generation model called the PSTDG-PeerTrust model. This is because the PeerTrust model by Li and Ling (2004) is purely a theoretical trust calculation model and Li and Ling provided the trust calculation equation. Even though the trust calculation equation is available, it is not necessary that the data generation model that will be created using that equation gives a valid trust data set, as errors can happen in the program design, data management, and trust calculation. The focus will therefore be mainly on Step 2 of the validation step of the four-step experimental design process described in Figure 5-1.

To build an RBFNN trust model called the PeerTrustRBFNN model using the PeerTrust model by Li and Ling (2004), the following steps will be followed, as stated in Figure 6-1.

Step 1: A PSTDG model called the PSTDG-PeerTrust model will be developed, using the first four steps of the PSTDG Framework discussed in Section 4.7. This will be done in Section 6.2.
Step 2: The PSTDG-PeerTrust model will be validated in Section 6.3, using Definition 9 described in Section 4.6.1.

Step 3: In Section 6.4, it will be described how a pure synthetic trust data set can be generated, using the validated PSTDG-PeerTrust model.

Step 4: The proposed RBFNN model called the PeerTrustRBFNN for trust calculation will be developed, using the generated data set in Section 6.5.

The chapter will be concluded in Section 6.6.

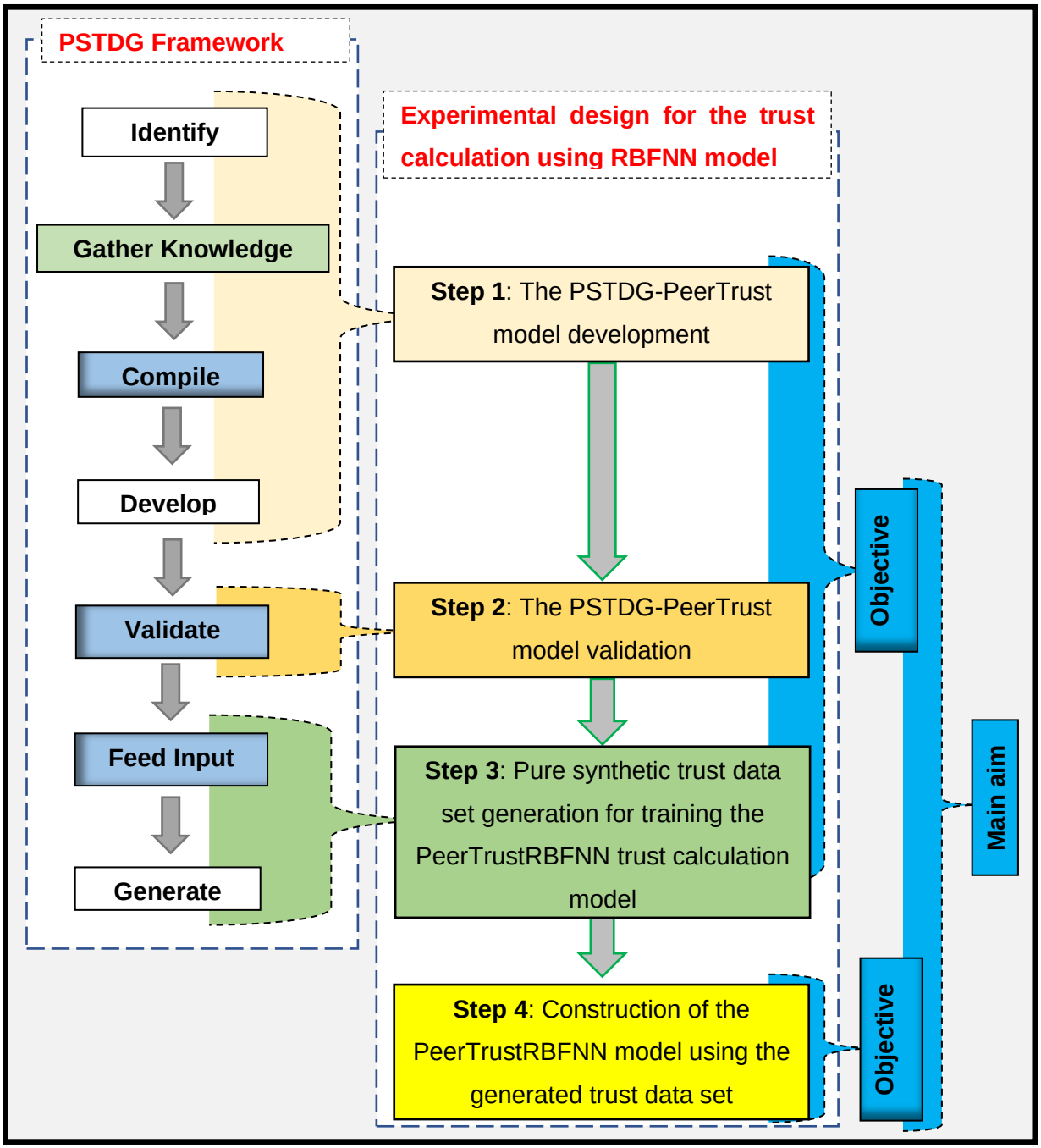


Figure 6-1: Experimental design for the construction of PeerTrustRBFNN

6.2 Step 1: The PSTDG-PeerTrust model development

As explained in Section 5.2, the first four parts of the PSTDG Framework which is given in Section 4.7 will be used to develop the data generation model called the PSTDG-PeerTrust.

6.2.1 Identify the need for synthetic data generation (SDG)

The need for SDG is explained in Section 5.2.1.

6.2.2 Gather expert knowledge

In this process, all knowledge, including qualitative expert knowledge regarding trust data, will be obtained, as described in Section 5.2.2. According to Section 2.4.6, Li and Ling (2004) identified the following five critical factors for evaluating the trust of a peer in P2P electronic communities:

1. The feedback a peer obtains from another peer in terms of the level of satisfaction
2. The feedback scope in terms of the number of transactions
3. The credibility of the feedback source
4. The transaction context factor
5. The community context factor

The following details of the interacting peer and the other peer must be specified, as shown in

Table 6-1.

Variable Name	Description
Interacting peer	Name of the interacting peer (for example Peer1)
Other peer	Name of the peer engaging with the interacting peer (for example Peer2)
Transaction size	The size of the transaction can be small, medium, large, extra large, etc.
Transaction size value	A numeric value for the specific transaction

Table 6-1: Variable list

Variable Name	Description
Feedback given by the interacting peer	Whether the interacting peer gives feedback to the other peer for a transaction(yes/no)
Feedback given by the other peer	Whether the other peer gives feedback to the interacting peer for a transaction
Level of satisfaction received from the other peer	Level (a numerical value) of satisfaction given to the interacting peer by the other peer for a transaction
Level of satisfaction given by the interacting peer	Level (a numerical value) of satisfaction given to the other peer by the interacting peer for a transaction
Transaction context factor	Whether the trust calculation included the transaction context factor
Community context factor	Whether the trust calculation included the community context factor
Current trust value of the interacting peer	Current trust value of the interacting peer
Current trust value of the other peer	Current trust value of the other peer
New trust value of the interacting peer	New trust value of the interacting peer after the transaction
New trust value of the other peer	New trust value of the other peer after the transaction

Table 6-1: Variable list (Continued)

Variable Name	Description
Interaction status	Status (occurred or not) of the current interaction will be recorded
Total interactions done by the interacting peer	Total interactions did so far by the interacting peer
Total interactions done by the other peer	Total interactions by the other peer done so far

Table 6-1: Variable list (Continued)

The importance of the five critical factors is explained by Li and Ling (2004) which is explained in Section 2.4.6. The parameters (critical factors) 1, 2 and 3 are important in trust value calculations in any P2P community. The weight factors α and β can have a default value or can be set explicitly to assign different weights to feedback-based evaluation and community context according to the situations.

The following information, distilled from the explanation given in Section 2.4.6, can be considered as the expert knowledge regarding the peer's trust value calculation.

Expert knowledge 1. The new trust value of the interacting peer must depend upon the satisfaction received from other peers after an interaction. The new trust value should be directly proportional to the satisfaction received.
Expert knowledge 2. The size of the transaction could play a vital role in trust value calculation. The new trust value of the interacting peers must depend upon the transaction size. The trust value should be directly proportional to the transaction size.
Expert knowledge 3. Feedback from the more trusted peer should have more weight.
Expert knowledge 4. The trust value should be calculated using all the feedback received during all previous transactions.
Expert knowledge 5. The trust value can depend upon the community context. Peers who submit feedback can be rewarded through the community context factor.

The above-identified expert knowledge will be used for compiling the constraints for the PSTDG-PeerTrust model development which will be explained in Section 6.2.3 and Section 6.2.4 and its validation in Section 6.3.

6.2.3 Compile constraints

According to Li and Ling (2004), a peer's trust value can be calculated differently in different contexts. A user can decide how to calculate trust depending upon the needs or situations. Any one of the following three cases can be implemented while developing the model to generate trust data sets:

6.2.3.1 Case 1: Basic Trust Matrix

According to Li and Ling (2004), the trust value for peer *a* is obtained by Equation (**2-12**). By turning off the transaction context factor ($TF(a,i) = 1$) and community context ($\alpha = 1 \text{ and } \beta = 0$) in Equation (2-12), the trust value for peer *a* is obtained by Equation (2-13) which is known as the basic trust matrix. The trust value for peer *a* will be the weighted average of each transaction's satisfaction received by peer *a.* As explained in Section 2.4.6, $Cr\big(p(a,i)\big)$ can be calculated in different ways. Credibility will be calculated as a function of trust in this study. Consequently, the trust value for peer *a* is obtained by Equation (2-14). In this case, the trust value is calculated using Expert knowledge 1, Expert knowledge 3 and Expert knowledge 4.

6.2.3.2 Case 2: Basic Trust Matrix with transaction context factor

In Case 2, the transaction context factor was incorporated into the basic trust matrix. In this case the trust value for peer *a* is obtained by Equation (2-17). As indicated in Section 2.4.6, the size, category or timestamp of the transaction can be applied as a transaction context. Compared to other transactions, more weight can be assigned for the feedback received for larger, more important, or more recent transactions. In this study, the transaction context factor will be calculated by using the size of the transaction. In addition to Expert knowledge 1, Expert knowledge 3 and Expert knowledge 4, Expert knowledge 2 is also incorporated in this case.

6.2.3.3 Case 3: Basic Trust Matrix with transaction context factor and community context factor

In Case 3, both the transaction context and community context factors were incorporated into the basic trust matrix. In this case, the trust value for peer *a* is obtained by Equation (2-19). As described in Section 2.4.6, to incorporate the transaction context factor and the community context factor into the basic trust matrix, the weight factors $\alpha \text{ and } \beta$ must be tuned in such a way as to control the reputation level that can be gained by giving feedback to other peers. A suitable choice for the weight factors $\alpha \text{ and } \beta$ must be made by the user to decide how much the community context factor must contribute to trust value. In this study, the weight factors $\alpha \text{ and } \beta$

will be set to the same value, namely 0.5 in Case 3 so that all factors can contribute equally to the trust value.

It is the user's choice to decide which of the above three cases must be used to calculate trust. In this study, the Basic Trust Matrix equation with the transaction context factor and the community context factor (Case 3) will be used to calculate trust. This is because Case 3 considers all of the five critical factors for evaluating a peer's trust value. In this case, the trust value is calculated using Expert knowledge 1, Expert knowledge 2, … Expert knowledge 5.

The PSTDG-PeerTrust model development will be explained in the next section.

6.2.4 The PSTDG-PeerTrust model development

The PSTDG-PeerTrust model can be developed by applying any one of the cases described in Section 6.2.3 to generate pure synthetic data sets. This can be achieved by changing the weight factors α and β and the transaction context factor. By assigning 0.5 to the weight factors α and β, equal importance can be given to all the five critical factors mentioned in Section 6.2.2 in the trust calculation. Since the trust value is calculated using Expert knowledge 1, Expert knowledge 2, Expert knowledge 3, Expert knowledge 4 and Expert knowledge 5 in Case 3, Case 3 will be implemented in this study by assigning 0.5 to the weight factors α and β. As a result, the transaction context and the community context will have equal importance in the trust calculation. The experiment will be performed by two stored procedures as follows:

MS SQL code is used to write the two stored procedures. Table 6-2 shows the inputs and outputs of the first procedure.

Inputs	Outputs
• *PeerCount* (the total number of peers in the network) • *InteractionCount* (the total number of transactions that need to be done)	• Given number of peers will be created with names. • Assign initial trust values to each peer. • Fix interaction details that should happen.

Table 6-2: First stored procedure

For each transaction, the transaction context (the value of the transaction and the functionality of the transaction) can be different as described in Section 2.4.6 and Section 6.2.3.2. In this

study, the transactions will be classified as Extra Large, Large, Medium, and Small. A uniform random value inside the given ranges will be assigned to each transaction according to the size, as shown in Table 6-3.

Transaction size	Transaction size value
Extra Large	0.75 < Value ≤ 1
Large	0.5 < Value ≤ 0.75
Medium	0.25 < Value ≤ 0.5
Small	0 ≤ Value ≤ 0.25

Table 6-3: Transaction size

The transaction size and its corresponding value will be stored in the SQL database. The details of the first stored procedure are given in Figure 6-2.

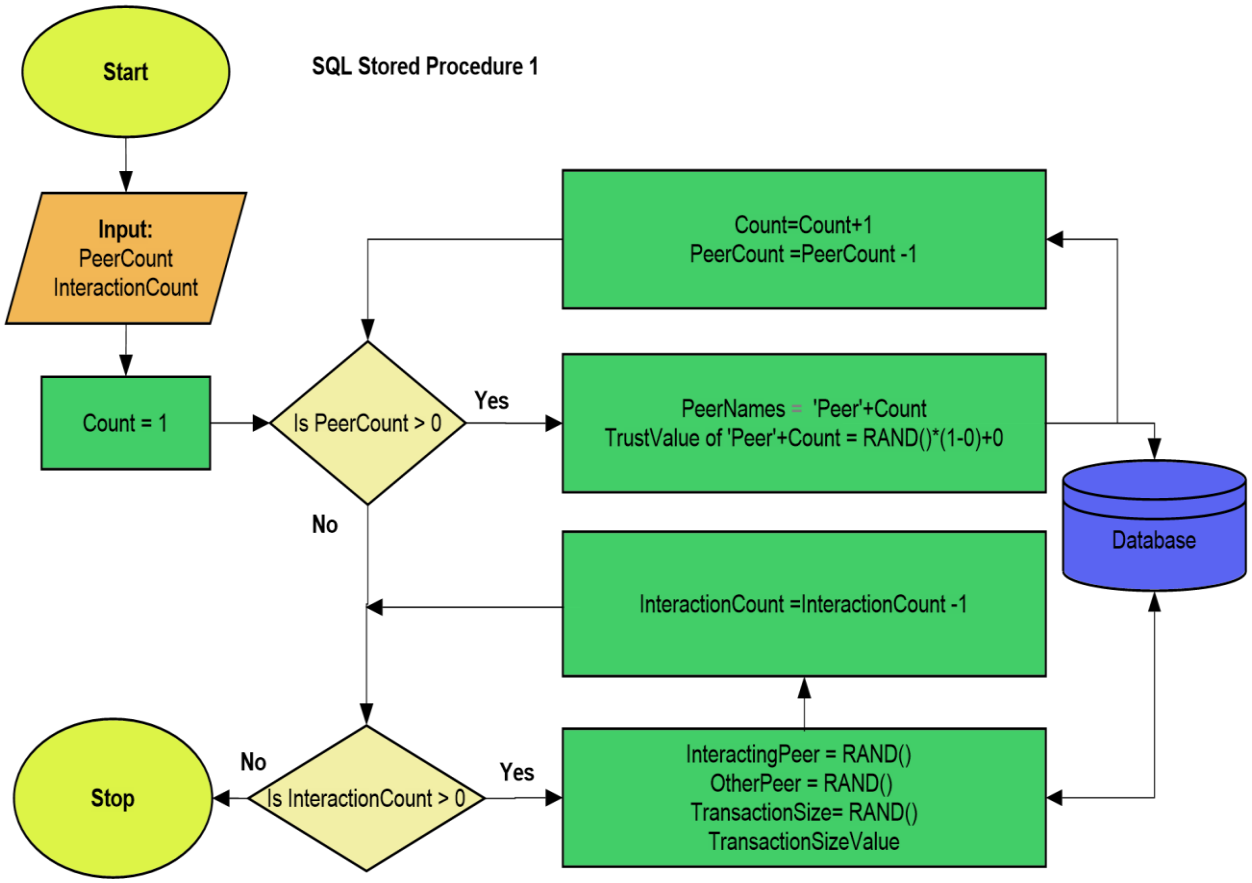


Figure 6-2: Stored Procedure 1 data flow

Table 6-4 shows the inputs and outputs of the second stored procedure.

Inputs	Outputs
• *PeerCount* (the total number of peers in the network) • *InteractionCount* (the total number of transactions that need to be done) • *TransactionContextFlag* (binary flag indicating whether the trust calculation must consider the transaction context factor) • *CommunityContextFlag* (binary flag indicating whether the trust calculation must consider the transaction community context factor)	• The given number of transactions will be done. • Each peer's trust value will be updated according to the feedback received, the feedback's credibility, the total number of transactions done by the peer, the transaction context factor, and the community context factor.

Table 6-4: Second stored procedure

The details of the second stored procedure are given in Figure 6-3 and Figure 6-4.

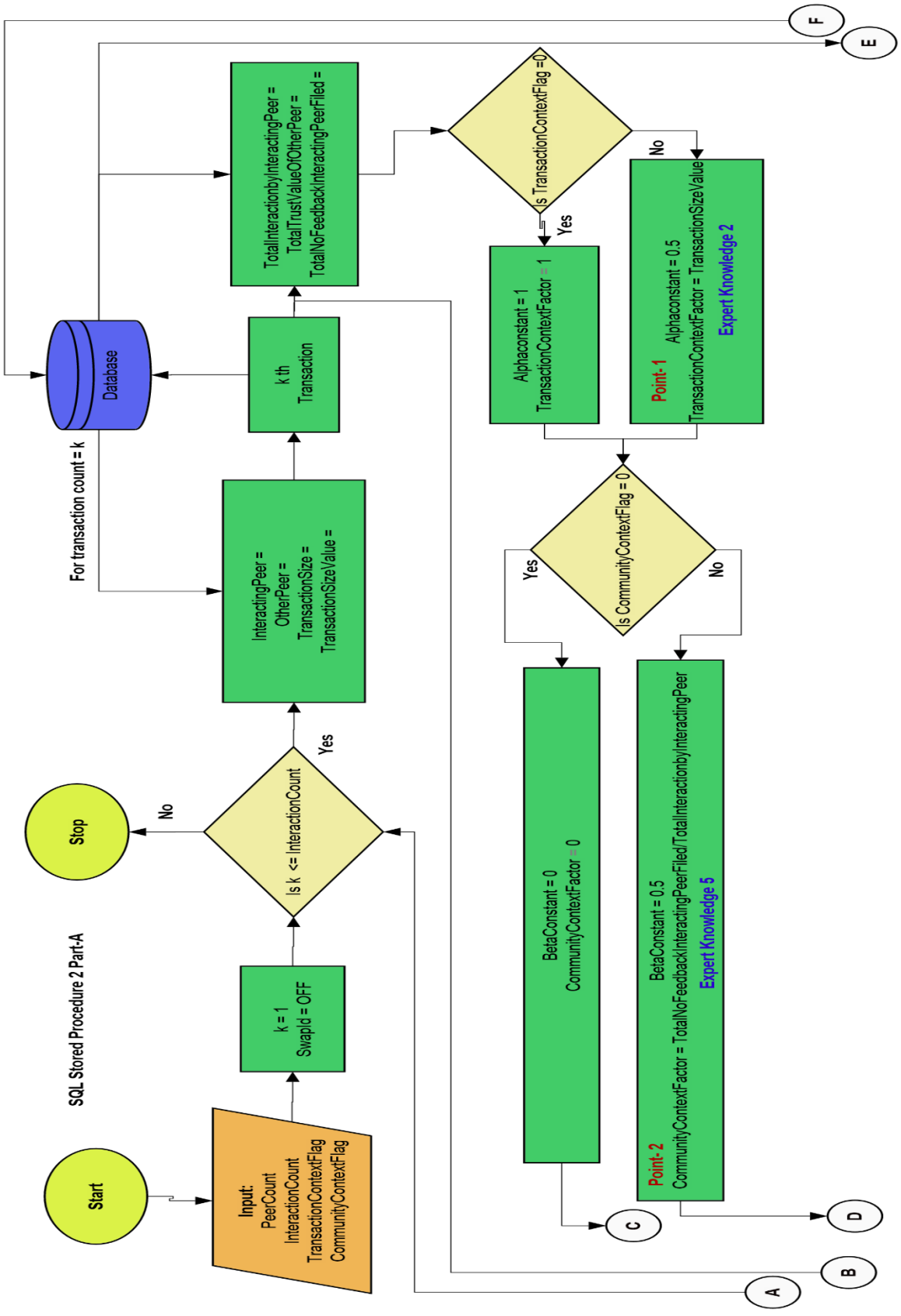


Figure 6-3: Stored procedure 2 - Part-A data flow

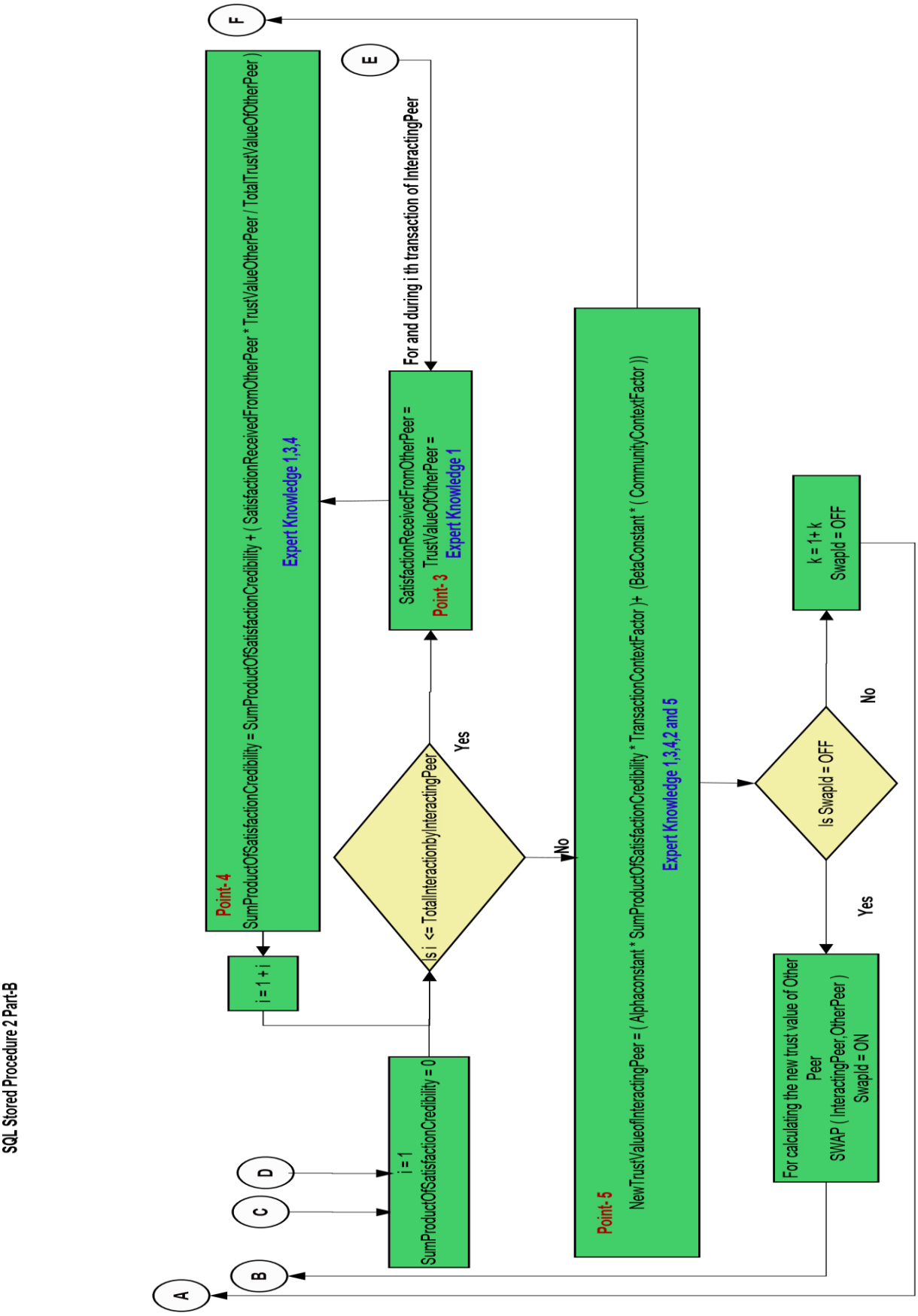


Figure 6-4: Stored procedure 2 - Part-B data flow

As shown in Figure 6-2, the first stored procedure will accept the following input values:

- The total number of peers that should be in the network
- The total number of interactions that need to be done

The stored procedure will then create the given number of peers with names, such as Peer1 and Peer2, etc., assign an initial trust value to each peer, and will decide on the given number of the interaction (which two peers should interact, the order of the given number of interactions that should happen, the transaction size and its value for each transaction) as outputs.

Figure 6-3 and Figure 6-4 provide the details of the second stored procedure as Part-A and Part-B respectively. The second stored procedure will accept the following input values.

- The total number of peers in the network
- The total number of interactions that should happen
- The transaction context flag (1 indicates that the trust calculation must consider the transaction context factor, 0 indicates that the trust calculation does not have to consider the transaction context factor)
- The community context flag (1 indicates that the trust calculation must consider the transaction community context factor, 0 indicates that the trust calculation does not have to consider the transaction community context factor)

The stored procedure will perform the given number of transactions in the decided order with the transaction size.

At Point-1 in Figure 6-3, Expert knowledge 2 which is described in Section 6.2.2 is applied to the PSTDG-PeerTrust model as the transaction context factor. The importance of the size of the transaction in the PeerTrust model's trust value calculation is explained in Section 2.4.6 and Section 6.2.3.2. The transaction size value will be assigned as the transaction context factor in the PSTDG-PeerTrust model.

Expert knowledge 5 which is described in Section 6.2.2 is applied at Point-2 in Figure 6-3 as the community context factor. In the PSTDG-PeerTrust model, the community context factor will be calculated as the ratio of the number of the feedback given by the interacting peer to other peers over the total number of transactions that the interacting peer has during that period. This is the same as how the community context factor is applied in the PeerTrust model which is described in Section 2.4.6 and Section 6.2.3.3.

In the PSTDG-PeerTrust model, the satisfaction received from the other peer is included as Expert knowledge 1 at Point-3 and Point-4 as shown in Figure 6-4. According to the basic trust matrix which is described in Section 2.4.6 and Section 6.2.3.1, the trust value for the interacting peer will be the weighted average of each transaction's satisfaction received by the interacting peer and the credibility of the peer who has submitted the feedback. This is applied in the PSTDG-PeerTrust model at Point-4 in Figure 6-4 as Expert knowledge 1, Expert knowledge 3 and Expert knowledge 4.

After each transaction, new trust values will be calculated according to the feedback received (Expert knowledge 1), the feedback's credibility (Expert knowledge 3), the total number of transactions done by the peer (Expert knowledge 1 and Expert knowledge 4), the transaction context factor (Expert knowledge 2), and the community context factor (Expert knowledge 5) using the equation described in Section 6.2.3.3. More details on this equation are given in Section 2.4.6. This is denoted as Point-5 in Figure 6-4 where all the expert knowledge identified in Section 6.2.2 is incorporated to calculate the new trust value for the interacting peer. Case 3 which is described in Section 6.2.3.3 is implemented as the PSTDG-PeerTrust model. The same procedure will be repeated to calculate the new trust value for the other peer taking part in each transaction. The new trust values for peers taking part in the transaction will be updated after each transaction. The PSTDG-PeerTrust model is developed, using the PSTDG Framework's first four steps.

The trust data generation model can be developed using any method for calculating new trust values. One of the available trust calculating methods must be chosen and expert knowledge must be derived by analysing it as shown in Section 6.2.2. All the knowledge regarding the data, including the characteristics of the required data, the analysis of the data to be generated, the identification of the fields to be generated, and the scope and the restrictions or rules that need to be used while generating the data set must be gathered and identified. Using the identified knowledge, constraints or formulas must be compiled as explained in Section 6.2.3. Finally, as in Section 6.2.4, a model or a program must be developed using the required constraints which satisfy all the identified knowledge or user requirements.

Even though Li and Ling (2004) are providing the trust calculation equation, the data generation model that was created using that equation did not necessarily produce a valid data set. Errors can happen in the program design, data management, and trust calculation. For these reasons, the data generation model's validation is required.

The PSTDG-PeerTrust model’s validation will be explained in the following section.

6.3 Step 2: PSTDG-PeerTrust model validation

The PSTDG-PeerTrust model is developed, using the Case 3 constraint, as explained in Section 6.2.3.3. The PSTDG-PeerTrust model must consider all of the five critical factors mentioned in Section 2.4.6 and Section 6.2.2 when calculating a peer's trust value. A valid PSTDG-PeerTrust model should be able to generate a trust data set that can satisfy all the expert knowledge identified in Section 6.2.2.

The validation of the developed PSTDG-PeerTrust model can be done by completing the same steps described in Section 4.6.2 and mentioned in Section 5.3. The steps mentioned in Section 5.3 will be applied to the PeerTrust model for determining the validity of the developed PSTDG-PeerTrust model.

The first step in the validation process is the compilation of What-if Scenarios regarding the calculation of trust values. As explained in Section 2.4.6, the PeerTrust model has five parameters for the calculation of trust values: the feedback a peer receives from other peers, the feedback scope (the number of transactions), the credibility of the feedback, transaction context factor and the community context factor. A valid data set should satisfy all the expert knowledge identified in Section 6.2.2 as described in Section 4.6.2. All the What-if Scenarios will be compiled using the expert knowledge identified in Section 6.2.2. According to Section 4.6.2, if the plotted graphs affirm the expectation of the identified expert knowledge, the data generation model can be said to be valid for its intended use. Trust data with scatter plots will be generated according to the compiled What-if Scenarios as described in Section 4.6.2. Thereafter, the plotted graphs will be analysed to determine if the generated trust data trends' visual representation matches the expert knowledge.

What-if Scenarios from Sections 6.3.1 to 6.3.4 and What-if Scenarios in Sections 6.3.7 to 6.3.8 involving a single parameter will be used for the validation of the PSTDG-PeerTrust model. What-if Scenarios in Sections 6.3.5 and 6.3.6 involves two parameters. All What-if Scenarios will be presented with a small description and parameters list. The trust data set visualised with scatter plots will subsequently be generated and analysed.

6.3.1 What-if Scenario 1 (Expert knowledge 1)

6.3.1.1 Compile

What is the effect on the **trust value** of Peer1 if the **satisfaction level** received from all other peers interacting with Peer1 is randomly **decreased**? The satisfaction level starts at a maximum

value of 1 and declines with a random value that lies between 0 and 0.001 in a series of transactions.

A series of transactions between Peer1 and other peers will be done. The satisfaction level received from other peers will be decreased with a random value between 0 and 0.001 for each transaction. After each transaction, the trust value of Peer1 will be calculated.

6.3.1.2 Generate and plot

The following parameters will be fixed in this scenario.

- The total number of transactions: 1 000
- The total number of peers: 1 001 (Peer1 to Peer1001), Peer1 interacting with all the other peers
- The transaction context value (transaction size value): 1
- The initial trust value of all peers: 1
- The weight factors α and β: 0.5

The new trust value of the interacting peer Peer1 will be plotted on the y-axis against the satisfaction received from the other peer on the x-axis for a series of 1 000 transactions in Figure 6-5.

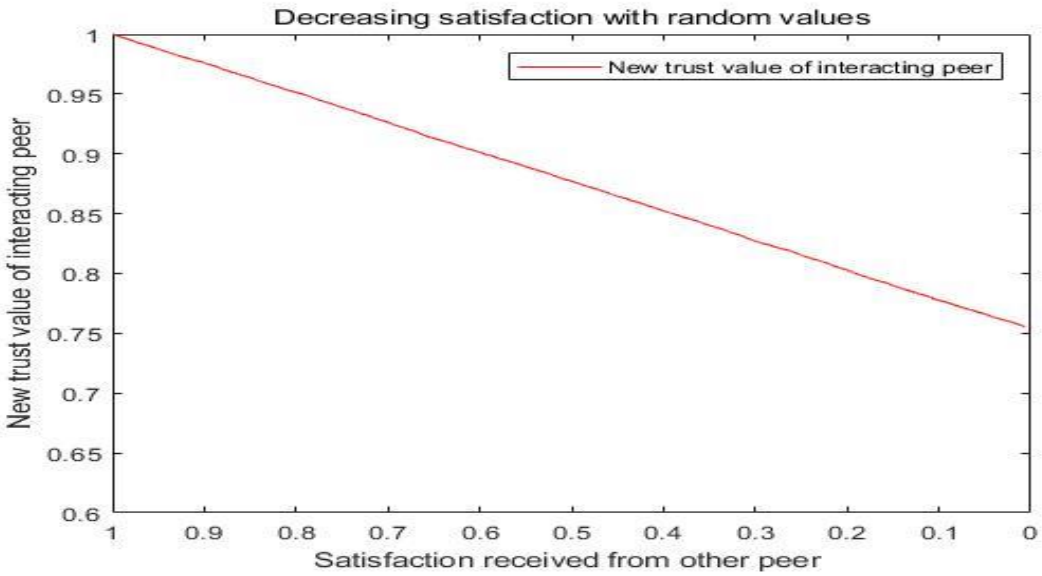


Figure 6-5: Trust value of 1 000 transactions when the satisfaction received is decreasing with random values

The new trust value of the interacting peer Peer1 will be plotted on the y-axis against the satisfaction received from other peers on the x-axis for a series of first 25 transactions in Figure 6-6.

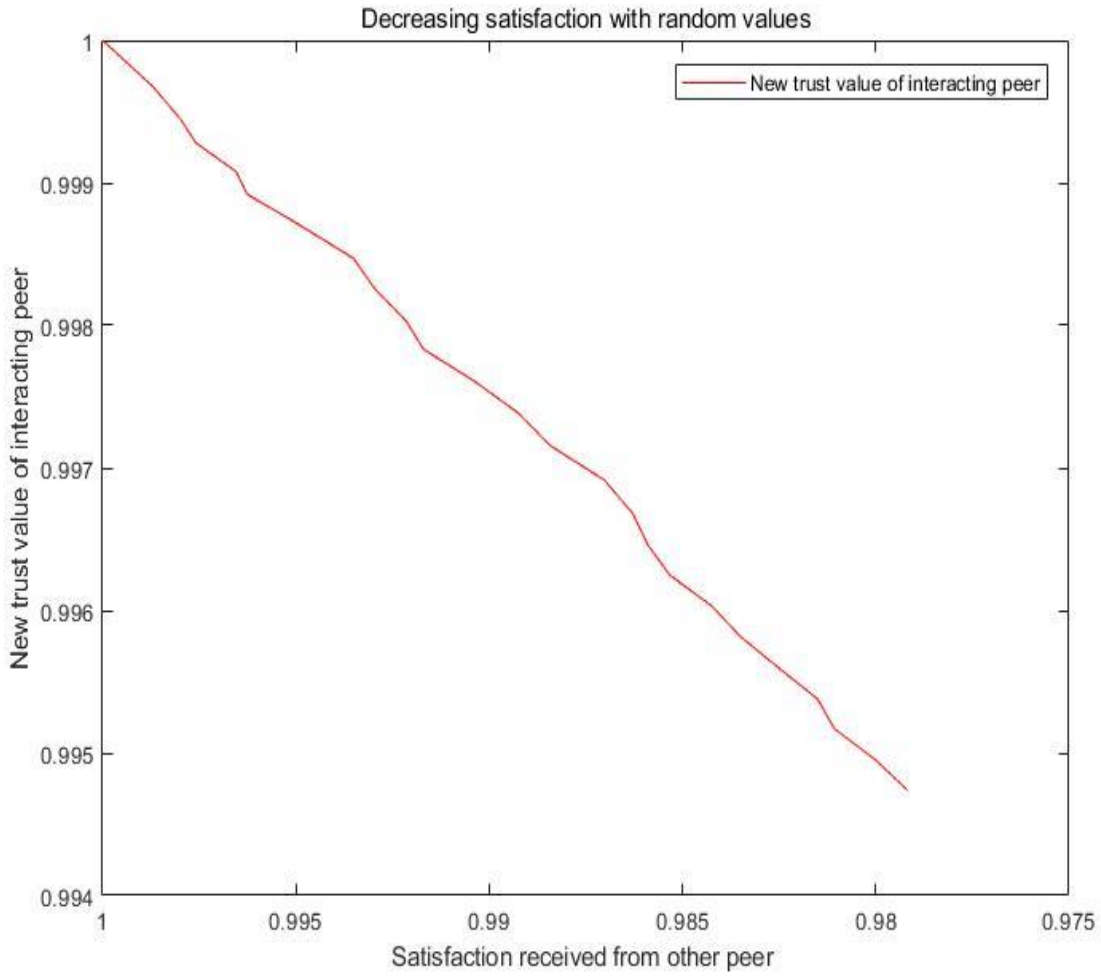


Figure 6-6: Trust value of first 25 transactions when the satisfaction received is decreasing with random values

6.3.1.3 Analysis

Figure 6-5 shows the graph of the new trust value of the interacting peer during 1 000 transactions when the satisfaction received from the other peer changes from maximum value (1) to a lower value, keeping all other parameters constant. Figure 6-6 shows the graph of the new trust value of the interacting peer during the first 25 transactions. The plot (Figure 6-5) for the larger sample looks like a straight line, but the plot (Figure 6-6) for the smaller sample is not a straight line. This difference in shape is due to the visual scale. However, both graphs' trend is exactly as expected.

According to Figure 6-5 and Figure 6-6, the parameter ‘satisfaction received from the other peer’ has an impact on the new trust value of the interacting peer, in such a way that the trust value of the interacting peer drops as the ‘satisfaction received from the other peer’ decreases. The plots show a very clear (nearly) linear relationship between the parameters ‘satisfaction received from the other peer’ and ‘new trust value of the interacting peer’.

For other scenarios, plots for the larger sample will be plotted. Since the randomised change will not be visible in the plot for the larger sample due to the scale problem, linear change in values will be used.

6.3.1.4 Conclusion

The plots in Figure 6-5 and Figure 6-6 show the new trust value is directly proportional to the satisfaction received and therefore the data satisfies Expert knowledge 1.

6.3.2 What-if Scenario 2 (Expert knowledge 1)

6.3.2.1 Compile

What will happen to the **trust valu**e of Peer1 when the **satisfaction level** received from the other peers **increases** linearly from a minimum value of (0) to a maximum value (0.999) in a series of transactions?

After each transaction, the new trust value of interacting peer Peer1 will be calculated when the satisfaction level received from the other peer increases linearly and keeps all other parameters constant. Here Peer1 will also be randomly interacting with all other Peers.

6.3.2.2 Generate and plot

The following will be the fixed parameters in this scenario.

- The total number of transactions: 1 000
- The total number of peers: 1 001 (Peer1 to Peer1001), Peer1 interacting with all the other peers
- The transaction context value (transaction size value): 1
- The initial trust value of all peers: 1
- The weight factors α and β: 0.5

The new trust value of the interacting peer Peer1 will be plotted on the y-axis against the satisfaction received from other peers on the x-axis for a series of 1 000 transactions in Figure 6-7.

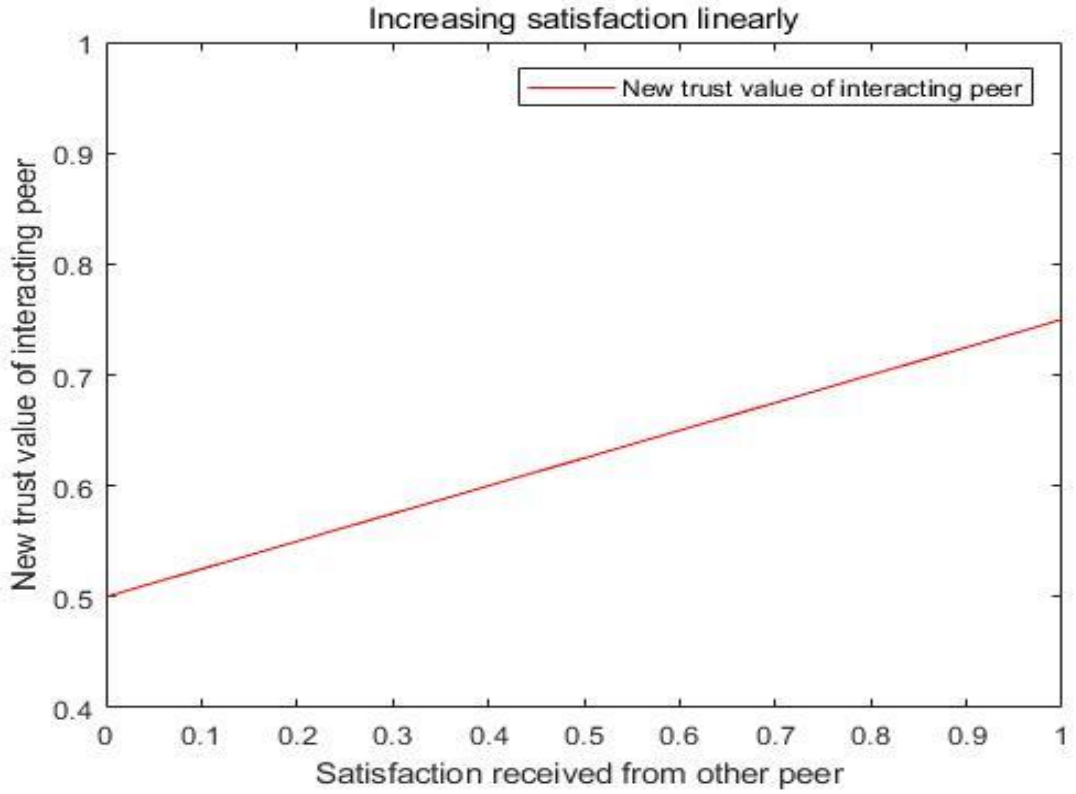


Figure 6-7: Trust value when the satisfaction received is increasing linearly

6.3.2.3 Analysis

Figure 6-7 shows that the interacting peer's trust value changes from 0.5 to 0.749 when 'satisfaction received from the other peer' changes from 0 to 0.999. When the satisfaction received from the other peer is 0, the new trust value of the interacting peer is 0.5. This value is determined only by the community context factor. The plots show a very clear (nearly) linear relationship between the parameters 'satisfaction received from the other peer' and 'new trust value of the interacting peer', as the trust value of the interacting peer increases when 'satisfaction received from the other peer' increases.

6.3.2.4 Conclusion

The plot in Figure 6-7 shows the new trust value is directly proportional to the satisfaction received and therefore the data satisfies Expert knowledge 1.

6.3.3 What-if Scenario 3 (Expert knowledge 2)

6.3.3.1 Compile

How does the **trust value** of Peer1 change when the **transaction context value** (transaction size value) **changes linearly** from maximum (1) to minimum value (0.001) in a series of transactions?

Interacting peer, Peer1, will be having 1 000 transactions. The new trust value of interacting peer, Peer1, will be calculated when the transaction size value decreases linearly and keeps all other parameters constant.

6.3.3.2 Generate and plot

The following will be the fixed parameters in this scenario.

- The total number of transactions: 1 000
- The total number of peers: 1 001 (Peer1 to Peer1001), Peer1 interacting with all the other peers
- The satisfaction received from the other peer: 1
- The initial trust value of all peers: 1
- The weight factors α and β: 0.5

The new trust value of the interacting peer, Peer1, will be plotted on the y-axis against the transaction size value on the x-axis for a series of 1 000 transactions in Figure 6-8.

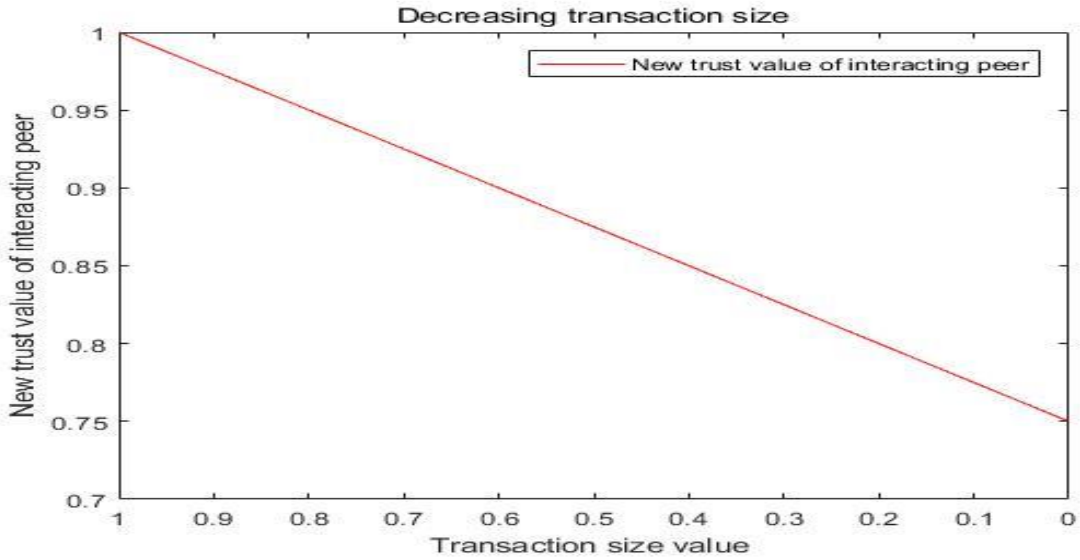


Figure 6-8: Trust value when the transaction size is decreasing linearly

6.3.3.3 Analysis

In Figure 6-8, the interacting peer's trust value changes from 1 to 0.750 when ‘transaction size’ changes from 1 to 0.001. The trust value of the interacting peer is dropping when the ‘transaction size’ decreases. The plot shows a very clear (nearly) linear relationship between the parameters ‘transaction size’ and ‘new trust value of the interacting peer’.

6.3.3.4 Conclusion

The plot in Figure 6-8 shows that the trust value is directly proportional to the transaction size and therefore, the data satisfies Expert knowledge 2.

6.3.4 What-if Scenario 4 (Expert knowledge 2)

6.3.4.1 Compile

What will happen to the **trust value** of Peer1 when the **transaction context value** (transaction size value) **changes linearly** from minimum (0.001) to maximum value (1) in a series of transactions?

After each transaction, the new trust value of interacting peer, Peer1, will be calculated when the transaction size value increases linearly and keeps all other parameters constant. Here Peer1 will also be randomly interacting with other peers in 1 000 transactions.

6.3.4.2 Generate and plot

The following will be the fixed parameters in this scenario.

- The total number of transactions: 1 000
- The total number of peers: 1 001 (Peer1 to Peer1001), Peer1 interacting with all the other peers
- The satisfaction received from the other peer: 1
- The initial trust value of all peers: 1
- The weight factors α and β: 0.5

The new trust value of the interacting peer, Peer1, will be plotted on the y-axis against the transaction size value on the x-axis for a series of 1 000 transactions in Figure 6-9.

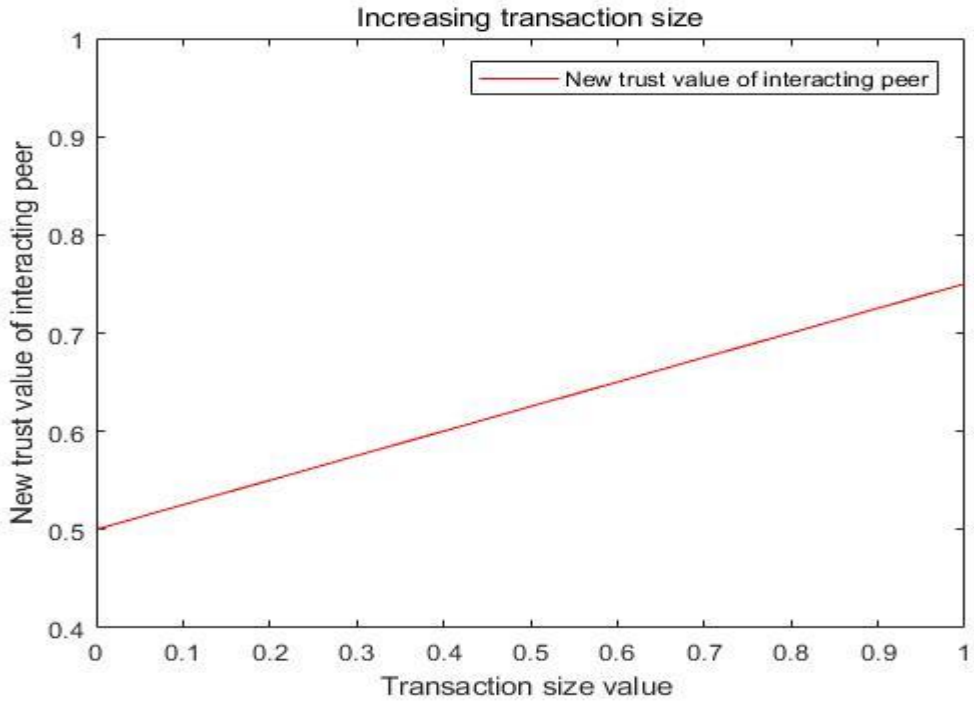


Figure 6-9: Trust value when the transaction size is increasing linearly

6.3.4.3 Analysis

According to Figure 6-9, the interacting peer's trust value changes from 0.5005 to 0.750 when 'transaction size' changes from 0.001 to 1. The trust value of the interacting peer is increasing when the 'transaction size' increases. The size of the transaction is playing a vital role in calculating the trust value of the interacting peer. Hence the trust value is directly proportional to the transaction size.

6.3.4.4 Conclusion

The plot in Figure 6-9 shows that the trust value is directly proportional to the transaction size and therefore, the data satisfies Expert knowledge 2.

6.3.5 What-if Scenario 5 (Expert knowledge 3)

6.3.5.1 Compile

What will happen to the **trust value** of Peer1 when the **satisfaction level** received from all other peers interacting with Peer1 **increases** linearly and the **trust value** of the other peer **changes** in a **particular transaction**, keeping all other parameters constant? The satisfaction level starts at a minimum value of 0.1 and linearly increases to a maximum value of 0.9. The other peer's trust value will be a minimum value of 0.1 for all transactions except one transaction.

In this scenario, the new trust value of interacting peer, Peer1, will be calculated after each transaction. The **satisfaction level** received from the other peer will be linearly increasing (0.1 to 0.9) during a series of five transactions. The other peer's **trust value** will be a value of 0.1 for all transactions except for the last transaction and all other parameters will be a constant. The five transactions will be repeated three times, and the trust value of the last peer will be changed for each iteration as follows:

1. When the other peer's trust value is 0.2 in the last transaction
2. When the other peer's trust value is 0.5 in the last transaction
3. When the other peer's trust value is 1 in the last transaction

6.3.5.2 Generate and plot

The following will be the fixed parameters in this scenario.

- The total number of transactions: 5
- The total number of peers: 6 (Peer1 to Peer6), Peer1 interacting with all the other peers.
- The transaction context value (transaction size value): 1
- The initial trust value of the interacting Peer: 0.1
- The weight factors α and β: 0.5

The new trust value of the interacting peer, Peer1, will be plotted on the y-axis against the satisfaction received from other peers on the x-axis for all the three cases in Figure 6-10.

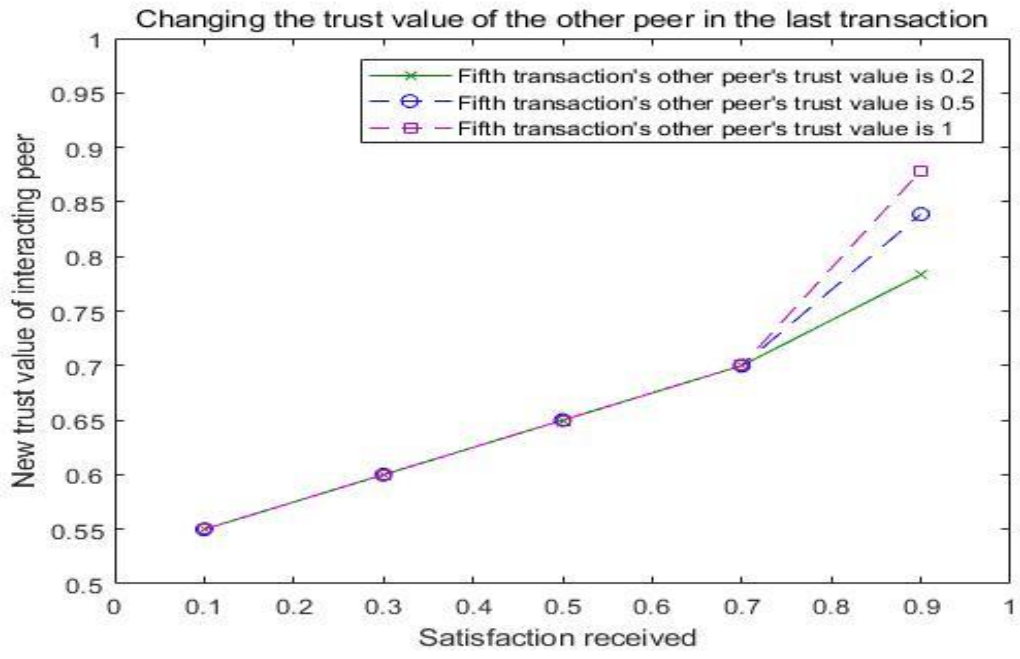


Figure 6-10: Changing the trust value of the other peer in the last transaction

6.3.5.3 Analysis

In Figure 6-10, all three graphs show the interacting peer's trust value changes from 0.55 to 0.7 when the satisfaction received from the other peer changes from 0.1 to 0.7 keeping all other parameters constant. Hence, the trust value of the interacting peer remains the same in all three cases during the first four transactions.

In case 1, for the last transaction, the other peer's trust value is 0.2 and the satisfaction received from the other peer is 0.9. In case 2, for the last transaction, the other peer's trust value is 0.5 and the satisfaction received from the other peer is 0.9. In case 3, for the last transaction, the other peer's trust value is 1 and the satisfaction received from the other peer is 0.9. Hence the satisfaction received from the other peer is the same for all three cases. The only difference is the trust value of the peer who submitted the feedback.

In Figure 6-10, there is an increase in the interacting peer's trust value when the other peer's trust value is increased. From Figure 6-10, it is clear that the satisfaction received from the other peer is the same in all three cases. Hence the other peer's trust value is impacting the interacting peer's trust value. It is evident that feedback from the more trusted peer is having more weight or contributes more to the trust value of the interacting peer and hence changing the trust value of the interacting Peer1. Therefore, as the trust value of the "other peer" is increased, it has an increased weight in the calculation of trust value.

6.3.5.4 Conclusion

The plot in Figure 6-10 shows feedback from the more trusted peer has more weight when calculating the trust value. The data therefore satisfies Expert knowledge 3.

6.3.6 What-if Scenario 6 (Expert knowledge 3)

6.3.6.1 Compile

What will happen to the **trust value** of Peer1 when the **trust value** of the other peers interacting with Peer1 changes and the **satisfaction level** received from all other peers interacting with Peer1 **increases** linearly from the minimum value (0) to a maximum value (0.999) in a series of transactions?

After each transaction, the new trust value of interacting peer, Peer1, will be calculated by keeping the satisfaction value received from the other peer linearly increasing and the other peer's trust value changing with all other parameters constant. Here Peer1 will also be

interacting with other peers in 1 000 transactions. The above transactions will be repeated by changing the other peer's trust value.

1. The **trust value** of the other peer **decreases** linearly from a maximum value (1) to a lower value (0.001).
2. The **trust value** of the other peer **increases** linearly from a minimum value (0.001) to a higher value (1)

After each transaction, the total amount of satisfaction received by Peer1 will be the same for both cases. However, the trust value of the peer submitting the feedback will be different. Nevertheless, after finishing the last transaction, the total sum of the trust value of the other peers who submitted the feedback to Peer1 will be the same in both cases.

6.3.6.2 Generate and plot

The following will be the fixed parameters in this scenario.

- The total number of transactions: 1 000
- The total number of peers: 1 001 (Peer1 to Peer1001), Peer1 interacting with all the other peers
- The transaction context value (transaction size value): 1
- The initial trust value of the interacting Peer: 1
- The weight factors α and β: 0.5

The new trust value of the interacting peer, Peer1, will be plotted on the y-axis against the satisfaction received from other peers on the x-axis for both cases in Figure 6-11.

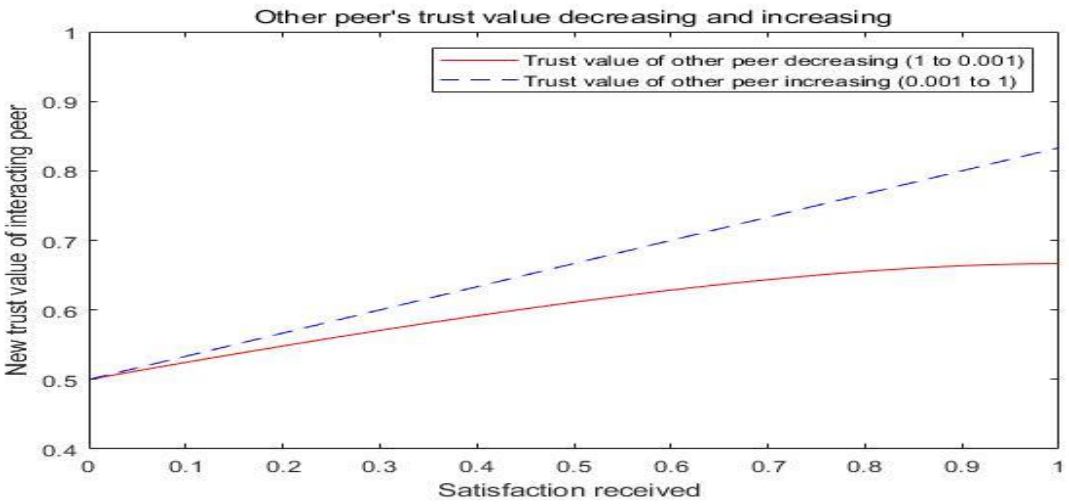


Figure 6-11: Increasing satisfaction received and increasing the trust value of the other peer

6.3.6.3 Analysis

The interacting peer's trust value changes from 0.5 to 0.833 in Figure 6-11, when the satisfaction received from the other peer changes from 0 to 0.999. The new trust value of the interacting peer is directly proportional to the satisfaction received.

In Figure 6-11, when the trust value of the other peer increases from 0.001 to 1, the interacting peer's trust value changes from 0.5 to 0.833 when the satisfaction received from the other peer changes from 0 to 0.999 and all other parameters are kept constant. When the trust value of the other peer decreases from 1 to 0.001, the interacting peer's trust value changes from 0.5 to 0.666 when the satisfaction received from the other peer changes from 0 to 0.999 and all other parameters are kept constant.

After each transaction, the total amount of satisfaction received by Peer1 is the same for both cases. After finishing the last transaction, the total sum of the trust value of the other peers who submitted the feedback to Peer1 is also the same in both cases. However, the obtained trust value of Peer1 is different in both cases. This difference in the trust value of Peer1 shows that the feedback from the more trusted peer is having more weight or contributes more to the trust value of the interacting peer.

6.3.6.4 Conclusion

Figure 6-11 shows the feedback from the more trusted peer has more weight when calculating the trust value. Hence the data satisfies Expert knowledge 3.

6.3.7 What-if Scenario 7 (Expert knowledge 4)

6.3.7.1 Compile

What will happen to the **trust value** of Peer1 when Peer1 interacts with other peers, but the trust value calculation **does not include all previous transactions**?

After each transaction, the new trust value of interacting peer, Peer1, will be calculated. Here Peer1 will be interacting with other peers in a series of 500 transactions.

The satisfaction level received from other peers will be randomly changing for each transaction. After each transaction, the trust value of Peer 1 will be calculated. Then the following cases will be performed:

1. Peer1 will interact with Peer502 as the 501st transaction.
2. Peer1 will interact with Peer502 as if it was its first transaction.

To keep the above two transactions under the same condition, the **satisfaction level** received from Peer502 will be fixed with a value of 0.1. The trust value of Peer1 will be calculated for both cases after the transaction.

6.3.7.2 Generate and plot

The following will be the fixed parameters in this scenario.

- The total number of transactions: 501
- The total number of peers: 502 (Peer1 to Peer502), Peer1 interacting with all the other peers
- The transaction context value (transaction size value): 1
- The initial trust value of the interacting Peer: 0.1
- The weight factors α and β: 0.5

The new trust value of the interacting peer, Peer1, will be plotted on the y-axis against the transaction count on the x-axis for the first 501 transactions in Figure 6-12.

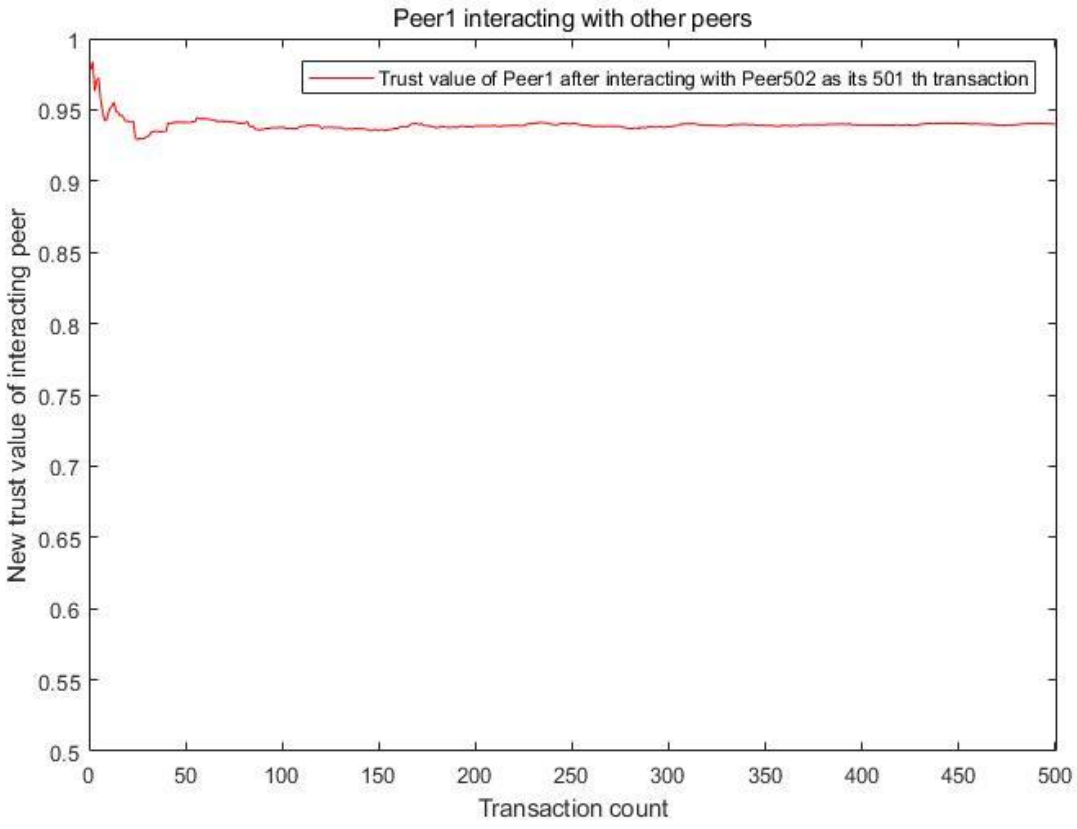


Figure 6-12: Trust value of interacting peer after interacting with other peers

6.3.7.3 Analysis

The interacting peer, Peer1's, trust value becomes 0.939 in Figure 6-12 when Peer1 interacts with Peer502 as the 501st transaction. However, the interacting peer, Peer1's, trust value is 0.55 when Peer1 interacts with Peer502 as if it was its first transaction. In both cases, transactions happened under the same conditions. The only difference was, in case 1, the interacting peer, Peer1, had 500 other transactions before interacting with Peer502. The trust value of Peer502 was the same for both cases. The **satisfaction level** received from Peer502 was a very low value of 0.1 for both cases. However, the interacting peer Peer1 is having a high trust value in case 1 and a lower trust value in case 2. Figure 6-12 also shows that the trust value of the interacting peer, Peer1, is stabilising after 500 transactions. This shows that in the long run, the interacting peer, Peer1, has a high trust value, which remains stable when calculating the trust value. Thus the trust value is calculated using all the feedback received during all previous transactions.

6.3.7.4 Conclusion

Figure 6-12 shows that the trust value is calculated using all the feedback received during all previous transactions. Hence the generated data satisfies Expert knowledge 4.

6.3.8 What-if Scenario 8 (Expert knowledge 5)

6.3.8.1 Compile

What will happen to the **trust value** of Peer1 when the interacting peer **fails** to give **feedback** during a set of transactions, keeping all other parameters constant in a series of transactions?

The new trust value of interacting peer, Peer1, will be calculated after each transaction. There will be 1 000 transactions. During the first 250 transactions, the interacting peer, Peer1, will give feedback to the other peers. For the next 250 transactions, Peer1 will not give feedback to other peers, then it will start giving feedback for the next 250 transactions and for the last 250 transactions, Peer1 will not give feedback to other peers, keeping all other parameters constant. Peer1 will be randomly interacting with all other Peers.

6.3.8.2 Generate and plot

The following will be the fixed parameters in this scenario.

- The total number of transactions: 1 000

- The total number of peers: 1 001 (Peer1 to Peer1001), Peer1 interacting with all the other peers
- The transaction context value (transaction size value): 1
- The initial trust value of all peers: 1
- The weight factors α and β: 0.5

In this scenario, two graphs will be plotted using the generated data set. The first graph is to visualise the change in feedback status of Peer1 during each transaction and the second graph is to visualise the change in trust value after each transaction. Hence, the transaction count will be on the x-axis for both graphs. A change in trust value due to the change in feedback status (the interacting peer fails to give feedback) can be analysed by comparing the two graphs.

The status of the feedback (whether given or not) given by the interacting peer, Peer1, will be plotted on the y-axis against the series of transactions (transaction count) on the x-axis in Figure 6-13.

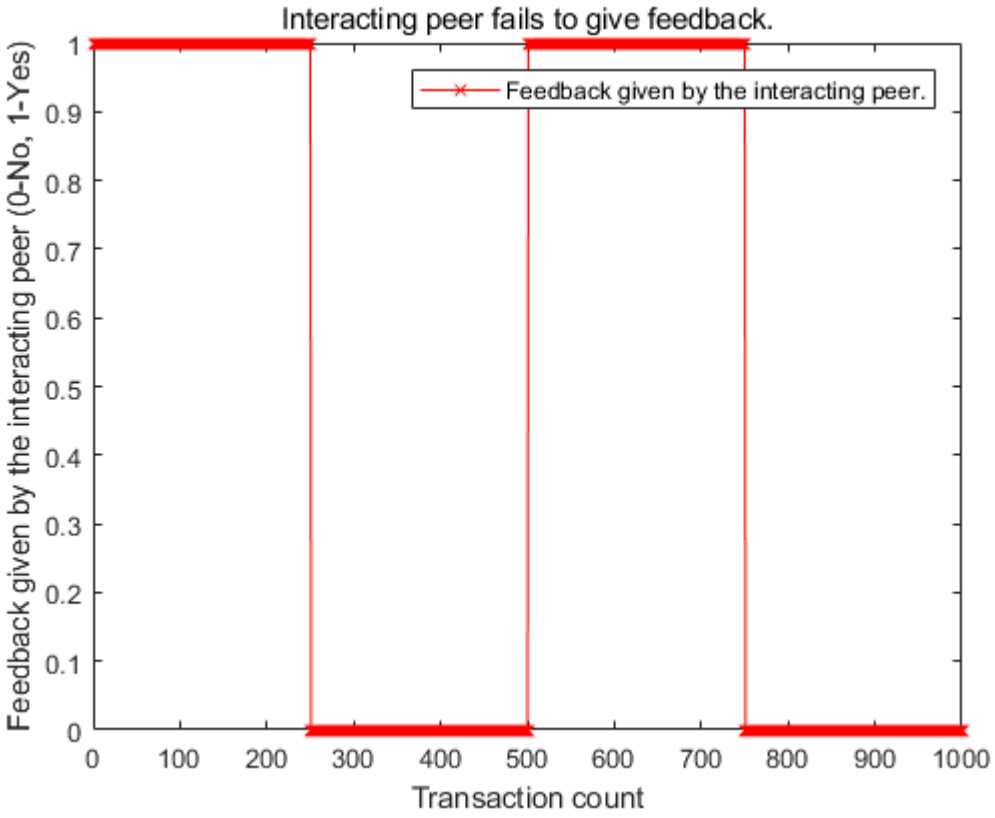


Figure 6-13: Interacting peer fails to give feedback

The new trust value of the interacting peer, Peer1, will be plotted on the y-axis against the series of transactions (transaction count) on the x-axis in Figure 6-14.

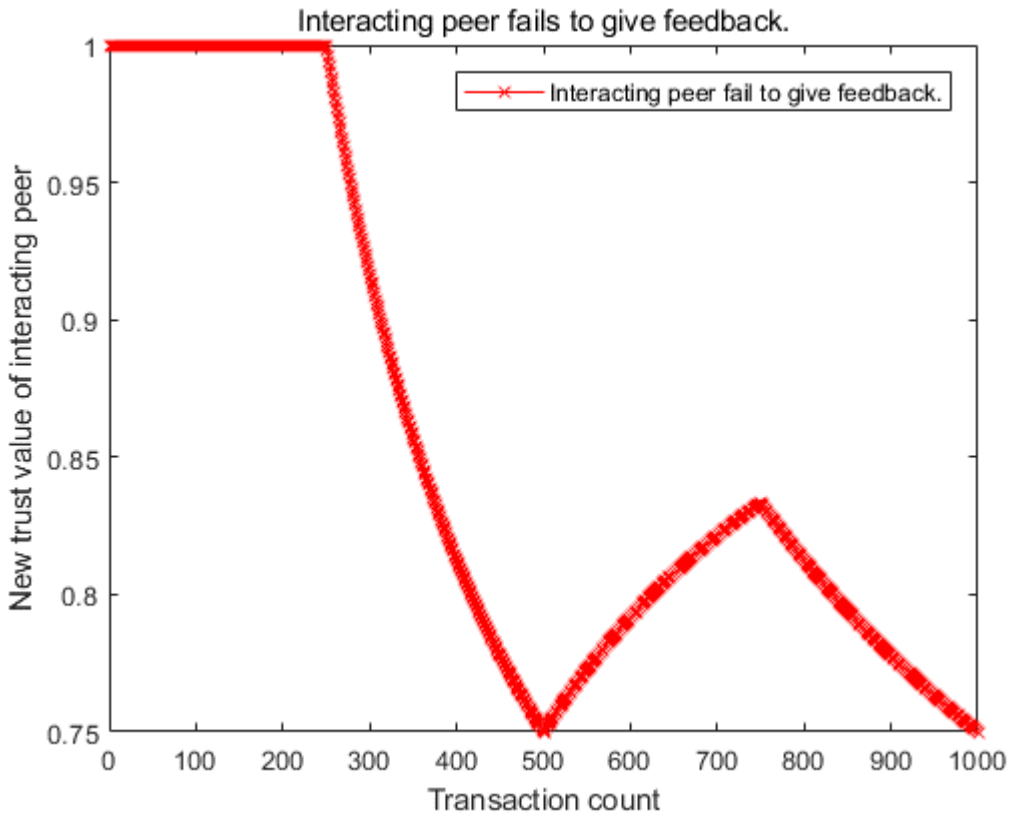


Figure 6-14: Trust value when the interacting peer fails to give feedback

6.3.8.3 Analysis

Figure 6-13 and Figure 6-14 show what will happen when the interacting peer fails to give feedback to the other peer during the transaction, keeping all other parameters constant. According to Figure 6-13, the interacting peer gave feedback from transaction count 1 to 250 and from transaction count 501 to 750. During all other transactions, the interacting peer failed to provide feedback to the other peer. Figure 6-14 shows the interacting peer's trust value decreases slowly from transaction count 251 to 500, then the trust value of the interacting peer increases up to transaction count 750. The trust value of the interacting peer is again declining slowly from transaction count 751 to 1 000. This shows that the interacting peer's trust value depends upon the community context factor.

6.3.8.4 Conclusion

Figure 6-13 and Figure 6-14 show that the interacting peer's trust value depends upon the community context factor. Hence, the generated data satisfies the Expert knowledge 5.

6.3.9 Conclusion

The above graphs and the analysis give the following indications.

1. The trust value is directly proportional to the satisfaction received (Expert knowledge 1) according to the What-if Scenarios 6.3.1 and 6.3.2 and the conclusions in Section 6.3.1.4 and Section 6.3.2.4.
2. As per the What-if Scenarios in Section 6.3.3 and Section 6.3.4 and its conclusions in Section 6.3.3.4 and Section 6.3.4.4, the trust value is directly proportional to the transaction size (Expert knowledge 2).
3. What-if Scenarios in Section 6.3.5 and Section 6.3.6 and its conclusions in Section 6.3.5.4 and Section 6.3.6.4 shows that the feedback from the more trusted peer will have more weight. The credibility of the feedback will also affect trust value (Expert knowledge 3).
4. Trust value is calculated using all the feedback received during all previous transactions (Expert knowledge 4) according to the What-if Scenario in Section 6.3.7 and its conclusion in Section 6.3.7.4.
5. According to the What-if Scenario in Section 6.3.8 and its conclusion in Section 6.3.8.4, peers who submit the feedback will be rewarded through the community context factor (Expert knowledge 5).

The above indications show that the PSTDG-PeerTrust model generates a data set for the PeerTrust algorithm which satisfies all the expert knowledge identified in Section 6.2.2. The developed PSTDG-PeerTrust model can be used to generate a valid trust data set.

Pure synthetic trust data set generation, using the PSTDG-PeerTrust model to train the PeerTrustRBFNN trust calculation model, will be discussed in the next section.

6.4 Step 3: Pure synthetic trust data set generation for training the PeerTrustRBFNN trust calculation model

The PSTDG-PeerTrust model, which is described in Section 6.2.4, is validated in Section 6.3. In this section, the pure synthetic trust data set will be generated using this validated PSTDG-PeerTrust model. The pure synthetic trust data set will be generated using the last two processes of the PSTDG Framework explained in Section 4.7. Before developing the data set, a set of input parameter values that needs to be used for the data set generation was selected as described in Sections 4.7 and 5.4.

In this study, as discussed in Section 6.2.3.3, the basic trust matrix with the transaction context and community context factors was used to calculate trust. This calculation will consider all the five critical factors for evaluating a peer's trust value. Hence, the trust value will be calculated using Expert knowledge 1, Expert knowledge 2, …, Expert knowledge 5. To give equal importance to the five critical factors, the weight factors α and β will be set to the same value, namely 0.5.

To generate the pure synthetic trust data set, the following two stored procedures as described in Section 6.2.4 will be used:

1. *2021PeerTrustSetupSameInputP2P* which is given in Appendix A.1
2. *2021PeerTrustNewSameInputP2P* which is given in Appendix A.2

As explained in Section 5.4 and shown in Figure 6-1, pure synthetic trust data generation using the developed model consists of two processes: feed input and generate. In this demonstration, the following parameters will be set in the first stored procedure:

- *PeerCount*: 25
- *InteractionCount*: 10 000

In addition, the following parameter values will be set in the second stored procedure:

- *PeerCount*: 25
- *InteractionCount*: 10 000
- *TransactionContextFlag*: 1
- *CommunityContextFlag*: 1

The data set size needs to be at least a factor of 50 to 1 000 times the number of prediction classes (Alwosheel *et al.*, 2018). *NewTrustValueofInteractingPeer* and *NewTrustValueofOtherPeer* are the two prediction classes in this experiment. There must be at least 2 000 samples in the data set. The *InteractionCount* is set to 10 000 to satisfy the above-mentioned rule-of-thumb. The 10 000 transactions provide 10 000 samples.

The *TransactionContextFlag* and *CommunityContextFlag* are set to 1 to satisfy the constraints described in Section 5.2.3.3. It will consider all of the five critical factors for evaluating a peer's trust value.

After executing the above-mentioned two stored procedures, 25 peers will be created and 10 000 transactions will be performed using these peers. A data set with the following information will be generated:

1. *InteractingPeer* - Name of the interacting peer, for example, *Peer1, Peer2*, etc.

2. *OtherPeer* - Name of the other peer participating in the interaction, for example, *Peer1, Peer2*, etc.
3. *TransactionSize* - Size of the transaction, for example, *Small*, *Large*, etc.
4. *TransactionSizeValue* - Value of the transaction which lies between 0 and 1.
5. *FeedbackGivenByOtherPeer* - Indicates whether the *InteractingPeer* received feedback from the *OtherPeer*. A 1 indicates yes, and a 0 indicates no.
6. *SatisfactionReceivedFromOtherPeer* - A value between 0 and 1 indicates the amount of satisfaction received from the *OtherPeer*.
7. *FeedbackGivenByInteractingPeer* - Indicates whether the *OtherPeer* received feedback from the *InteractingPeer*. A 1 indicates yes, and a 0 means no.
8. *SatisfactionReceivedFromInteractingPeer* - A value between 0 and 1 indicates the amount of satisfaction received from the *InteractingPeer*.
9. *CurrentTrustValueofInteractingPeer* - A value that lies between 0 and 1 indicates the current trust value of *InteractingPeer*.
10. *CurrentTrustValueOfOtherPeer* - A value that lies between 0 and 1 indicates the current trust value of the *OtherPeer*.
11. *TotalNoFeedbackInteractingPeerFiled* - An integer that indicates the total number of feedback the *InteractingPeer* has given.
12. *TotalNoFeedbackOtherPeerFiled* - An integer that indicates the total number of feedback the *OtherPeer* has given.
13. *NewTrustValueofInteractingPeer* - A value that lies between 0 and 1 indicates the new trust value of the *InteractingPeer* after the transaction.
14. *NewTrustValueofOtherPeer* - A value that lies between 0 and 1 indicates the new trust value of the *OtherPeer*.

The construction of the RBFNN model, named PeerTrustRBFNN, using the generated data set, will be discussed in the next section.

6.5 Step 4: Construction of the PeerTrustRBFNN model using the generated data set

This experimental demonstration aims to develop an alternative trust calculation method using an RBFNN for the PeerTrust model by Li and Ling (2004). As described in Section 5.5, to build an accurate RBFNN model that can determine trust values, suitable model hyperparameters must be chosen to train the model. This will be done by training several candidate PeerTrustRBFNN models using uniform randomly sampled hyperparameters and selecting the best PeerTrustRBFNN model. The PeerTrustRBFNN model for trust calculation will be built on the generated trust data set. This process will also include the following three steps: data pre-

processing, identifying the best PeerTrustRBFNN model, and evaluating the best model as described in Section 5.5.

6.5.1 Data pre-processing

The need for data pre-processing is explained in Section 5.5.1. The data set obtained in Section 6.4 needs to be pre-processed before constructing the PeerTrustRBFNN model, since the generated inputs vary across different ranges. As mentioned in Section 5.5.1, this may improve the accuracy of the PeerTrustRBFNN model, decrease the computational cost, and accelerate the learning process; the pre-processing of the input variables also helps to better match the predicted output.

The following input values need to be converted into numeric binary values, as these input values consist of binary text values:

- *InteractingPeer*
- *OtherPeer*
- *TransactionSize*

In this study, the *PeerCount* will be set to 25, as mentioned in Section 6.4. An ANN can work only with numerical data (Koval, 2018). Each peer name will be converted to one-hot encoded binary values using 25 inputs. The input *TransactionSize* can have any of the four values mentioned in Section 6.2.4. Each *TransactionSize* will be converted to one-hot encoded binary values, using four binary inputs.

For the ANN to treat the values equally, data must lie in the same range (Koval, 2018). The following inputs must be normalised to a range of [0, 1] as these have data ranges that vary across a wide interval (Zhou & Ooka, 2021):

- *TransactionSizeValue*
- *SatisfactionReceivedFromOtherPeer*
- *SatisfactionReceivedFromInteractingPeer*
- *CurrentTrustValueofInteractingPeer*
- *CurrentTrustValueOfOtherPeer*
- *TotalNoFeedbackInteractingPeerFiled*
- *TotalNoFeedbackOtherPeerFiled*

To do the above-required data pre-processing, the *2021PeerTrust_Do_BinaryNorm* stored procedure, written using MS SQL code, will be used. This procedure can be found in Appendix

A.3. The input for the above-mentioned stored procedure will be the trust data set obtained in Section 6.4, and the output will be a data set having binary and normalised values. After pre-processing has been performed, identification of the best RBFNN trust model will be discussed in the next section.

6.5.2 Identifying the best RBFNN model for trust calculation

In this section, experiments will be done to identify the best model hyperparameters for the RBFNN model, called PeerTrustRBFNN. The program utilised for this purpose is written in the Python programming language with the TensorFlow library and Keras application program interface (API) used for RBFNN training, as mentioned in Section 5.5.2. The program code is given in Appendix A.4. As explained in Section 5.5.2, before the search for the best hyperparameters is performed, the following hyperparameter search space is defined:

Hyperparameter	Description
beta_min: 0.	The RBFNN beta value
beta_max: 2.	
hidden_nodes_min: 1.	The number of hidden RBFNN nodes
hidden_nodes_max: 200.	
window_size_min: 1.	The training set window size
window_size_max: 15.	
learning_rate: $10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5},$ or 10^{-6}.	The RBFNN learning rate
hours_to_run: 24.	Maximum time allowed for finding the best hyperparameters.
epochs_value: 100.	The number of epochs per trial

Table 6-5: Hyperparameter search space

The computation time increases as the *epochs_value* increases. Increasing the *epochs_value* will not necessarily produce a good result. The default *learning_rate* of the optimiser is 0.001. Increasing the number of hidden nodes increases the complexity of the model and hence the runtime will also increase. The accuracy of the model will decrease when the number of hidden

nodes increases after a certain value (Gulli *et al.*, 2019). Therefore, obtaining an optimal value for the hyperparameter is necessary.

The following search algorithm will be used to determine the best hyperparameters:

Algorithm: Identifying the best RBFNN model for trust calculation.

Input: Hyperparameter search space specified in Table 6-5, generated trust data set and time limit.

Output: Hyperparameter values for the best RBFNN model: *NumberOfHiddenNodes*, *WindowSize*, *betas_value*, *learning_rate*, *epochs_value* and MSE error on the test set.

1. While (Time limit has not passed) do
2. Obtain random values for the *betas_value*, *WindowSize*, *NumberOfHiddenNodes* and *learning_rate* hyperparameter variables using the ranges specified in Table 6-5.
3. Create a data set called *Trust2021* by converting the generated trust data set into input windows of *WindowSize* sizes.
4. Randomly sample a training set (70%), validation set (20%) and test set (10%) from the *Trust2021* data set.
5. Train an RBFNN model using the hyperparameters sampled in Step 2 and determine the MSE error on the validation set.
6. Record the randomly sampled hyperparameter values, and the MSE error on the validation set and the test set.
7. End while
8. Evaluate the best RBFNN model found by Steps 1 to 7 (in terms of the MSE error on the validation set) on the test set.

Algorithm 1: Identifying the best RBFNN model for trust calculation

In the next section, the evaluation of the best RBFNN trust model found will be described.

6.5.3 Evaluation of the best RBFNN model for trust calculation

In the allowed time (24 hours run on an Intel® Core™ i7-8550U CPU at 1.80GHz laptop with 8.00 GB RAM), 162 candidate models were explored. The best RBFNN model found by Algorithm 1 has the following hyperparameter values:

- *NumberOfHiddenNodes*: 72
- *WindowSize*: 1
- *betas_value*: 0.0713706285882029
- *learning_rate*: 0.001 and
- *epochs_value*: 100.

The MSE on the validation set has a value of 3.84893×10^{-4}. The MSE on the test set is 2.58369×10^{-4}. The MSE value closer to zero shows better model performance (Elzwayie *et al.*, 2017).

The requirement of large trust data sets which was the most important challenge for the training of an RBFNN is solved by the development of the PSTDG-PeerTrust model. The next problem identified was the validation of the pure synthetic trust data set. Using What-if analysis and sensitivity analysis, the validation of the PSTDG-PeerTrust model is also done using the new definition which was given for the validation of the pure synthetic trust data set in Section 4.6.1. Therefore, the problem of the validity of pure synthetic data trust data set was solved. This solved the problem of not having enough data for the training of the RBFNN model. An RBFNN model, called PeerTrustRBFNN, was constructed for the calculation of trust for the PeerTrust, using the generated data set. The best model was developed, tested, and validated. In this way, the main aim of "an alternative way of calculating trust using an RBFNN" was achieved.

6.6 Conclusion

In this chapter, a data generation model called the "PSTDG-PeerTrust model" was developed, using the PeerTrust model by Li and Ling (2004). This was done based on the PSTDG Framework described in the Section 4.7. The PSTDG-PeerTrust model was validated, and a large data set was generated. The construction of the RBFNN trust model, called the PeerTrustRBFNN, was carried out for a purely theoretical trust calculation algorithm, namely the PeerTrust model by Li and Ling (2004). The demonstration was executed, using the experimental design described in the previous chapter. In Chapter 7, the demonstration of the proposed solution of generating a pure synthetic trust data set and the development of an RBFNN model to calculate trust for a real-world problem will be carried out. The context will be restricted to availability and durability. This demonstration will be undertaken, using the ARDS (Amazon, 2019) as an example.

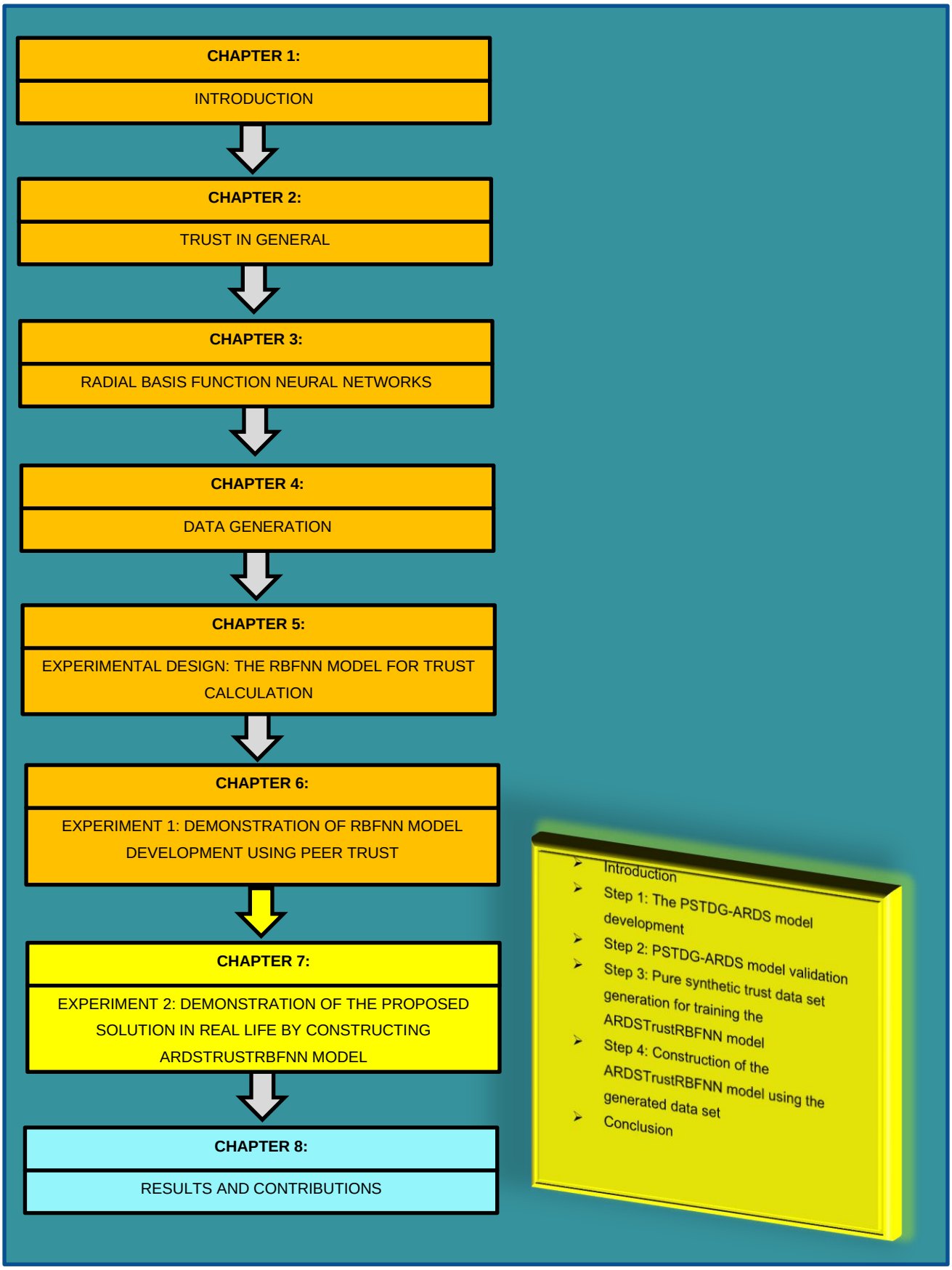


Figure 6-15: Signpost diagram of Chapter 7

CHAPTER 7 EXPERIMENT 2: DEMONSTRATION OF THE PROPOSED SOLUTION IN REAL LIFE BY CONSTRUCTING THE ARDSTRUSTRBFNN MODEL

7.1 Introduction

A four-step experimental design process was described in Chapter 5 to achieve the main aim of this study, namely to find an alternative trust calculation method using radial basis function neural networks (RBFNN). In Chapter 6, an RBFNN trust model called the PeerTrustRBFNN model, using the PeerTrust model by Li and Ling (2004) was developed using the four-step experimental design process described in Chapter 5. The PeerTrust model by Li and Ling (2004) is purely a theoretical trust calculation model which was studied in Section 2.4.6. The focus during the development of the RBFNN trust model, called the PeerTrustRBFNN, was mainly on the validation of the trust data generation model, called the PSTDG-PeerTrust model. This was because of the fact that the PeerTrust model by Li and Ling (2004) is purely a theoretical trust calculation model and Li and Ling provided the trust calculation equation. The data generation model that was created using the provided equation does not necessarily provide a valid trust data set, as errors can occur in program design, data management, and trust calculation. Hence, the focus was mainly on Step 2, the validation step of the four-step experimental design process described in Figure 5-1.

In this chapter, the demonstration of the proposed solution of generating a pure synthetic trust data set in any context and development of an RBFNN model to calculate trust will be done by using the Amazon Relational Database Service (ARDS) (Amazon, 2019) as an example. The context will be restricted to availability and durability for the demonstration. To build a model for the ARDS using an RBFNN that can determine trust values, the same four steps will be followed in this chapter as given in Figure 7-1. The pure synthetic trust data set generation model developed for the ARDS will be called the PSTDG-ARDS model. The RBFNN model for the ARDS developed for the calculation of trust values will be called the ARDSTrustRBFNN model.

In this chapter, the focus will be mainly on Step 1 and Step 2 of the four-step experimental design process described in Figure 5-1. This is because there is no algorithm or equations available to do a trust calculation for the ARDS (Amazon, 2019). It is necessary to show how the expert knowledge can be obtained, and how trust values can be calculated as a part of Step 1 of the four-step experimental design process described in Figure 5-1. After the successful completion of Step 1 of the four-step experimental design process, the developed data generation model needs to be validated in Step 2 of the four-step experimental design process

described in Figure 5-1 to make sure that the model can satisfy the identified expert knowledge. The following are the steps that will be followed in this chapter as given in Figure 7-1:

Step 1: A PSTDG model, called the PSTDG-ARDS model, will be developed, using the first four processes of the PSTDG Framework discussed in Section 4.7. This will be done in Section 7.2.
Step 2: The PSTDG-ARDS model will be validated in Section 7.3, using Definition 9 described in Section 4.6.1. This will be detailed in Section 7.3.
Step 3: In Section 7.4 how a pure synthetic trust data set can be generated will be described, using the validated PSTDG-ARDS model.
Step 4: The proposed RBFNN model called the ARDSTrustRBFNN for trust calculation will be developed using the generated data set in Section 7.5.

The chapter will be concluded in Section 7.6.

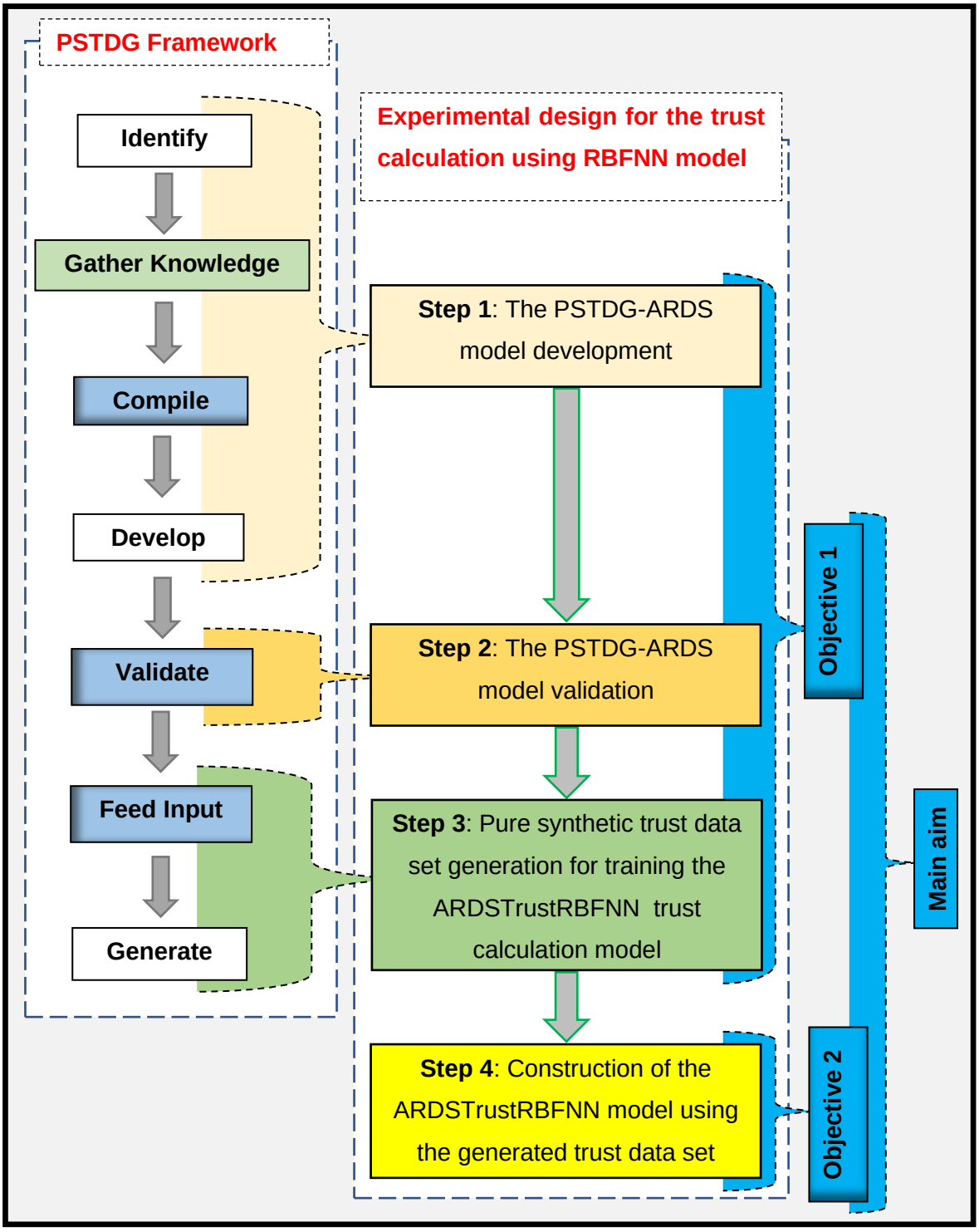


Figure 7-1: Experimental design for the construction of ARDSTrustRBFNN

7.2 Step 1: The PSTDG-ARDS model development

The first four parts of the PSTDG Framework which is given in Section 4.7, explained at the beginning of Section 5.2 and also demonstrated in Section 6.2 will be used to develop the data generation model called the PSTDG-ARDS.

7.2.1 Identify the need for synthetic data generation (SDG)

In Section 5.2.1, the need for SDG is explained.

7.2.2 Gather expert knowledge

In this process, all knowledge, including qualitative expert knowledge regarding trust data, will be obtained, as described in Section 5.2.2 and demonstrated in Section 6.2.2. According to the **New Definition** for Trust used in this study which is given in Section 2.2, trust needs to be calculated using information from sources, based on some standards or guidelines, resulting in some desired properties for the trust values. The properties of trust values depend on the data used, or alternatively the standards or guidelines. According to Section 5.2.2, in order to determine the expert knowledge, all the input sources, the features offered by an entity, the critical factors that can be used for evaluating the trust value, any standards that can be used to evaluate trust value and the relationship between critical factors or input sources and the output trust value needs to be determined. In the example in Chapter 6, an algorithm to do this was available. However, in the case of the ARDS, there is no algorithm available to do a trust calculation. For this reason, the service level agreement (SLA), the published features and standards of the ARDS given in Section 2.6 will be examined to gather the expert knowledge relating to the contexts of availability and durability.

As described in Section 2.6, the ARDS (Amazon, 2019) has its features and standards in the specific context, as well a commitment to an SLA. All the features offered by the ARDS is given in Section 2.6. Since the context of this experiment is restricted to availability and durability as mentioned in Section 7.1, Automated Backups, Database Snapshots, Multi-AZ Deployments, and Automatic Host Replacement are the features offered by the ARDS under the availability and durability context. The ARDS also provides an SLA in terms of Monthly Uptime Percentage (MUP). This MUP can be used as a standard to evaluate the trust value. The standards are described in terms of the MUP of the parameters or features. To calculate trust within the context of availability and durability, one must consider the SLA and MUP or percentage availability of all the features offered by the ARDS under the feature availability and durability.

After examining the SLA, MUP or percentage availability of the ARDS, the features offered by the ARDS under the context of availability and durability, traditional trust calculating algorithms described in Section 2.4, and findings in Sections 2.5.3, the following parameters can be distilled, and this can be used for the calculation of trust in the availability and durability context of the ARDS:

- The number of features offered in the context of availability and durability

- Percentage availability or percentage success of Automated Backups in terms of satisfaction in MUP
- Percentage availability or percentage success of Database Snapshots in terms of satisfaction in MUP
- Percentage availability or percentage success of Multi-AZ Deployments in terms of satisfaction in MUP
- Percentage availability or percentage success of Automatic Host Replacement in terms of satisfaction in MUP
- The number of transactions that took place so far
- Credit Return Success or Failure of the transaction
- The number of Credit returns failed so far
- Trust values that are achieved due to previous transactions or performance

As shown in Table 2-1, the ARDS offers the MUP not less than 99.95% (Amazon, 2019) in its latest SLA. In the case of the ARDS, the standards are defined in terms of MUP. As the ARDS offers the MUP of >= 99.95%, there is a 100% credit back if the MUP is < 95% and a 25% credit back if the MUP is < 99% and greater than or equal to 95%, and a 10% credit back if percentage availability is < 99.95% and greater than or equal to 99%. The trust value must go down when the MUP or the percentage availability goes down. There must be a reduction in trust value if there is a failure in the offered credit return. The amount of the reduction of trust value must increase with every credit return failure. The trust value must go down when the percentage availability goes down.

In summary, the ARDS offers an MUP of not less than 99.95% and a commitment of offering a credit return if a failure happens in the offered MUP. If the ARDS's MUP went lower and failed to give the offered credit return, then the trust value for the ARDS must go down. The following information, distilled from the above observations, can be considered as the expert knowledge regarding the ARDS trust value calculation in the context of availability and durability.

Expert knowledge 1. The new trust value of the ARDS in the context of availability and durability must depend upon the success of the MUP. The new trust value should be directly proportional to the MUP. The trust value must decrease when the percentage availability decreases.

Expert knowledge 2. The trust value should be calculated using all of the MUP or the percentage availability or percentage success of all the features received from all peers or customers.

Expert knowledge 3. The trust value of the ARDS in the context of availability and durability must depend upon the success or failure of credit return. The amount of the reduction of trust value must increase with every credit return failure.

The above-identified expert knowledge will be used for compiling the constraints for the PSTDG-ARDS model development which will be explained in Section 7.2.3 and Section 7.2.4 and the validation thereof in Section 7.3.

7.2.3 Compile constraints

The ARDS's trust value can be calculated differently in different contexts. It was identified that no published model calculated the trust values for the ARDS. Using the expert knowledge identified in Section 7.2.2, several equations must be developed that can be used in the trust calculations. A user can decide how to calculate trust, depending upon the needs or situations. Any one of the following two cases can be implemented while developing the model to generate trust data sets:

7.2.3.1 Case 1: Basic Trust Matrix

One way of calculating the trust value for the ARDS is to simply take the average of each customer's satisfaction in MUP received. This can be denoted as the availability context factor. The trust value for the ARDS can be obtained by the following equation which can be called the basic trust matrix:

$$T(ARDS) = \sum_{i=1}^{I(p)} S(P, i), \tag{7-1}$$

where $S(P, i)$ is the satisfaction (or peer's feedback or customer's feedback) regarding MUP between 0 and 1 given by each customer, $T(ARDS)$ is the trust value of the ARDS and $I(p)$ is the total number of peers' feedback (customers' feedback) during the recent time window. $S(P, i)$ can be calculated in different ways:

- Single value satisfaction (or peer's feedback or customer's feedback) regarding the MUP.
- The weighted average of satisfaction (or peer's feedback or customer's feedback) regarding the MUP for each feature.

In the case of single value satisfaction, one satisfaction value between 0 and 1 will be obtained from each peer or customer. Under the feature availability and durability, the ARDS offers four parameters or sub-features as mentioned in Section 7.2.2. In the case of weighted average satisfaction, $S(P,i)$ can be obtained by

$$S(P,i) = \gamma_1 \cdot S(p_1,i) + \gamma_2 \cdot S(p_2,i) + \gamma_3 \cdot S(p_3,i) + \gamma_4 \cdot S(p_4,i), \quad (7\text{-}2)$$

where $\gamma_1, \dots \gamma_4$ are the weight factors for the availability of the four features and $S(p_1,i), \dots S(p_4,i)$ is the satisfaction (or peer's feedback or customer's feedback) regarding the MUP between 0 and 1 given by each customer for each parameter or sub-features mentioned in Section 7.2.2.

Therefore, Equation (7-1) becomes

$$T(ARDS) = \sum_{i=1}^{I(p)} \gamma_1 \cdot S(p_1,i) + \gamma_2 \cdot S(p_2,i) + \gamma_3 \cdot S(p_3,i) + \gamma_4 \cdot S(p_4,i). \quad (7\text{-}3)$$

In this case, the trust value of the ARDS is calculated using Expert knowledge 1 and Expert knowledge 2.

7.2.3.2 Case 2: Basic Trust Matrix with credit return success context factor

In Case 2, the credit return success context factor will be incorporated into the basic trust matrix (availability context factor). The weight factors α and β can also be incorporated into Equation (7-1). Thus Equation (7-1) becomes

$$T(ARDS) = \alpha \cdot \textstyle\sum_{i=1}^{I(p)} S(P,i) - \beta \cdot CRF(ARDS), \quad (7\text{-}4)$$

where $CRF(ARDS)$ is the credit return success context factor, which is a deduction, α is the weight factor for the availability context factor and β is the weight factor for the deduction factor for the credit return success context factor. The credit return success context factor will be calculated as the ratio of the total number of credit failures that happened during the given

period over the total number of credit returns required during the given period. Hence, $CRF(ARDS)$ becomes

$$CRF(ARDS) = \frac{I(c_f)}{I(c_{rr})}, \tag{7-5}$$

where $I(c_f)$ is the total number of credit failures that happened during the given period and $I(c_{rr})$ is the total number of credit returns required during the given period.

After incorporating the satisfaction (or peer's feedback or customer's feedback) regarding the MUP as the weighted average of satisfaction and the credit return success context factor as the ratio of the total number of credit failures that happened during the given period over the total number of credit returns required during the given period, the trust value of the ARDS becomes

$$T(ARDS) = \alpha \cdot \sum_{i=1}^{I(p)} \gamma_1 \cdot S(p_1, i) + \gamma_2 \cdot S(p_2, i) + \gamma_3 \cdot S(p_3, i) + \gamma_4 \cdot S(p_4, i) - \beta \frac{I(c_f)}{I(c_{rr})}. \tag{7-6}$$

To incorporate the availability context factor and the credit return success context factor into the basic trust matrix, the weight factors α and β must be tuned in such a way to control the reputation level that can be reduced if there is a failure in credit return from the ARDS. A suitable choice for the weight factors α and β must be carefully made by the user to indicate how much the credit return success context factor must contribute to trust value as different choices will deliver different trust values. In this study, the weight factors α and β will be set to the values 1 and 0.25 respectively in Case 2.

It is again necessary for the user to decide how much each feature must contribute to the availability context factor of the trust value, as different choices will deliver different trust values. In this demonstration γ_1, γ_2, γ_3 and, γ_4 will be set to the same value, namely 0.25 to give equal importance to all four features of the ARDS in the context of availability and durability.

In this case, the trust value of the ARDS is calculated using Expert knowledge 1, Expert knowledge 2, and Expert knowledge 3.

7.2.3.3 Constraint selection for the PSTDG-ARDS model

It is the user's choice to decide which of the above two cases must be used to calculate trust, as each case gives a different trust value. In this study, the Basic Trust Matrix equation with the availability context factor and credit return success context factor (Case 2) will be used to calculate trust. This is because in Case 2, the trust value of the ARDS is calculated using Expert knowledge 1, Expert knowledge 2 and Expert knowledge 3 while in Case 1, the trust value of ARDS is calculated using Expert knowledge 1 and Expert knowledge 2 only.

The PSTDG-ARDS model development will be explained in the next section.

7.2.4 The PSTDG-ARDS model development

As selected in the previous section, the PSTDG-ARDS model will be developed by applying Case 2 which was described in Section 7.2.3.2 to generate pure synthetic trust data sets. The experiment will be performed by two stored procedures as follows:

MS SQL code is used to write the two stored procedures.

Table 7-1 shows the inputs and outputs of the first procedure.

Inputs	Outputs
• *MonthCount* (the total number of months that participated in trust calculation) • *MaxNumberOfPeersPerMonth* (the maximum number of peers (or customers) that submitted feedback per month) • *TrustContext* (the context at which trust needs to be calculated) • *FeatureCount* (the number of features offered by the entity under the given context) • *PercentageUptime* (MUP offered by the entity)	• Given number of months will be created with names and the number of peers or customer count will be assigned for each month.

Table 7-1: First stored procedure

The *FeatureCount, PercentageUptime, MonthNames* and their corresponding values will be stored in the SQL database. The details of the first stored procedure are given in Figure 7-2.

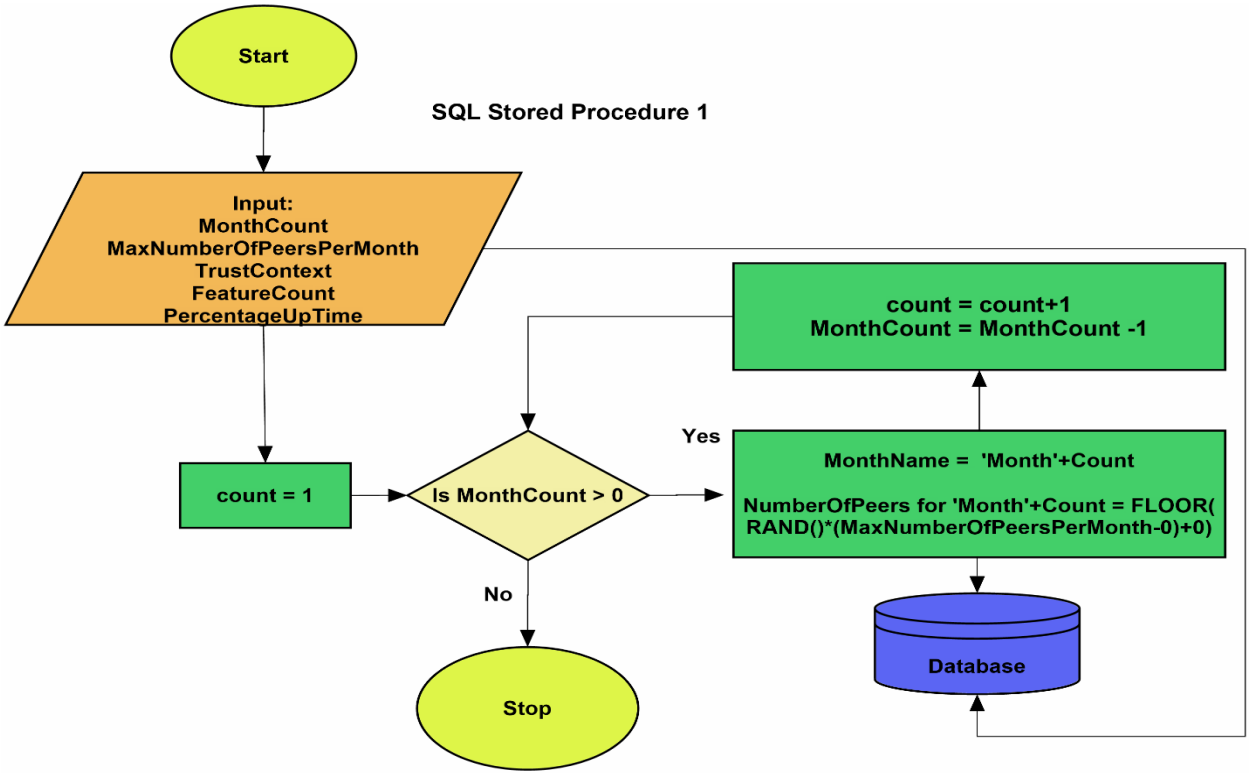


Figure 7-2: Stored Procedure 1 data flow

Table 7-2 shows the inputs and outputs of the second stored procedure.

Inputs	Outputs
• *Gamma1* (weight factor of the first feature) • *Gamma2* (weight factor of the second feature) • *Gamma3* (weight factor of the third feature) • *Gamma4* (weight factor of the fourth feature) • *Alpha* (weight factor of the availability context factor) • *Beta* (weight factor of the credit return success context factor)	• The given number of feedback will be submitted per month. • The ARDS's trust value will be updated according to the feedback received, credit return success or failure, availability context factor, the weight factor of each feature and credit return success context factor.

Table 7-2: Second stored procedure

The details of the second stored procedure are given in Figure 7-3 and Figure 7-4.

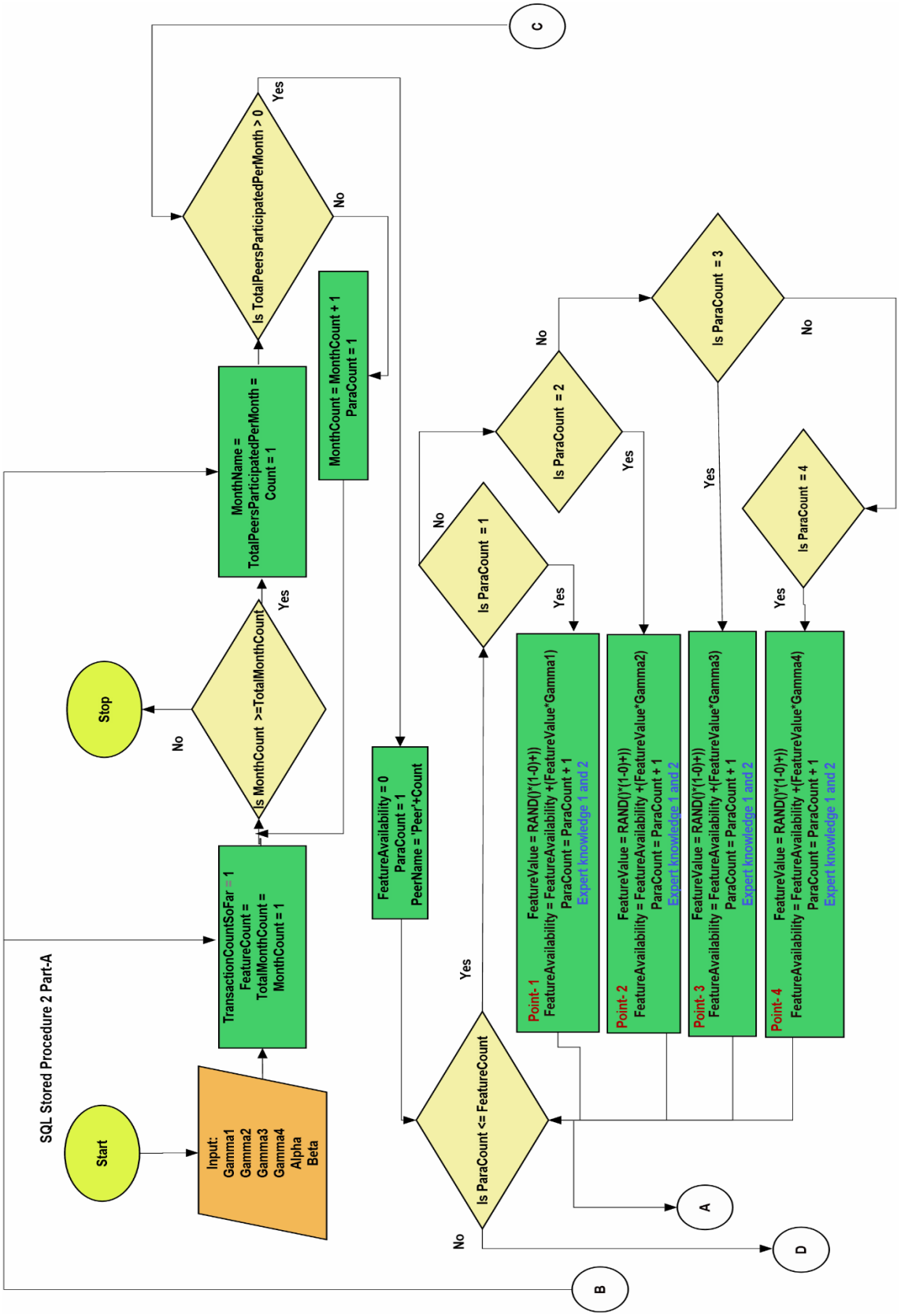


Figure 7-3: Stored Procedure 2 - Part-A data flow

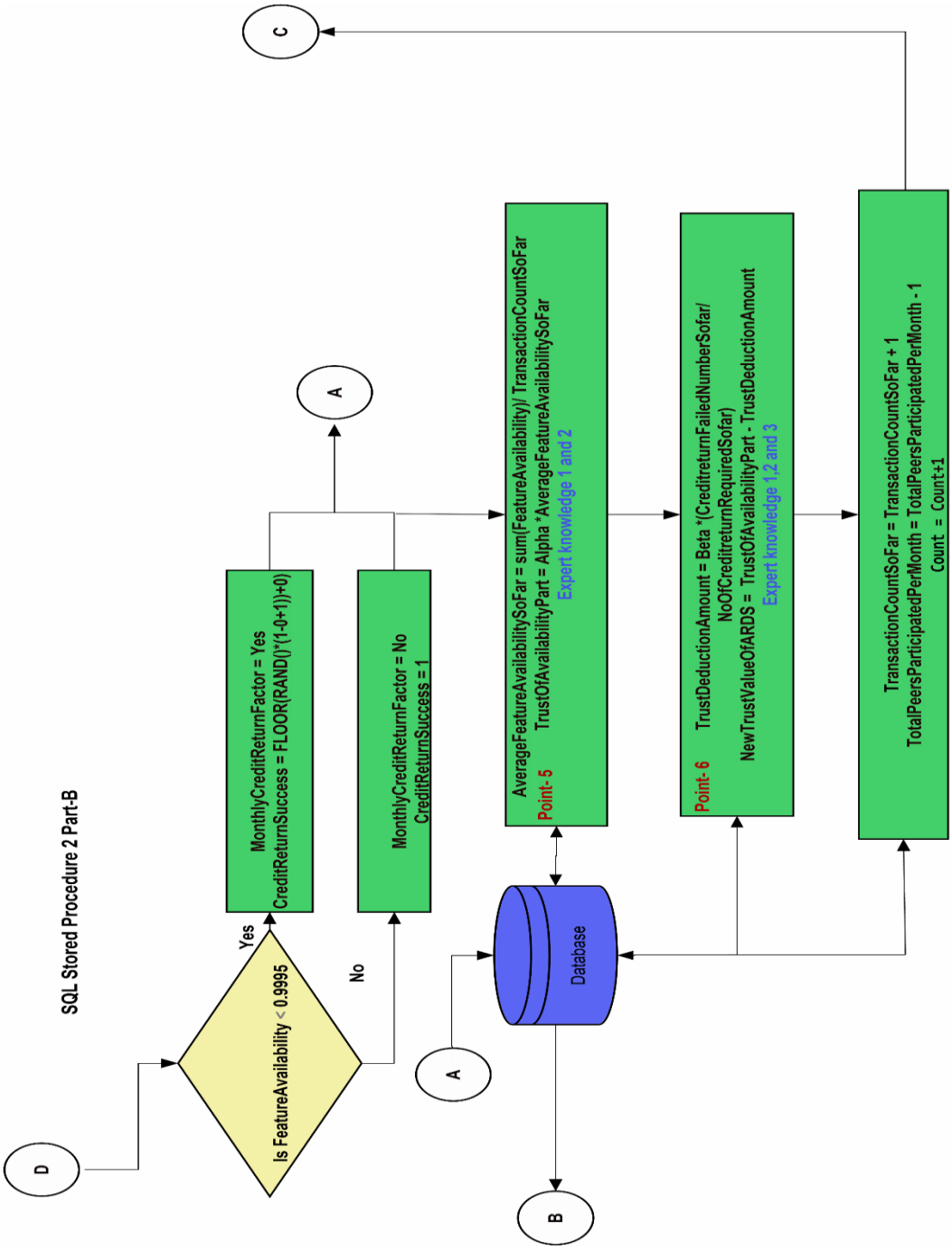


Figure 7-4: Stored Procedure 2 - Part-B data flow

As shown in Figure 7-2, the first stored procedure will accept the following input values:

- The total number of months that should be taking part in the trust calculation process
- The maximum number of peers (or customers) that can take part in the trust calculation process per month
- Context of trust that should be used for the calculation of trust value
- The total number of features or sub-features offered by the ARDS under the specific context

- The percentage uptime offered by the ARDS

The stored procedure will then create the given number of months with names, such as Month1 and Month2, etc., assign the number of peers (or customers) to each month, and will save all the given input values together with the month name and number of peers of each month into the database.

Figure 7-3 and Figure 7-4 provide the details of the second stored procedure as Part-A and Part-B respectively. The second stored procedure will accept the following input values:

- The weight factor for the first feature
- The weight factor for the second feature
- The weight factor for the third feature
- The weight factor for the fourth feature
- The weight factor for the availability context factor
- The weight factor for the credit return success context factor

The stored procedure will perform the given number of feedback submissions from the decided number of peers (or customers) per month. After each submission, a new trust value will be calculated for the ARDS.

At Point-5 in Figure 7-4, Expert knowledge 1 and Expert knowledge 2 which is described in Section 7.2.2 is applied into the PSTDG-ARDS model as the availability context factor.

Expert knowledge 3 which is described in Section 7.2.2 is applied at Point-6 in Figure 7-4 as the credit return success context factor. In the PSTDG-ARDS model, the credit return success context factor will be calculated as the ratio of the total number of credit failures that happened over the total number of credit returns required during that period as described in Section 7.2.3.2.

According to the basic trust matrix which is described in Section 7.2.3.1, the trust value for the ARDS will be the weighted average of each peer's (or customer's) satisfaction or percentage availability received for each feature. This is applied in the PSTDG-ARDS model at Point-1, -2, -3 and -4 in Figure 7-3 and Point-5 in Figure 7-4 as Expert knowledge 1 and Expert knowledge 2.

After each feedback submission, new trust values will be calculated according to the feedback received or percentage availability received for each feature (Expert knowledge 1), the total number of feedback submissions done by the peer (or customer) (Expert knowledge 1 and

Expert knowledge 2), the availability context factor (Expert knowledge 1), and the credit return success context factor (Expert knowledge 3) using the equation described in Section 7.2.3.2.

More details on this equation are given in Section 7.2.3.2. This is denoted as Point-6 in Figure 7-4 where all the expert knowledge identified in Section 7.2.2 is incorporated to calculate the new trust value for the ARDS. Case 2, which is described in Section 7.2.3.2, is implemented as the PSTDG-ARDS model. The new trust value for ARDS will be updated after receiving each feedback. The PSTDG-ARDS model is developed, using the PSTDG Framework's first four steps.

The PSTDG-ARDS model's validation will be explained in the next section.

7.3 Step 2: PSTDG-ARDS model validation

The PSTDG-ARDS model is developed using the Case 2 constraint as explained in Section 7.2.3.2. The model must consider all expert knowledge mentioned in Section 7.2.2 when calculating the trust value for the ARDS. A valid PSTDG-ARDS model should be able to generate a trust data set that can satisfy all the expert knowledge identified in Section 7.2.2.

The validation of the developed PSTDG-ARDS model can be done by completing the same steps explained in Section 4.6.2, mentioned in Section 5.3 and demonstrated in Section 6.3. The steps mentioned in Section 5.3 will be applied to the PSTDG-ARDS model for determining its validity.

All the What-if Scenarios will be compiled using the expert knowledge identified in Section 7.2.2. The following are the What-if Scenarios compiled, using the expert knowledge identified in Section 7.2.2.

- What-if Scenario 1 (Expert knowledge 1): What is the effect on the trust value of the ARDS if the satisfaction (feedback regarding the percentage availability of features) decreases?
- What-if Scenario 2 (Expert knowledge 1): What will happen to the trust value of the ARDS when the satisfaction (feedback regarding the percentage availability of features) received from peers (customers) increases linearly from a minimum value (0) to a maximum value (0.999) in a series of transactions?
- What-if Scenario 3 (Expert knowledge 2): What will happen to the trust value of the ARDS when the ARDS receives feedback from other peers (or customers), but the trust value calculation does not include all previous feedback from all customers or peers?

- What-if Scenario 4 (Expert knowledge 2): What will happen to the trust value of the ARDS when the ARDS receives feedback from other peers (or customers), but the trust value calculation does not include feedback regarding all features or sub-features?
- What-if Scenario 5 (Expert knowledge 3): What will happen to the trust value of the ARDS when the ARDS fails to give credit return during a set of transactions, keeping all other parameters constant in a series of transactions?

The full details of the validation of the PSTDG-ARDS model similar to Section 6.3 is given in Appendix B and shows that the developed PSTDG-ARDS model can be used to generate a valid trust data set for creating the ARDSTrustRBFNN model.

Pure synthetic trust data set generation using the PSTDG-ARDS model to train the ARDSTrustRBFNN model will be discussed in the next section.

7.4 Step 3: Pure synthetic trust data set generation for training the ARDSTrustRBFNN model

The PSTDG-ARDS model, which is developed in Section 7.2.4, is validated in Section 7.3. In this section, the pure synthetic ARDS trust data set will be generated using this validated PSTDG-ARDS model. The pure synthetic ARDS trust data set will be generated using the last two processes of the PSTDG Framework explained in Section 4.7. Before generating the data set, a set of input parameter values that needs to be used for the data set generation will be selected as described in Sections 4.7 and 5.4.

In Section 7.2.3.3, it was decided that the Case 2 constraint explained in Section 7.2.3.2, will be implemented in the PSTDG-ARDS model for the demonstration purpose. It was also explained in Section 7.2.3.2 that the user must take a suitable decision on how much each feature of the ARDS in the context of availability and durability must contribute to trust value. It is also mentioned that the user must decide on how much the availability context factor and the credit return success context factor must contribute to the trust value as different choices will deliver different trust values. A suitable choice for the parameter values must be made to get the desired outputs. In order to keep the maximum trust value as 1, the weight factor α must be set to 1. To give equal importance to each feature of the ARDS in the context of availability and durability, weight factors γ_1, γ_2, γ_3, and γ_4 must have the same value. To keep the maximum trust value as 1, weight factors γ_1, γ_2, γ_3, and γ_4 can each have a value of 0.25. The weight factor of the credit return success context factor β can have any value as mentioned in Section 7.2.3.2.

The following two stored procedures which are described in Section 7.2.4 will be used to generate the data set:

1. *2021ARDSGenerateMonths* which is given in Appendix A.5
2. *2021ARDSDoSingleMonthlyuptimeCreditbackTransactionsRandom* which is given in Appendix A.6

As explained in Section 5.4 and shown in Figure 7-1, pure synthetic trust data generation using the developed model consists of the feed input process and the generate process. In this demonstration, the following values will be set for the parameters in the first stored procedure:

- *MonthCount*: 50
- *MaxNumberOfPeersPerMonth*: 500
- *TrustContext*: Availability and Durability
- *FeatureCount*: 4
- *PercentageUptime*: 99.95

The following values will be given for the parameters in the second stored procedure to achieve Case 2 constraint described in Section 7.2.3.2 and selected in Section 7.2.3.3 for this experiment:

- *Gamma1:* 0.25
- *Gamma2:* 0.25
- *Gamma3:* 0.25
- *Gamma4:* 0.25
- *Alpha:* 1
- *Beta:* 0.25

TrustValueOfARDS is the only prediction class in this demonstration. After executing the above-mentioned two stored procedures, 13 727 samples were generated for the data set. As mentioned in Section 6.4, at least 1 000 samples need to be collected for a single prediction class. Consequently, 13 727 samples satisfy the rule-of-thumb of having the minimum requirement.

After executing the above-mentioned two stored procedures, 50 months will be created, and the trust value for the ARDS will be calculated after each peer's feedback submission. A data set with the following information will be generated:

1. *TransactionCountSoFar* - An integer that indicates the number of transactions that happened so far.
2. *MonthName* - Name of the month, for example, *Month1, Month2*, etc.
3. *PeerName* - Name of the peer, for example, *Peer1, Peer2*, etc.
4. *AvailabilityFeature1* - A value between 0 and 1 indicates the amount of satisfaction (percentage availability or percentage success regarding the first feature) received from peers or customers.
5. *AvailabilityFeature2* - A value between 0 and 1 indicates the amount of satisfaction (percentage availability or percentage success regarding the second feature) received from peers or customers.
6. *AvailabilityFeature3* - A value between 0 and 1 indicates the amount of satisfaction (percentage availability or percentage success regarding the third feature) received from peers or customers.
7. *AvailabilityFeature4* - A value between 0 and 1 indicates the amount of satisfaction (percentage availability or percentage success regarding the fourth feature) received from peers or customers.
8. *FeatureAvailability* - A value between 0 and 1 indicates the amount of satisfaction (percentage availability or percentage success regarding all the four features) calculated using the weight factors γ_1, γ_2, γ_3, and γ_4.
9. *MonthlyCreditReturnFactor* - Indicates whether the peer or customer is eligible for credit return from the ARDS. The peer or customer is eligible if the *FeatureAvailability* is less than the *PercentageUptime* (0.9995) offered by the ARDS. A 1 indicates yes, and a 0 means no.
10. *CreditReturnSuccess* – Indicates whether the ARDS succeeded in credit return to eligible peers or customers. A 1 indicates yes, and a 0 means no.
11. *CreditReturnFailedNumberSofar* - An integer that indicates the total number of failures that happened in credit return so far.
12. *NoOfCreditreturnRequiredSofar* - An integer that indicates the total number of credit returns required so far.
13. *TrustValueOfARDS* - A value that lies between 0 and 1 indicates the new trust value of the *ARDS* after the transaction or submission of monthly feedback from each peer or customer.

The construction of the ARDSTrustRBFNN model, using the generated data set, will be discussed in the next section.

7.5 Step 4: Construction of the ARDSTrustRBFNN model using the generated data set

This demonstration aims to develop an alternative trust calculation method using an RBFNN for the ARDS in the context of availability and durability. As mentioned in Section 5.5, to build an accurate RBFNN model that can determine trust values, suitable model hyperparameters must be chosen to train the model. As was done in Section 6.5, this will be done by training several candidate ARDSTrustRBFNN models, using uniform randomly sampled hyperparameters and selecting the best ARDSTrustRBFNN model. The ARDSTrustRBFNN model for trust calculation will be built on the generated trust data set. This process will also include the following three steps: data pre-processing, identifying the best ARDSTrustRBFNN model, and evaluating the best model as described in Section 5.5.

7.5.1 Data pre-processing

The data set obtained in Section 7.4 needs to be pre-processed before constructing the ARDSTrustRBFNN model since the generated inputs vary across different ranges. This can improve the accuracy of the ARDSTrustRBFNN model. It can decrease the computational cost, accelerate the learning process and also the pre-processing of the input variables helps to better match the predicted output.

From the generated data set unwanted fields must be deleted. Since the data should lie in the same range for the ANN to treat the values equally, the following inputs needs be normalised to a range of [0, 1] as these have data ranges that vary across a wide interval:

- *AvailabilityFeature1*
- *AvailabilityFeature2*
- *AvailabilityFeature3*
- *AvailabilityFeature4*

To do the above-required data pre-processing, the *2021ARDSTrust_Do_BinaryNorm* stored procedure written using MS SQL code will be used. This procedure can be found in Appendix A.7. The trust data set obtained in Section 7.4 will be the input for the above-mentioned stored procedure, and the output will be a data set having normalised values. This will include only the following fields:

- *AvailabilityFeature1*
- *AvailabilityFeature2*
- *AvailabilityFeature3*
- *AvailabilityFeature4*

- *MonthlyCreditReturnFactor*
- *CreditReturnSuccess*
- *TrustValueOfARDS*

After pre-processing was performed, identification of the best ARDSTrustRBFNN model will be discussed in the next section.

7.5.2 Identifying the best ARDSTrustRBFNN model for trust calculation

In this section, experiments will be done to identify the best model hyperparameters for the ARDSTrustRBFNN model. The program utilised for this purpose is written in the Python programming language with the TensorFlow library and Keras application program interface (API) used for RBFNN training, as performed in Section 6.5.2. The program code is given in Appendix A.8. Before the search for the best hyperparameters is performed, the following hyperparameter search space is defined:

Hyperparameter	Description
beta_min: 0.	The RBFNN beta value
beta_max: 2.	
hidden_nodes_min: 1.	The number of hidden RBFNN nodes
hidden_nodes_max: 200.	
window_size_min: 1.	The training set window size
window_size_max: 15.	
learning_rate: $10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5},$ or 10^{-6}.	The RBFNN learning rate
hours_to_run: 250.	Maximum time allowed for finding the best hyperparameters
epochs_value: 128.	The number of epochs per trial

Table 7-3: Hyperparameter search space for ARDSTrustRBFNN

The following search algorithm will be used to determine the best hyperparameters:

Algorithm:	Identifying the best ARDSTrustRBFNN model for trust calculation.
Input:	Hyperparameter search space specified in Table 7-3, generated trust data set and time limit.
Output:	Hyperparameter values for the best RBFNN model: *NumberOfHiddenNodes*, *WindowSize*, *betas_value*, *learning_rate*, *epochs_value* and MSE error on the test set.

1. While (Time limit has not passed) do
2. Obtain random values for the *betas_value*, *WindowSize*, *NumberOfHiddenNodes* and *learning_rate* hyperparameter variables using the ranges specified in Table 7-3.
3. Create a data set called *ARDSTrust2021* by converting the generated trust data set into input windows of *WindowSize* sizes.
4. Randomly sample a training set (70%), validation set (20%) and test set (10%) from the *ARDSTrust2021* data set.
5. Train an ARDSTrustRBFNN model using the hyperparameters sampled in Step 2 and determine the MSE error on the validation set.
6. Record the randomly sampled hyperparameter values, and the MSE error on the validation set.
7. End while
8. Evaluate the best ARDSTrustRBFNN model found by Steps 1 to 7 (in terms of the MSE error on the validation set) on the test set.

Algorithm 2: Identifying the best ARDSTrustRBFNN model for ARDS's trust calculation

In the next section, the evaluation of the best RBFNN trust model found will be described.

7.5.3 Evaluation of the best ARDSTrustRBFNN model for trust calculation

In the allowed time (250 hours run on an Intel® Core™ i7-8550U CPU at 1.80GHz laptop with 8.00 GB RAM), 4 531 candidate models were explored. The best RBFNN model found by Algorithm 2 has the following hyperparameter values:

- *NumberOfHiddenNodes*: 126;
- *WindowSize*: 14;
- *betas_value*: 0.770550472430907;
- *learning_rate*: 0.0001; and
- *epochs_value*: 128.

The MSE on the validation set has a value of $1.06869638329953 \times 10^{-5}$. The MSE on the test set is 7.718475×10^{-6}. As mentioned in Section 5.5.3, the MSE value closer to zero shows better model performance.

The development of the PSTDG-ARDS model solved the problem of not having a large data set for the training of an RBFNN model to calculate trust values for the ARDS. The validity of the pure synthetic trust data set was the next identified problem which was solved by validating the PSTDG-ARDS model. The validation of the PSTDG-ARDS model is done by What-if analysis and sensitivity analysis, using the new definition which was given for the validation of the pure synthetic trust data set in Section 4.6.1. The trust data set for the training of the ARDSTrustRBFNN model is generated, using the validated PSTDG-ARDS model. Research **Objective 1** is achieved. An RBFNN model, called the ARDSTrustRBFNN, was constructed for the calculation of trust values for the ARDS, using the generated data set. The best model was constructed, validated, and tested. Research **Objective 2** was achieved. The main aim, namely "An alternative trust calculation method using RBFNN" was therefore achieved.

7.6 Conclusion

In this chapter, a data generation model called the PSTDG-ARDS model was developed, using the ARDS (Amazon, 2019). This was also carried out, based on the PSTDG Framework described in Section 4.7. The PSTDG-ARDS model was validated, and a large trust data set was then generated, using the validated PSTDG-ARDS model. Finally, the construction of the RBFNN trust model, called the ARDSTrustRBFNN, was performed. The demonstration of the proposed solution to solve a real-life problem was carried out by using the ARDS (Amazon, 2019) as an example. The summary of the study, the results obtained, and the contributions made by this study and the summary of the most significant contributions will be discussed in Chapter 8.

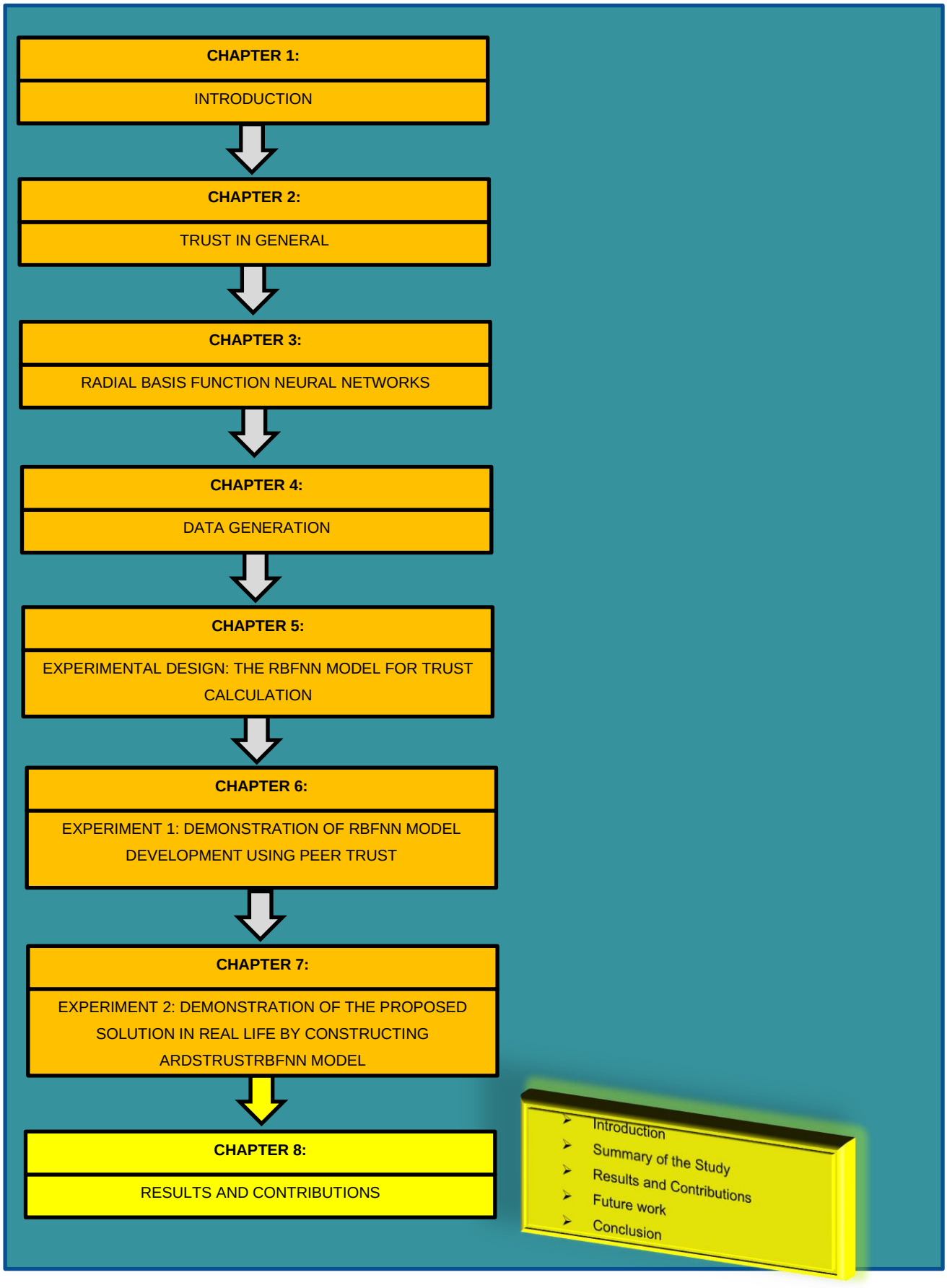


Figure 7-5: Signpost diagram of Chapter 8

CHAPTER 8 RESULTS AND CONTRIBUTIONS

8.1 Introduction

In Chapter 1, the problems in establishing trust within the electronic environment were introduced.

In Chapter 2, the problems associated with trust calculations were detailed. In the field of information technology and computer science, no widely accepted definition for the term *trust* could be found. Hence, the first contribution in Chapter 2 was a new definition for the term trust which states that trust comprises three dimensions, namely the dimension of context, the dimension of calculation or quantification of trust and the dimension of information sources on which the calculation is based. Moreover, trust is non-symmetric, non-transitive and context dependent. The trust value will be different in a different context. In Chapter 2, it was also identified that there is a need for an alternative method for quantifying trust values in any context. Another insight provided in Chapter 2 was that when the complexity of the electronic entity networks increases or when the context complexity increases, the complexity of the calculation of trust value will increase. The trust value quantification and management are time-consuming due to trust data storage and access. An alternative trust calculation method needed to be proposed to solve the identified problems.

In Chapter 3, it was shown that an RBFNN can solve complex pattern classification problems and the calculation of trust value is a non-separable linear problem. Hence, an RBFNN could be used in trust value classification.

This lead to the statement in Section 1.2 of the research question: ***"How can an RBFNN be used to calculate trust values between electronic entities?"***.

Training an RBFNN to calculate trust values between electronic entities in randomly varying situations requires accurate large data sets, as discussed in Section 4.2. There is a scarcity of trust data due to privacy issues, data storage problems due to the need for a centralised system to manage trust data, and network problems due to the storage and access of large data sets for trust calculation. According to the **new definition** of trust as formulated in Section 2.2, the trust value can be calculated by accessing or using information from different sources, where the information sources are the third dimension of trust. A possible solution to the scarcity of training data is the generation of pure synthetic trust data.

To address the research question, the proposed solution and the objectives of this study are discussed in the next section.

To answer the research question, an alternative trust calculation method, using an RBFNN would be developed which was the main aim of this study. To achieve the main aim, two objectives needed to be achieved. As given in Section 1.1.2, a possible solution to the scarcity of training data was to generate authentic synthetic data. Hence, the first objective of this research was to develop a process to generate accurate large data sets for the training of the RBFNN trust model which was done in Chapter 4. The first objective is given below:

Objective 1- Data Generation: Provide a generic framework for valid pure synthetic trust data generation.

To achieve **Objective 1**, the Pure Synthetic Trust Data Generation (PSTDG) Framework was developed which was discussed in Section 4.7. Using this developed PSTDG Framework, the PSTDG model was developed to generate a trust data set. The validation of the PSTDG model was also carried out. Thereafter trust data sets were generated, using the validated PSTDG model.

After achieving **Objective 1** and **Objective 2** the answer to the research question, namely “How can an RBFNN be used to calculate trust values between electronic entities?” was provided. The second objective of this research is given below:

Objective 2- Construction of an RBFNN trust model: Provide a suitable theoretical experimental design process for developing an RBFNN-based trust calculation model.

To achieve **Objective 2**, a theoretical experimental design process for the development of an RBFNN model-based trust calculation model was provided in Chapter 5. This developed theoretical experimental design process was demonstrated using two experiments; firstly, by means of a theoretical trust calculation model and secondly, by means of a real-world problem. The first experiment was done using an available model, namely the PeerTrust model by Li and Ling (2004) which is a purely theoretical trust calculation model. The second experiment was to show that the proposed solution could be applied to real-life problems. The second experiment was performed using the Amazon Relational Database Service (ARDS) (Amazon, 2019). The demonstration of the first experiment was explained in Chapter 6.

In the first experiment, a data generation model called the PSTDG-PeerTrust model was developed, it was validated, a large trust data set was generated and the RBFNN trust model called the PeerTrustRBFNN was trained using this data set. In this experiment, the focus during the development of the RBFNN trust model called the PeerTrustRBFNN was mainly on the validation of the trust data generation model called the PSTDG-PeerTrust model. This was because the PeerTrust model by Li and Ling (2004) is purely a theoretical trust calculation

model and Li and Ling provided the trust calculation equation. Even though the trust calculation equation was available, the developed data generation model did not necessarily provide a valid trust data set.

In the second experiment, to show that the proposed solution could be applied to real-life problems, the pure synthetic trust data set generation model called the PSTDG-ARDS model was developed, using the Amazon Relational Database Service (ARDS) (Amazon, 2019), it was validated, a large trust data set was generated and the RBFNN trust model, called the ARDSTrustRBFNN, was developed, using the generated trust data set. In this experiment, the focus was mainly on the PSTDG-ARDS model development and the PSTDG-ARDS model validation. This was because there was no algorithm or equations available to do a trust calculation for the ARDS (Amazon, 2019). It was necessary to show how expert knowledge could be obtained, and how trust values could be calculated. The PSTDG-ARDS model validation was to make sure that the model could satisfy the identified expert knowledge. The demonstration of the second experiment was performed in Chapter 7

The aim of Chapter 8 is, firstly, to summarise and reflect on the research as described in the study and, secondly, to provide the reader with the results and contributions of this research. The remainder of this chapter is organised as follows. In Section 8.2, a summary of each chapter will be presented. The results and contributions of this research and its summary will be explained in Section 8.3. Possible future work will be discussed in Section 8.4. Finally, the chapter will be concluded in Section 8.5.

8.2 Summary of the Study

The layout and the structure of the study were presented in Section 1.5. The focus of each chapter is now briefly revisited.

8.2.1 Chapter 1

In Chapter 1, the background of this research was presented and the purpose of this research was described. The main research question was compiled and asked:

"How can an RBFNN be used to calculate trust values between electronic entities?"

The research argued that trust is context-dependent, and it can be applied in various contexts which were detailed in Chapter 2. Trust values will be different in a different context. It was also mentioned that the trust context of information security is mostly studied and the influence of the specific context on trust will be the first problem to be investigated which was also carried out in Chapter 2. The dimension of calculation or quantification of trust is the second dimension of

trust. It was also argued that traditional algorithms seem to be very complex which was identified in Chapter 2 and therefore, models that enable the calculation of trust between software entities in a rapidly changing environment with an easy and efficient method, irrespective of the trust context, need to be investigated. The dimension of information sources that one needs to do the calculation of trust was also discussed. Traditional algorithms use different methods for storing the data and accessing the data of previous transactions. It was mentioned that the data sources are usually not readily available, which leads to the third area of investigation.

The power of artificial neural networks (ANN), especially the advantages of an RBFNN which was identified in Chapter 3, made the argument that a model which is based on an RBFNN approach can reduce the problem of calculation complexities in the quantification of trust. Training an RBFNN to calculate trust values requires accurate large data sets. Insufficient trust data sets could be found due to privacy issues, data storage problems due to the need for a centralised system to manage trust data, and network problems due to the storage and access of large data sets for trust calculation. A possible solution to the scarcity of training data was to generate authentic synthetic data.

The first objective of this research was to develop a process to generate accurate large data sets for the training of the RBFNN trust model which was done in Chapter 4. To achieve the main aim, the following two objectives were derived:

The first objective is given below:

Objective 1- Data Generation: Provide a generic framework for valid pure synthetic trust data generation.

The second objective is given below:

Objective 2- Construction of RBFNN trust model: Provide a suitable theoretical experimental design process for developing an RBFNN-based trust calculation model.

In Chapter 1, the research methodology and study overview were also discussed.

8.2.2 Chapter 2

In Chapter 2, the literature on trust and the theoretical background regarding trust calculation were provided. The three dimensions of trust were identified, and a **new definition** of trust was given in Section 2.2. The three dimensions of trust were also discussed in Section 2.3.1. Several examples of trust calculations were studied and the influence of the three dimensions

and context in trust calculation were identified. The standards of the several products available in the market were identified and the features offered by the Amazon Relational Database Service (ARDS) were also gathered to develop the model ARDSTrustRBFNN as envisaged in **Objective 2**. The contribution of Chapter 2 to achieve the main aim will be discussed in Section 8.3.

8.2.3 Chapter 3

In Chapter 3, artificial neural network (ANN) history, the architecture of the RBFNN, the advantages and disadvantages of RBFNNs, non-separable linearity of trust value, the training of the RBFNN and the time series prediction using an RBFNN were discussed. The contribution of Chapter 3 to answer the **main research question** will be discussed in Section 8.3.

8.2.4 Chapter 4

The literature on problems in trust data collection, synthetic data, and a generic approach to synthetic data generation was reviewed in Chapter 4. A generic approach to synthetic data generation (SDG) was discussed in Section 4.4. The validity of the synthetic trust data set was discussed in Section 4.5 and a **new definition** for synthetic trust data validation was given in Section 4.6.1. A method for validating the generated pure synthetic trust data set was developed in Section 4.6.2 to achieve **Objective 1**. In Section 4.7, a framework named the PSTDG Framework for generating a valid pure synthetic trust data set was also developed for the successful achievement of **Objective 1**. How the research in Chapter 4 contributed to achieving **Objective 1** will be included in Section 8.3.

8.2.5 Chapter 5

In Chapter 5, a generic explanation of a four-step experimental design process as shown in Figure 5-1 to achieve the main aim of this study to build a model using an RBFNN that can determine trust values between electronic entities was advanced. A theoretical description of a process to achieve **Objective 1** and **Objective 2** was provided. The crux of this research is described in this chapter. The contents of this chapter are regarded as the highlight of this study.

8.2.6 Chapter 6

In Chapter 6, the first experiment to demonstrate how to build an RBFNN trust model called the PeerTrustRBFNN model, using the PeerTrust model by Li and Ling (2004) which is a theoretical model was implemented. As described in Section 8.1, the focus was mainly on the validation of the developed PSTDG model (**Objective 1**).

The PSTDG model was developed as a part of achieving **Objective 1** in Section 6.2 for the trust data set generation using the PeerTrust model by Li and Ling (2004). The validation of the developed PSTDG model (**Objective 1**) and pure synthetic trust data set generation using the validated model (**Objective 1**) for training the RBFNN trust calculation model was discussed in Section 6.3 and Section 6.4. In Section 6.5, the details regarding the construction of the RBFNN model using the generated data set were given. The experiments to identify the best RBFNN model for trust calculation using the PeerTrust model by Li and Ling (2004) (**Objective 2**) and the development, training, validation, and evaluation for the identified best model using the generated data set (**Objective 2**) were also included in Section 6.5. The contribution of Chapter 6 to achieve the main aim will be discussed in Section 8.3.

8.2.7 Chapter 7

In Chapter 7, the second experiment to demonstrate the proposed solution of generating a pure synthetic trust data set and the development of an RBFNN model to calculate trust for a real-world problem was presented. The context was restricted to availability and durability. This demonstration was done using the ARDS (Amazon, 2019) as an example. In this experiment, the focus was mainly on PSTDG model (**Objective 1**) development and PSTDG model (**Objective 1**) validation as described in Section 8.1.

The PSTDG model for the ARDS was developed as part of achieving **Objective 1** in Section 7.2 for the trust data set generation using the ARDS (Amazon, 2019). The validation of the developed PSTDG model (**Objective 1**) and pure synthetic trust data set generation using the validated model (**Objective 1**) for training the RBFNN trust calculation model were discussed in Section 7.3 and Section 7.4. In Section 7.5, the details regarding the construction of the RBFNN model using the generated data set were given. The experiments to identify the best RBFNN model for trust calculation using the ARDS (Amazon, 2019) (**Objective 2**) and the development, training, validation, and evaluation for the identified best model using the generated data set (**Objective 2**) were also included in Section 7.5. The contribution of Chapter 7 to achieve the main aim of the research will be discussed in Section 8.3.

The results obtained in this study and how the two research objectives, **Objective 1** and **Objective 2** that are mentioned at the beginning of this chapter were accomplished during this study to achieve the main aim will be discussed in the next section.

8.3 Results and Contributions

In Section 1.3 and at the beginning of this chapter it was given that the main aim of this research was to establish the following: "*How can an RBFNN be used to calculate trust values*

between electronic entities?" To achieve the main aim, it was identified that the two research objectives **Objective 1** and **Objective 2** that are mentioned at the beginning of this chapter need to be achieved.

The proposal of developing an alternative trust calculation method, using an RBFNN, as well as the results obtained, are discussed below:

The first focus point in Chapter 2 was to analyse and define the term trust within the electronic environment. In Section 2.2, many definitions of trust were identified. In many identified definitions of trust, it was defined within the context in which it was applied. No definition could be found for the term trust within the electronic environment that could be used in any context. Hence, as a first step, by identifying the commonalities and differences between several definitions, three dimensions of trust were identified, namely trust context, calculations for the quantification of trust, and information sources. Then, based on the three dimensions, a **new definition** for the term trust in an electronic environment that could be used in any context, was developed in Section 2.2. This was the first contribution towards the main aim.

The following observations relevant to this study which can be grouped under the identified dimensions of trust were also made in Sections 2.4, 2.5.3, and 2.6, contributing to the fact that there was a need to find an alternative way of calculating trust in an electronic environment in any context:

Trust context:

- Trust in the information security context is well documented because in most of the trust calculations, the focus is strongly on the information security context and information sources to calculate trust value are readily available.

Calculations for the quantification of trust:

- When the complexity of the networks grows the calculation becomes more complex.
- The calculation complexity increases as the context complexity increases.
- The algorithm also gets complex when the number of parameters used for the quantification of trust increases.

Information sources for the quantification of trust:

- To calculate trust, information can be obtained from different sources.
- When the complexity of the network grows, the number of parameters bounding the information needs to be increased to increase the confidence in the information.

- Each algorithm uses different models to store and manage data while calculating trust values.
- Each electronic entity has its features and standards in the specific context.

The above observations lead to the fact that there is a need for a trust calculation method that works in any context, and with less calculation complexity than the available trust calculation methods. This leads to finding an alternative way of calculating trust in an electronic environment in any context. In Chapter 2, the trust calculation, the challenges in calculating trust in an electronic environment, and different trust calculating algorithms were studied. Therefore, Step 1, Step 2, and Step 3 identified in Section 1.3 were completed.

In Chapter 3, the main aim was to study the RBFNN as a possible solution to calculate trust. The neural network architecture identified in Section 3.3.2 led to a solution for the research question, "How can an RBFNN be used to calculate trust values between electronic entities?" More specifically, a summary of the properties of the RBFNN is as follows:

- The RBFNN is a feed-forward neural network with a relatively simple structure and has been widely used as a universal function approximator.
- The learning capability in an RBFNN is faster than in a multi-layer perceptron.
- An RBFNN has a better approximation capability compared to traditional neural networks.
- An RBFNN has good generalisation properties.
- Trust value classification may be done using an RBFNN, since an RBFNN can solve complex pattern classification problems.
- An RBFNN is good at pattern recognition.
- Compared to other neural networks, an RBFNN is much easier to design.
- An RBFNN has a high tolerance to input noise and online learning ability.
- The curse of dimensionality is one of the disadvantages of using an RBFNN. This is because the number of basis functions required will increase as the dimension of the input space increases.
- The runtime speed of an RBFNN can be low due to the large number of hidden units needed in many problems.
- An RBFNN's strong tolerance to the input noise increases the designed system's stability.
- The RBFNN will also help to reduce the problem of storing and accessing information of all the previous transactions.

In Section 3.3.3, it was identified that the calculation of trust value could be considered as a non-separable linear problem and the ability of an RBFNN to convert a non-separable linear problem to a linearly separable problem could be used for trust calculation. This insight shed some light on the main aim of the research that was to determine "*How can an RBFNN be used to calculate trust values between electronic entities?"*

As described in Section 3.2.4, 3.3.3, and 3.3.5, by using an RBFNN one could reduce the problem of calculation complexities in the calculation of trust. Calculating trust using an RBFNN could answer the research question "How can an RBFNN be used to calculate trust values between electronic entities?" Therefore, in Chapter 3, the RBFNN was studied and Step 4 identified in Section 1.3 was completed.

In Chapter 4, an investigation of synthetic data generation was undertaken. All the problems in trust data collection were identified in Section 4.2. The observations identified in Section 4.2 motivated one of the two objectives of this research, '**Data generation**' mentioned in Section 1.3. The following observations were identified in Section 4.2:

- Training of an RBFNN to calculate trust values between electronic entities in randomly varying situations requires large data sets.
- In traditional algorithms, there must be a method and an architecture for accessing, managing, and calculating the trust values.
- Access to stored data can affect the speed of the trusted network and the trust calculation time.
- Collecting data to calculate trust values is time-consuming and costly.
- Real-world data are scarce due to privacy concerns.

The investigation regarding the challenges in using an RBFNN for trust calculation was done and Step 4 identified in Section 1.3 was completed. The identified fact was that a large trust data set was not available for the training of the RBFNN model. The above observation regarding the challenges in using an RBFNN and trust data collection resulted in the identification of **Objective 1** given in Section 8.1 as stated below:

Objective 1- Data Generation: Provide a generic framework for valid pure synthetic trust data generation.

A generic four-step framework for synthetic data generation was identified in Section 4.4 which leads to finding a solution for **Objective 1**.

The following observations relevant to this study regarding **Objective 1** were made in Section 4.5:

- A major concern regarding the quality of a synthetic data set is realism.
- Inaccurate data can lead to the development of an inaccurate RBFNN model to predict trust.
- The data should be based on standards or guidelines to have some desired property within a specified context to quantify trust.

The above observations can be concluded, as a validated synthetic trust data set is necessary to train the RBFNN model, as inaccurate data can lead to the development of an inaccurate RNFNN model to predict trust. Observations that were identified in Section 4.5 motivated the need for obtaining a valid trust data set. The investigation regarding the challenges in generating a valid trust data set was carried out and Step 6 identified in Section 1.3 was completed.

In Section 4.6.1, many definitions of validation were identified. From many identified definitions of validation, it was identified that validation is defined, based on the needs of a user or a researcher. Validation should provide pieces of evidence to show that the model is sufficiently accurate for its intended use. Therefore, the focus would be on the intended use of the data set. The validation of the data generation model should be done by determining if the intended use was satisfied which resulted in the need for having a definition for the term validation in the context of generated trust data. The term validation in the context of generated trust data needed to be defined. Based on observations identified in Section 4.6.1, a **new definition** for the term validation in the context of generated trust data was developed in Section 4.6.1.

A three-step process was also developed in Section 4.6.2 to validate the data generation model according to the **new definition** for the term validation. The following was the developed three-step process that resulted in the achievement of **Objective 1**:

1) A set of What-if Scenarios should be compiled using the available expert knowledge regarding the trust calculations.
2) The trust data set should be generated with scatter plots for each scenario.
3) The plotted graphs should be analysed to determine if the visual representation of the change in the generated trust data matches the expert knowledge.

As described in Section 4.7, three new steps were included in the identified generic four-step framework for synthetic data generation to develop a seven-step framework for generating a valid pure synthetic trust data set as given in

. This newly developed seven-step framework called the PSTDG Framework given in

solved the problem of not having a valid synthetic trust data set. A framework for generating a valid pure synthetic trust data set was developed successfully for the accomplishment of **Objective 1** and a three-step method described in Section 4.6.2 could be used for the validation of the data generation model according to the **new definition** of validation. This also contributed to the successful achievement of **Objective 1**.

In order to achieve the main aim of this study to build a model using an RBFNN that could determine trust values between electronic entities, a generic explanation of a four-step experimental design process was described in Chapter 5, as shown in Figure 5-1. In this chapter, a theoretical description of a process to achieve **Objective 1** and **Objective 2** was provided. The four-step experimental design process described in Chapter 5 was the highlight of this study. Two experiments were carried out to demonstrate this four-step experimental design process, one in Chapter 6 and the other in Chapter 7.

In Chapter 6, the first experiment was used to show how to build an RBFNN trust model called the PeerTrustRBFNN model using the PeerTrust model by Li and Ling (2004) which is a theoretical model. This was done using the four-step experimental design process as shown in Figure 6-1. In this experiment, the focus was mainly on the validation of the developed PSTDG model (**Objective 1**) as mentioned in Section 6.1. In Section 6.2, the PSTDG model called the PSTDG-PeerTrust model was developed, using the first four steps of the PSTDG Framework using the PeerTrust model by Li and Ling (2004). This developed model was validated in Section 6.3, using the three-step method described in Section 4.6.2. Then, a pure synthetic trust data set for PeerTrust was generated in Section 6.4, using the validated **PSTDG-PeerTrust model**. This solved the problem of not having a large enough data set for the training of the RBFNN model. **Objective 1** '**Data Generation'** identified in Section 1.3 and given in Section 8.1 was therefore successfully achieved. An RBFNN model called the PeerTrustRBFNN was constructed for the calculation of trust, using an available model which is a purely theoretical trust calculation, namely the PeerTrust model by Li and Ling (2004), using the generated data set in Section 6.5. The best model was developed, tested, and validated. Thus, in this experiment, **Objective 2**, identified in Section 1.3 and given in Section 8.1 was successfully achieved. The four-step experimental design process given in Figure 5-1 could be used for the successful completion of the development of an RBFNN trust model, using a theoretical trust model.

Since the development of the RBFNN model called the PeerTrustRBFNN was completed using a purely theoretical trust calculation model (PeerTrust model by Li and Ling (2004)), the next

experiment was to show that the four-step experimental design process given in Figure 5-1 could be used to build a trust model for a real-world problem.

The second demonstration was done using the ARDS as indicated in Section 7.1. For demonstration, the context was restricted to availability and durability. As a first step, the PSTDG-ARDS model was developed for the generation of the trust data set for the ARDS in the context of availability and durability. As explained in Section 7.1, in this demonstration, the focus was mainly on Step 1 (pure synthetic trust data generation model development) and Step 2 (trust data generation model validation) of the four-step experimental design process described in Figure 5-1 and Figure 7-1. In Section 7.2.3, it was identified that no published model to calculate the trust values for the ARDS could be found. Various equations needed to be developed for the calculations of trust values. Using the published features and standards of the ARDS given in Section 2.6, three areas of expert knowledge were identified in Section 7.2.2. Using the expert knowledge identified in Section 7.2.2, several equations were developed in Section 7.2.3 which gave rise to a solution for not having a published model to calculate the trust values for the ARDS. The equations were developed for the following cases in Section 7.2.3.1 and Section 7.2.3.2:

Case 1: Basic Trust Matrix
Case 2: Basic Trust Matrix with credit return success context factor

The development of equations in Section 7.2.3.1 and Section 7.2.3.2 served as a contribution to the successful development of the PSTDG-ARDS model to achieve **Objective 1** for this demonstration. The development of the PSTDG-ARDS model was completed in Section 7.2.4 with the equation developed in Section 7.2.3.2 and using the first four steps of the PSTDG Framework which was developed in Section 4.7 and also as shown in the four-step experimental design process in Figure 7-1. This developed data generation model named the PSTDG-ARDS model was validated in Section 7.3, using the three-step method described in Section 4.6.2. A pure synthetic trust data set for the ARDS in the context of availability and durability was generated in Section 7.4, using the validated PSTDG-ARDS model. This solved the problem of not having enough data for the training of the ARDSTrustRBFNN model.

In Section 7.5, an ARDSTrustRBFNN model was constructed for the calculation of trust in the context of availability and durability using the data set generated in Section 7.4. The best model was developed, validated and tested, thereby successfully achieving **Objective 2** identified in Section 1.3 and given in Section 8.1. The four-step experimental design process shown in Figure 5-1 could be used to build an RBFNN trust model for a real-world problem. The four-step

experimental design process shown in Figure 5-1 could be utilised for the development of an RBFNN trust model to calculate trust values between electronic entities.

Next, a summary of the contributions of this study is presented.

8.3.1 Summary of Contributions

The **main** contributions of this study are summarised below:

8.3.1.1 Contribution 1

A seven-step framework called the PSTDG Framework to generate a validated trust data set in any context for the training of the RBFNN model was introduced in Section 4.7 as shown in Figure 4-2.

8.3.1.2 Contribution 2

A four-step theoretical experimental design process for the development of an RBFNN-based trust calculation model was introduced in Chapter 5 as shown in Figure 5-1. This was the highlight of this study, as it answered the main research question.

The following can also be considered as contributions:

8.3.1.3 Contribution 3

A new definition of trust was developed in Section 2.2 as follows:

“Trust is a quantified belief or a probability of belief of an entity, which can be calculated by accessing or using information from different sources which are based on some set of standards or guidelines, to have some desired property within a specified context.”

8.3.1.4 Contribution 4

A new definition for synthetic trust data set validation which was developed in Section 4.6.1 is as follows:

*“*Validation is a process or methods to provide evidence that can be used to determine whether the synthetic trust data generation model is generating a valid trust data set under its specific scenarios for its intended use.”

8.3.1.5 Contribution 5

A three-step process was developed in Section 4.6.2 to validate the trust data generation model according to the **new definition** for synthetic trust data set validation which was developed in Section 4.6.1.

8.3.1.6 Contribution 6

The PSTDG-PeerTrust model for the generation of the trust data set using the PeerTrust model by Li and Ling (2004) was developed and validated, using the first five steps of the PSTDG Framework in Section 6.2 and Section 6.3.

8.3.1.7 Contribution 7

The construction of the RBFNN model called the PeerTrustRBFNN for the PeerTrust model using the generated data set was completed in Section 6.5.

8.3.1.8 Contribution 8

The PSTDG-ARDS model for the generation of a trust data set for the ARDS in the context of availability and durability was completed in Section 7.2.4 with the equation developed in Section 7.2.3.2.

8.3.1.9 Contribution 9

The ARDSTrustRBFNN model for the calculation of trust in the context of availability and durability was constructed in Section 7.5, using the data set generated in Section 7.4.

“An alternative trust calculation method using radial basis function neural networks” is successfully achieved in this study.

8.3.1.10 Contribution 10

A peer-reviewed conference paper (Zacaria *et al.*, 2022) on this alternative method developed in this study, was accepted, published and presented at the SATNAC conference in Fancourt, George, Western Cape, South Africa in 2022 (Appendix C).

8.4 Future work

In this study, an alternative trust calculation method using an RBFNN was proposed. The RBFNN trust model is trained using a trust data set which was generated using a valid PSTDG model. There was no need for a precise analysis regarding the performance of the model

versus other models or the accuracy of the trust model, as this was an alternative trust model which gives validated results. There was no comparative study with other trust models. The comparative study could be a topic for further research. Other neural network architectures have been growing in popularity. Therefore, implementation of trust models using other types of neural network could be investigated to determine whether the trust model's performance could be improved. Finally, a neural network could be investigated to predict future trust values.

8.5 Conclusion

The contribution of all other chapters to the aim and objectives of this study was addressed in this final chapter. It provided a brief description of what was achieved throughout each chapter. The results and contributions of this research were discussed, and future research was recommended.

REFERENCE LIST

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., ... Zheng, X. 2016. *TensorFlow: A system for large-scale machine learning*. Paper presented at the Proceedings of the 12th USENIX conference on Operating Systems Design and Implementation, Savannah, GA, USA. https://www.usenix.org/system/files/conference/osdi16/osdi16-abadi.pdf Date of access: 25 Aug. 2021.

Abdul-Rahman, A. & Hailes, S. 2000. *Supporting trust in virtual communities*. Paper presented at the Proceedings of the 33rd Annual Hawaii International Conference on System Sciences, Maui, Hawaii, USA. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=926814 Date of access: 25 Aug. 2020.

Aberer, K. 2001. P-Grid: A self-organizing access structure for P2P information systems. In: Batini, C., Giunchiglia, F., Giorgini, P. & Mecella, M., eds. *Conference proceedings*. 9th International Conference on Cooperative Information Systems (CoopIS 2001), Trento, Italy. Berlin Heidelberg: Springer. pp. 179-194.

Aberer, K. & Despotovic, Z. 2001. *Managing trust in a peer-2-peer information system*. Paper presented at the Proceedings of the tenth international conference on Information and knowledge management (CIKM '01), Atlanta, Georgia, USA. https://dl.acm.org/doi/pdf/10.1145/502585.502638 Date of access: 15 Mar. 2020.

Abhari, A. & Soraya, M. 2010. Workload generation for YouTube. *Multimedia Tools and Applications*, 46(1):91-118. https://doi.org/10.1007/s11042-009-0309-5

Abiodun, O.I., Jantan, A., Omolara, A.E., Dada, K.V., Mohamed, N.A. & Arshad, H. 2018. State-of-the-art in artificial neural network applications: A survey. *Heliyon*, 4(11):e00938. https://doi.org/10.1016/j.heliyon.2018.e00938

Aboulnaga, A., Naughton, J.F. & Zhang, C. 2001. *Generating Synthetic Complex-Structured XML Data*. Paper presented at the Proceedings of the Fourth International Workshop on the Web and Databases, (WebDB 2001) in conjunction with ACM PODS/SIGMOD, Santa Barbara, California, USA. https://www.csd.uoc.gr/~hy561/Data/Papers/xmldatagen-webdb01.pdf Date of access: 14 Aug. 2020.

Agatonovic-Kustrin, S. & Beresford, R. 2000. Basic concepts of artificial neural network (ANN) modeling and its application in pharmaceutical research. *Journal of Pharmaceutical and Biomedical Analysis*, 22(5):717-727.

http://www.sciencedirect.com/science/article/pii/S0731708599002721
https://doi.org/10.1016/S0731-7085(99)00272-1

Agostini, A. & Moro, G. 2004. Identification of Communities of Peers by Trust and Reputation. In: Bussler, C. & Fensel, D., eds. *Conference Proceedings*. 11th International Conference of Artificial Intelligence: Methodology, Systems, and Applications (AIMSA 2004), Varna, Bulgaria. Berlin Heidelberg: Springer pp. 85-95.

Al-Shargabi, B. 2016. *Security Engineering for E-Government Web Services: A Trust Model*. Paper presented at the International Conference on Information Systems Engineering (ICISE 2016), Los Angeles, CA, USA 20-22 Apr. https://ieeexplore.ieee.org/document/7486205 Date of access: 15 Mar. 2018.

Almenárez, F., Marín, A., Campo, C. & Garcia, C. 2004. *PTM: A pervasive trust management model for dynamic open environments*. Paper presented at the The First Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services Boston, USA. 12 Jan. 2017.

Alshamrani, A. & Bahattab, A. 2015. A comparison between three SDLC models waterfall model, spiral model, and Incremental/Iterative model. *International Journal of Computer Science Issues (IJCSI)*, 12(1):106-111.

Alwosheel, A., Van Cranenburgh, S. & Chorus, C.G. 2018. Is your dataset big enough? Sample size requirements when using artificial neural networks for discrete choice analysis. *Journal of Choice Modelling*, 28:167-182. https://doi.org/10.1016/j.jocm.2018.07.002

Amazon. 2018a. *Amazon RDS Service Level Agreement*. https://aws.amazon.com/rds/sla/ Date of access: 23 Jun. 2018.

Amazon. 2018b. *Amazon S3 Service Level Agreement*. https://aws.amazon.com/s3/sla/ Date of access: 23 Jun. 2018.

Amazon. 2019. *Amazon RDS Service Level Agreement.* https://aws.amazon.com/rds/sla/ Date of access: 23 Jun. 2020.

Anderson, J.W., Kennedy, K.E., Ngo, L.B., Luckow, A. & Apon, A.W. 2014. *Synthetic data generation for the internet of things*. Paper presented at the IEEE International Conference on Big Data (Big Data 2014), Washington, DC, USA, 27-30 Oct. 2014. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7004228 Date of access: 24 Jun. 2020.

Au, R., Looi, M. & Ashley, P. 2001. *Automated cross-organisational trust establishment on extranets*. Paper presented at the Proceedings Workshop on Information Technology for Virtual Enterprises. ITVE 2001, Gold Coast, Queensland, Australia, 29-30 Jan. 2001. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=904483 Date of access: 20 Mar. 2018.

Awchi, T.A. 2008. Application of radial basis function neural networks for reference evapotranspiration prediction. *Al-Rafidain Engineering Journal (AREJ)*, 16(1):117-130. https://doi.org/10.33899/rengj.2008.44029

Ayala-Rivera, V., Portillo, O., Murphy, L. & Thorpe, C. 2016. COCOA: A Synthetic Data Generator for Testing Anonymization Techniques. In: Domingo-Ferrer., J. & Pejić-Bach., M., eds. *Conference proceedings*. International Conference on Privacy in Statistical Databases (PSD 2016), Dubrovnik, Croatia. Switzerland: Springer, Cham. pp. 163-177.

Aziz, K.A.A. & Abdullah, S.S. 2009. *Face Detection Using Radial Basis Functions Neural Networks With Fixed Spread*. Paper presented at the The Second International Conference on Control, Instrumentation and Mechatronic Engineering (CIM09) Malacca, Malaysia. https://arxiv.org/abs/1410.2173 Date of access: 18 Mar. 2020.

Azmi, R., Hakimi, M. & Bahmani, Z. 2011. Dynamic Reputation Based Trust Management Using Neural Network Approach. *International Journal of Computer Science Issues (IJCSI)*, 8(5):161-165.

Azure, M. 2008. *Service Level Agreements*. https://azure.microsoft.com/en-us/support/legal/sla/ Date of access: 19 Sep. 2018.

Bahga, A. & Madisetti, V.K. 2011. Synthetic workload generation for cloud computing applications. *Journal of Software Engineering and Applications*, 4(7):396-410. https://doi.org/10.4236/jsea.2011.47046

Barse, E.L., Kvarnstrom, H. & Jonsson, E. 2003. *Synthesizing test data for fraud detection systems*. Paper presented at the 19th Annual Computer Security Applications Conference (ACSAC 2003). Las Vegas, NV, USA. https://dl.acm.org/doi/10.5555/956415.956464 Date of access: 16 Mar. 2019.

Basheer, I.A. & Hajmeer, M. 2000. Artificial neural networks: fundamentals, computing, design, and application. *Journal of Microbiological Methods*, 43(1):3-31. https://doi.org/10.1016/S0167-7012(00)00201-3

Baskaran, R., Arunachalam, S., Manjunath, K. & Kubendran, T. 2008. Artificial Neural Networks for the Prediction of Thermo Physical Properties of Liquid Mixtures. *Computer and Information Science*, 1(3):3-12.

Ben-Gan, I. 2012. *Microsoft SQL Server 2012 High-Performance T-SQL Using Window Functions*. 1st ed. 1005 Gravenstein Highway NorthSebastopol, California: O'Reilly Media, Inc.

Bergstra, J. & Bengio, Y. 2012. Random search for hyper-parameter optimization. *The Journal of Machine Learning Research*, 13(2):281−305.

Bezerianos, A., Papadimitriou, S. & Alexopoulos, D. 1999. Radial basis function neural networks for the characterization of heart rate variability dynamics. *Artificial Intelligence in Medicine*, 15(3):215-234. http://www.sciencedirect.com/science/article/pii/S0933365798000554 https://doi.org/10.1016/S0933-3657(98)00055-4

Bhaumik, P., Sayeem Reaz, A., Murayama, D., Suzuki, K.-I., Yoshimoto, N., Kramer, G. & Mukherjee, B. 2015. IPTV over EPON: Synthetic traffic generation and performance evaluation. *Optical Switching and Networking*, 18(Part 2):180-190. http://www.sciencedirect.com/science/article/pii/S1573427714000411 https://doi.org/10.1016/j.osn.2014.05.007

Billings, S.A. & Zheng, G.L. 1995. Radial basis function network configuration using genetic algorithms. *Neural Networks*, 8(6):877-890. https://doi.org/10.1016/0893-6080(95)00029-Y

Botta, A., Dainotti, A. & Pescapé, A. 2012. A tool for the generation of realistic network workload for emerging networking scenarios. *Computer Networks*, 56(15):3531-3547. https://doi.org/10.1016/j.comnet.2012.02.019

Brockwell, P.J. & Davis, R.A. 2002. *Introduction to time series and forecasting*. 2. Verlag New York: Springer.

Broomhead, D. & Lowe, D. 1988. *Radial basis functions, multi-variable functional interpolation and adaptive networks*. (Royal Signals And Radar Establishment Malvern (United Kingdom) memorandum No, 4148). https://apps.dtic.mil/sti/pdfs/ADA196234.pdf Date of access: 11 Jun. 2020.

Carley, K.M. 1996. *Validating computational models*. (Center for the Computational Analysis of Social and Organizational Systems CASOS technical report). http://reports-archive.adm.cs.cmu.edu/anon/anon/home/ftp/usr0/ftp/isr2017/CMU-ISR-17-105.pdf Date of access: 11 Jan. 2020.

Catrambone, V., Greco, A., Vanello, N., Scilingo, E.P. & Valenza, G. 2019. Time-Resolved Directional Brain–Heart Interplay Measurement Through Synthetic Data Generation Models. *Annals of Biomedical Engineering*, 47(6):1479-1489. https://doi.org/10.1007/s10439-019-02251-y https://doi.org/10.1007/s10439-019-02251-y

Chan, Y., Correa, C.D. & Ma, K. 2010. *Flow-based scatterplots for sensitivity analysis*. Paper presented at the 2010 IEEE Symposium on Visual Analytics Science and Technology, Salt Lake City, UT, USA 25-26 Oct. 2010. https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=5652460 Date of access: 15 Aug. 2019.

Chen, S., Cowan, C.F.N. & Grant, P.M. 1991. Orthogonal least squares learning algorithm for radial basis function networks. *IEEE Transactions on Neural Networks*, 2(2):302-309. https://doi.org/10.1109/72.80341

Chien-Cheng, L., Pau-Choo, C., Jea-Rong, T. & Chein, I.C. 1999. Robust Radial Basis Function Neural Networks. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 29(6):674-685. https://doi.org/10.1109/3477.809023

Chollet, F. 2020. *Introduction to Keras for Researchers*. (Everything you need to know to use Keras & TensorFlow for deep learning research). https://keras.io/getting_started/intro_to_keras_for_researchers/ Date of access: 25 Aug. 2021.

Christopher Frey, H. & Patil, S.R. 2002. Identification and review of sensitivity analysis methods. *Risk analysis*, 22(3):553-578. https://doi.org/10.1111/0272-4332.00039

Commerce, B.E., Jøsang, A. & Ismail, R. 2002. *The beta reputation system*. Paper presented at the Proceedings of the 15th Bled Electronic Commerce Conference, Bled, Slovenia. https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.60.5461&rep=rep1&type=pdf Date of access: 20 Mar. 2021.

Cui, L., Zhang, Q., Yang, L. & Bai, C. 2021. A Performance Prediction Method Based on Sliding Window Grey Neural Network for Inertial Platform. *Remote Sensing*, 13(23), 4864. https://doi.org/10.3390/rs13234864

Das, T. & Teng, B.-S. 2004. The Risk-Based View of Trust: A Conceptual Framework *Journal of Business and Psychology*, 19(1):85-116. https://doi.org/10.1023/B:JOBU.0000040274.23551.1b

Dash, C.S.K., Behera, A.K., Dehuri, S. & Cho, S.-B. 2016. Radial basis function neural networks: a topical state-of-the-art survey. *Open Computer Science*, 6(1):33-63. https://doi.org/10.1515/comp-2016-0005

Daskapan, S., Nurtanti, I. & Berg, J.v.d. 2008. *Improving trust valuation for file sharing in P2P networks*. Paper presented at the 2008 IEEE International Conference on Systems, Man and Cybernetics, Singapore, 12-15 Oct. https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=4811595 Date of access: 12 Sep. 2018.

Delon, C., Serca, D., Boissard, C., Dupont, R., Dutot, A., Laville, P., ... Delmas, R. 2007. Soil NO emissions modelling using artificial neural network. *Tellus B: Chemical and Physical Meteorology*, 59(3):502-513. https://doi.org/10.1111/j.1600-0889.2007.00254.x

Dennett, S., Feinler, E.J. & Perillo, F. 1985. *Arpanet Information Brochure*. https://apps.dtic.mil/sti/pdfs/ADA164353.pdf Date of access: 11 Jun. 2019.

Denning, D. 1993. A New Paradigm for Trusted Systems. In: Michael., J.B., Ashby., V. & Meadows., C., eds. *Conference Proceedings*. Proceedings on the 1992-1993 workshop on New security paradigms (NSPW '92-93), Little Compton Rhode Island USA. New York, NY, United States: Association for Computing Machinery. pp. 36-41.

Drechsler, J., Bender, S. & Rässler, S. 2007. *Comparing Fully and Partially Synthetic Datasets for Statistical Disclosure Control in the German IAB Establishment Panel*. Paper presented at the EUNECE: Work session on statistical data confidentiality, Manchester, United Kingdom. https://nces.ed.gov/FCSM/pdf/2007FCSM_Drechsler-III-A.pdf Date of access: 12 Aug. 2021.

Dubey, A.D. 2015. *K-Means Based Radial Basis Function Neural Networks for Rainfall Prediction*. Paper presented at the 2015 International Conference on Trends in Automation, Communications and Computing Technology (I-TACT-15), Bangalore, India, 21-22 Dec. 2015. https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=7492664 Date of access: 23 Jan. 2021.

Duda, R.O., Hart, P.E. & Stork, D.G. 2000. *Pattern classification and scene analysis*. 2nd ed. New York: Wiley

E. Rumelhart, D., E. Hinton, G. & McClelland, J. 1986. A General Framework for Parallel Distributed Processing. In. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*. 1. Cambridge, Massachusetts: MIT Press. pp. 45-76.

Eluyode, O. & Akomolafe, D.T. 2013. Comparative study of biological and artificial neural networks. *European Journal of Applied Engineering and Scientific Research*, 2(1):36-46.

Elzwayie, A., El-Shafie, A., Yaseen, Z.M., Afan, H.A. & Allawi, M.F. 2017. RBFNN-based model for heavy metal prediction for different climatic and pollution conditions. *Neural Computing and Applications*, 28(8):1991-2003. https://doi.org/10.1007/s00521-015-2174-7

Er, M.J., Wu, S., Lu, J. & Toh, H.L. 2002. Face recognition with radial basis function (RBF) neural networks. *IEEE transactions on neural networks*, 13(3):697-710. http://dx.doi.org/10.1109/TNN.2002.1000134

Erol, R., Oğulata, S.N., Şahin, C. & Alparslan, Z.N. 2008. A Radial Basis Function Neural Network (RBFNN) Approach for Structural Classification of Thyroid Diseases. *Journal of Medical Systems*, 32(3):215-220. https://doi.org/10.1007/s10916-007-9125-5

Frank, R.J., Davey, N. & Hunt, S.P. 2000. *Input window size and neural network predictors*. Paper presented at the Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium, Como, Italy 27-27 July 2000. https://uhra.herts.ac.uk/bitstream/handle/2299/6719/900909.pdf?sequence=1 Date of access: 12 Jan. 2019.

Frank, R.J., Davey, N. & Hunt, S.P. 2001. Time Series Prediction and Neural Networks. *Journal of Intelligent and Robotic Systems*, 31(1):91-103. https://doi.org/10.1023/A:1012074215150 https://doi.org/10.1023/A:1012074215150

Gambetta, D. 1988. Can We Trust Trust? In. *Trust: Making and Breaking Cooperative Relations*. New York: Blackwell. pp. 213-237.

Ghinita, G., Tao, Y. & Kalnis, P. 2008. *On the Anonymization of Sparse High-Dimensional Data*. Paper presented at the 2008 IEEE 24th International Conference on Data Engineering, Cancun, Mexico, 7-12 April 2008. https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=4497480 Date of access: 18 Jan. 2019.

Goodfellow, I., Bengio, Y. & Courville, A. 2016. *Deep learning*. Cambridge, MA, USA: The MIT Press.

GoogleCloud. 2018a. *Google Cloud Storage SLA* https://cloud.google.com/storage/sla-20161020 Date of access: 20 Aug. 2018.

GoogleCloud. 2018b. *Google Cloud Datastore Service Level Agreement (SLA)* https://cloud.google.com/datastore/sla-20161004 Date of access: 2 Jan. 2017.

Govier, T. 1998. *Dilemmas of trust*. Canada: McGill-Queen's University Press-MQUP.

Grandison, T. & Sloman, M. 2002a. A survey of trust in internet applications. *IEEE Communications Surveys & Tutorials*, 3(4):2-16. http://dx.doi.org/10.1109/COMST.2000.5340804

Grandison, T. & Sloman, M. 2002b. Specifying and Analysing Trust for Internet Applications. In: Monteiro, J.L., Swatman, P.M.C. & Tavares, L.V., eds. 2nd IFIP Conference on e-Commerce, e-Business, e-Government, Lisbon, Portugal. Boston, MA: Springer pp. 145-157.

Gulli, A., Kapoor, A. & Pal, S. 2019. *Deep learning with TensorFlow 2 and Keras: regression, ConvNets, GANs, RNNs, NLP, and more with TensorFlow 2 and the Keras API*. Packt Publishing Ltd.

Hagan, M.T., Demuth, H.B., Beale, M.H. & Jess, O.D. 2014. *Neural Network Design*. 2nd ed. 0 Park Plaza Boston, MAUnited States: PWS Publishing Co.

Hang, C.W., Kalia, A.K. & Singh, M.P. 2012. *Behind the Curtain: Service Selection via Trust in Composite Services*. Paper presented at the 2012 IEEE 19th International Conference on Web Services, Honolulu, HI, USA https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6257784 Date of access: 19 Apr. 2018.

Haviluddin, H. & Tahyudin, I. 2015. Time Series Prediction Using Radial Basis Function Neural Network. *International Journal of Electrical and Computer Engineering (IJECE)* 5(4):765-771. http://doi.org/10.11591/ijece.v5i4.pp765-771

Hawala, S. 2008. *Producing partially synthetic data to avoid disclosure*. Paper presented at the Proceedings of the Joint Statistical Meetings Alexandria, VA: American Statistical Association. http://www.asasrms.org/Proceedings/y2008/Files/301018.pdf Date of access: 17 Aug. 2019.

Haykin, S. 1994. *Neural networks: a comprehensive foundation*. 2nd ed. Singapore: Pearson Education, Inc.

Hota, H., Handa, R. & Shrivas, A. 2017. Time Series Data Prediction Using Sliding Window Based RBF Neural Network. *International Journal of Computational Intelligence Research (IJCIR)*, 13(5):1145-1156.

Jadid, M.N. & Fairbairn, D.R. 1996. Neural-network applications in predicting moment-curvature parameters from experimental data. *Engineering Applications of Artificial Intelligence*, 9(3):309-319. http://www.sciencedirect.com/science/article/pii/0952197696000218 https://doi.org/10.1016/0952-1976(96)00021-8

Jain, A.K., Jianchang, M. & Mohiuddin, K.M. 1996. Artificial neural networks: A tutorial. *Computer*, 29(3):31-44. http://dx.doi.org/10.1109/2.485891

Jayawardena, A.W., Xu, P.C., Tsang, F.L. & Li, W.K. 2006. Determining the structure of a radial basis function network for prediction of nonlinear hydrological time series. *Hydrological Sciences Journal*, 51(1):21-44. https://doi.org/10.1623/hysj.51.1.21

Jian-min, H., Hui-qun, Y., Juan, Y. & Ting-ting, C. 2008. *A Complete (alpha,k)-Anonymity Model for Sensitive Values Individuation Preservation*. Paper presented at the 2008 International Symposium on Electronic Commerce and Security, Guangzhou, China, 3-5 Aug. 2008. https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=4606080 Date of access: 17 Sep. 2021.

Jin, H., Tu, X., Han, Z. & Liao, X. 2005. A Community-Based Trust Model for P2P Networks. In: Yang, L.T., Rana, O.F., Di Martino, B. & Dongarra, J., eds. *Conference proceedings*. International Conference on High Performance Computing and Communications (HPCC 2005), Sorrento, Italy,. Berlin Heidelberg: Springer pp. 419-428.

Jones, W.W., Peacock, R.D., Forney, G.P. & Reneke, P.A. 2004. *Verification and Validation of CFAST. A Model of Fire Growth and Smoke Spread*. (NISTIR 7080). https://nvlpubs.nist.gov/nistpubs/Legacy/IR/nistir7080.pdf Date of access: 25 Jan. 2021.

Kamvar, S.D., Schlosser, M.T. & Garcia-Molina, H. 2003. *The Eigentrust algorithm for reputation management in P2P networks*. Paper presented at the Proceedings of the 12th international conference on World Wide Web, Budapest, Hungary.

Kennedy, K., Delany, S.J. & Mac Namee, B. 2011. *A framework for generating data to simulate application scoring*. Paper presented at the Credit Scoring and Credit Control XII Conference, Edinburgh. https://arrow.tudublin.ie/cgi/viewcontent.cgi?article=1099&context=scschcomcon Date of access: 17 Jun. 2021.

Khazaei, M., Sadat-Hosseini, H., Marjaninejad, A. & Daneshvar, S. 2017. A Radial Basis Function Neural Network approximator with fast terminal sliding mode-based learning algorithm and its application in control systems. In. 2017 Iranian Conference on Electrical Engineering (ICEE), Tehran, Iran. pp. 812-816.

Kleijnen, J.P. 1998. *Validation of simulation, with and without real data*. (Tilburg University, Center for Economic Research, Discussion Paper Series No. 1998-22). https://ssrn.com/abstract=138010 Date of access: 10 Jun. 2019.

Kleijnen, J.P.C. 1997. Sensitivity analysis and related analyses: A review of some statistical techniques. *Journal of Statistical Computation and Simulation*, 57(1-4):111-142. https://doi.org/10.1080/00949659708811805

Koval, S.I. 2018. *Data preparation for neural network data analysis*. Paper presented at the 2018 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus), Moscow and St. Petersburg, Russia https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8317233 Date of access: 29 Jan. 2020.

Kushiro, K., Harada, Y. & Takeno, J. 2013. Robot uses emotions to detect and learn the unknown. *Biologically Inspired Cognitive Architectures*, 4:69-78. https://doi.org/10.1016/j.bica.2013.01.002

Kuźniar, K. & Zając, M. 2017. Some methods of pre-processing input data for neural networks. *Computer Assisted Methods in Engineering and Science*, 22(2):141-151. https://cames.ippt.pan.pl/index.php/cames/article/view/33/28

Law, A.M. 2019. *How to build valid and credible simulation models*. Paper presented at the 2019 Winter Simulation Conference (WSC), National Harbor, MD, USA. https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=9004789 Date of access: 18 Aug. 2021.

Li, W., Y. Ng, W.W., Wang, T., Pelillo, M. & Kwong, S. 2021. HELP: An LSTM-based approach to hyperparameter exploration in neural network learning. *Neurocomputing*, 442:161-172. https://www.sciencedirect.com/science/article/pii/S0925231221003337 https://doi.org/10.1016/j.neucom.2020.12.133

Li, X. & Ling, L. 2002. *Building trust in decentralized peer-to-peer electronic communities*. Paper presented at the Fifth International Conference on Electronic Commerce Research (ICECR-5), Montreal, Canada. https://www.cc.gatech.edu/projects/disl/PeerTrust/pub/xiong02building.pdf Date of access: 16 Jan. 2019.

Li, X. & Ling, L. 2004. PeerTrust: supporting reputation-based trust for peer-to-peer electronic communities. *IEEE Transactions on Knowledge and Data Engineering*, 16(7):843-857. https://doi.org/10.1109/TKDE.2004.1318566

Li, Y. & Wu, H. 2012. A Clustering Method Based on K-Means Algorithm. *Physics Procedia*, 25:1104-1109. https://doi.org/10.1016/j.phpro.2012.03.206

Lin, G.-F. & Wu, M.-C. 2011. An RBF network with a two-step learning algorithm for developing a reservoir inflow forecasting model. *Journal of Hydrology*, 405(3-4):439-450. http://www.sciencedirect.com/science/article/pii/S0022169411003702 https://doi.org/10.1016/j.jhydrol.2011.05.042

Lin, P.J., Samadi, B., Cipolone, A., Jeske, D.R., Cox, S., Rendon, C., ... Rui, X. 2006. *Development of a Synthetic Data Set Generator for Building and Testing Information Discovery Systems*. Paper presented at the Third International Conference on Information Technology: New Generations (ITNG'06), Las Vegas, NV, 10-12 April 2006. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1611688 Date of access: 23 Aug. 2019.

Ling, Z. & Bo, Z. 1999. A geometrical representation of McCulloch-Pitts neural model and its applications. *IEEE Transactions on Neural Networks*, 10(4):925-929. https://doi.org/10.1109/72.774263

Little, R. & Liu, F. 2003. *Selective multiple imputation of keys for statistical disclosure control in microdata*. https://biostats.bepress.com/cgi/viewcontent.cgi?article=1005&context=umichbiostat Date of access: 11 Jan. 2021.

Liu, R., Fang, B., Tang, Y.Y. & Chan, P.P.K. 2016. *Synthetic Data Generator for Classification Rules Learning*. Paper presented at the 2016 7th International Conference on Cloud Computing and Big Data (CCBD), Macau, China 16-18 Nov. 2016. https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=7979934 Date of access: 17 Mar. 2018.

M. Berg, A., Mol, S., Kismihók, G. & Sclater, N. 2016. The role of a reference synthetic data generator within the field of learning analytics. *Journal of Learning Analytics*, 3(1):107-128. http://dx.doi.org/10.18608/jla.2016.31.7

Maan, A.K., Jayadevi, D.A. & James, A.P. 2017. A Survey of Memristive Threshold Logic Circuits. *IEEE Transactions on Neural Networks and Learning Systems*, 28(8):1734-1746. https://doi.org/10.1109/TNNLS.2016.2547842

MacQueen, J. 1967. Some methods for classification and analysis of multivariate observations. In: Cam., L.M.L. & Neyman., J., eds. *Conference proceedings*. 5th Berkeley symposium on mathematical statistics and probability, Statistical Laboratory University of California. Berkeley and Los Angeles: University of California press. pp. 281-297.

Manna, A., Sengupta, A. & Mazumdar, C. 2016. A Survey of Trust Models for Enterprise Information Systems. *Procedia Computer Science*, 85:527-534.

http://www.sciencedirect.com/science/article/pii/S1877050916305609
https://doi.org/10.1016/j.procs.2016.05.212

McCullagh, J. & Bluff, K. 1993. *Genetic modification of a neural networks training data*. Paper presented at the Proceedings 1993 The First New Zealand International Two-Stream Conference on Artificial Neural Networks and Expert Systems, Dunedin, New Zealand https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=323006 Date of access: 12 aUG. 2019.

McCulloch, W.S. & Pitts, W. 1943. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115-133. https://doi.org/10.1007/BF02478259 https://doi.org/10.1007/BF02478259

McLachlan, S., Dube, K. & Gallagher, T. 2016. *Using the CareMap with Health Incidents Statistics for Generating the Realistic Synthetic Electronic Healthcare Record*. Paper presented at the 2016 IEEE International Conference on Healthcare Informatics (ICHI), Chicago, IL, USA, 4-7 Oct. 2016. https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=7776401 Date of access: 10 Aug. 2019.

McLachlan, S., Dube, K., Gallagher, T., Daley, B. & Walonoski, J. 2018. The ATEN Framework for Creating the Realistic Synthetic Electronic Health Record*.* In: Reyer, Z., Hugo, G., Ana, F. & Badia., S.B.i., eds*. Conference proceedings*. 11th International Joint Conference on Biomedical Engineering Systems and Technologies (BIOSTEC 2018), Funchal, Madeira, Portugal. SciTe Press. pp. 220-230.

Mengshu, H., Xianliang, L., Xu, Z. & Chuan, Z. 2005. A trust model of p2p system based on confirmation theory. *ACM SIGOPS Operating Systems Review*, 39(1):56-62. https://doi.org/10.1145/1044552.1044558

Mohanasundaram, N. 2020. *Non Linear Predictive Modelling for IC Engine Using Artificial Neural Network*. Paper presented at the 2020 Fourth International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC), Palladam, India, 7-9 Oct. 2020. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9243342 Date of access: 19. Apr. 2021.

Mohd Nawi, N., Atomi, W. & Rehman Gillani, S.M. 2013. The Effect of Data Pre-processing on Optimized Training of Artificial Neural Networks. *Procedia Technology*, 11:32-39. https://doi.org/10.1016/j.protcy.2013.12.159

Mohsenzadeh, A. & Motameni, H. 2015. A trust model between cloud entities using fuzzy mathematics. *Journal of Intelligent & Fuzzy Systems*, 29(5):1795-1803. http://nwulib.nwu.ac.za/login?url=http://search.ebscohost.com/login.aspx?direct=true&db=buh&AN=110936732&site=eds-live https://doi.org/10.3233/IFS-151657

Mongillo, M. 2011. Choosing Basis Functions and Shape Parameters for Radial Basis Function Methods. *Society for Industrial and Applied Mathematics (SIAM) Undergraduate Research Online*, 4:190-209. http://dx.doi.org/10.1137/11S010840

Moody, J. & Darken, C.J. 1989. Fast learning in networks of locally-tuned processing units. *Neural computation*, 1(2):281-294. https://doi.org/10.1162/neco.1989.1.2.281

Mui, L., Mohtashemi, M. & Halberstadt, A. 2002. *A computational model of trust and reputation*. Paper presented at the Proceedings of the 35th Annual Hawaii International Conference on System Sciences, Big Island, HI, USA 10-10 Jan. https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=994181 Date of access: 13 Aug. 2020.

Nagisetty, A. & Gupta, G.P. 2019. *Framework for detection of malicious activities in IoT networks using keras deep learning library*. Paper presented at the 2019 3rd international conference on computing methodologies and communication (ICCMC), Erode, India. https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=8819688 Date of access: 29 Mar 2020.

Narayanan, A. & Shmatikov, V. 2008. *Robust de-anonymization of large sparse datasets*. Paper presented at the IEEE Symposium on Security and Privacy (sp 2008), Oakland, CA, USA https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4531148 Date of access: 10 Sep. 2021.

Nepal, S., Sherchan, W., Hunklinger, J. & Bouguettaya, A. 2010. *A Fuzzy Trust Management Framework for Service Web*. Paper presented at the 2010 IEEE International Conference on Web Services, Miami, FL, USA 5-10 Jul 2010. https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=5552769 Date of access: 21 Aug. 2020.

Neruda, R. & Kudová, P. 2005. Learning methods for radial basis function networks. *Future Generation Computer Systems*, 21(7):1131-1142. https://doi.org/10.1016/j.future.2004.03.013

Noor, R.A.M., Ahmad, Z., Don, M.M. & Uzir, M.H. 2010. Modelling and control of different types of polymerization processes using neural networks technique: A review. *The Canadian Journal of Chemical Engineering*, 88(6):1065-1084. https://onlinelibrary.wiley.com/doi/abs/10.1002/cjce.20364 https://doi.org/10.1002/cjce.20364

Nori, F., Deypir, M., Hadi, M. & Ziarati, K. 2011. A new sliding window based algorithm for frequent closed itemset mining over data streams. *Journal of Systems and Software*, 86(3):615-623. https://doi.org/10.1016/j.jss.2012.10.011

Ocampo-Vega, R., Sanchez-Ante, G., Falcon-Morales, L.E. & Sossa, H. 2016. Automatic construction of radial-basis function networks through an adaptive partition algorithm. In: Martínez-Trinidad., J., Carrasco-Ochoa., J.A., Ayala-Ramírez., V., Olvera-López., J. & Jiang., X., eds*. Conference Proceedings*. 8th Mexican Conference on Pattern Recognition (MCPR 2016), Guanajuato, Mexico. Switzerland: Springer. pp. 198-207.

Olmedilla, D., Rana, O.F., Matthews, B. & Nejdl, W. 2006. *Security and trust issues in semantic grids*. (Dagstuhl Seminar Proceedings). https://drops.dagstuhl.de/opus/volltexte/2006/408/pdf/05271.OlmedillaDaniel.Paper.408.pdf Date of access: 11 Jun. 2019.

Park, J. & Sandberg, I.W. 1991. Universal Approximation Using Radial-Basis-Function Networks. *Neural Computation*, 3(2):246-257. https://doi.org/10.1162/neco.1991.3.2.246

Pillai, G.G., Putrus, G.A. & Pearsall, N.M. 2014. Generation of synthetic benchmark electrical load profiles using publicly available load and weather data. *International Journal of Electrical Power & Energy Systems*, 61:1-10. http://www.sciencedirect.com/science/article/pii/S0142061514001070 https://doi.org/10.1016/j.ijepes.2014.03.005

Pislaru, C. & Shebani, A. 2014. *Identification of nonlinear systems using radial basis function neural network*. Paper presented at the International Conference on Automation and Control Engineering (ICACE 2014), London, United Kingdom. https://publications.waset.org/abstracts/14775/identification-of-nonlinear-systems-using-radial-basis-function-neural-network Date of access: 15 Mar. 2021.

Preetham, V.V. 2016. *Mathematical foundation for Activation Functions in Artificial Neural Networks*. https://medium.com/autonomous-agents/mathematical-foundation-for-activation-functions-in-artificial-neural-networks-a51c9dd7c089 Date of access: 7 Dec. 2019.

Qasem, S., Shamsuddin, S.M. & Zain, A. 2013. Multi-objective hybrid evolutionary algorithms for radial basis function neural network design. *Knowledge-Based Systems*, 27:475–497. https://doi.org/10.1016/j.knosys.2011.10.001

Raitoharju, J., Kiranyaz, S. & Gabbouj, M. 2016. Training radial basis function neural networks for classification via class-specific clustering. *IEEE transactions on neural networks and learning systems*, 27(12):2458-2471. http://dx.doi.org/10.1109/TNNLS.2015.2497286

Reiter, J.P. 2004. *Simultaneous use of multiple imputation for missing data and disclosure limitation*. (Survey Methodology). https://www150.statcan.gc.ca/n1/en/pub/12-001-x/2004002/article/7755-eng.pdf?st=sFJCaULK Date of access: 20 Aug. 2018.

Rich, J. & Mulalic, I. 2012. Generating synthetic baseline populations from register data. *Transportation Research Part A: Policy and Practice*, 46(3):467-479. https://doi.org/10.1016/j.tra.2011.11.002

Rivas, V.M., Merelo, J., Castillo, P., Arenas, M.G. & Castellano, J. 2004. Evolving RBF neural networks for time-series forecasting with EvRBF. *Information Sciences*, 165(3-4):207-220. https://doi.org/10.1016/j.ins.2003.09.025

Rojas, I., Pomares, H., Gonzalez, J., Ros, E., Salmeron, M., Ortega, J. & Prieto, A. 2000. *A new radial basis function networks structure: application to time series prediction*. Paper presented at the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium, 2000, Como, Italy, 27-27 July 2000. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=860812 Date of access: 14 Sep. 2021.

Rosenblatt, F. 1961. *Principles of neurodynamics perceptrons and the theory of brain mechanisms*. https://apps.dtic.mil/sti/pdfs/AD0256582.pdf Date of access: 23 Jun. 2020.

Roychowdhury, V.P., Siu, K.-Y. & Kailath, T. 1995. Classification of Linearly Non-Separable Patterns by Linear Threshold Elements. *IEEE Transactions on Neural Networks*, 6(2):318-331. http://dx.doi.org/10.1109/72.363468

Rubin, D.B. 1993. Discussion: Statistical disclosure limitation. *Journal of official Statistics*, 9(2):461-468.

Rumelhart, D.E., Hinton, G.E. & Williams, R.J. 1986. Learning representations by back-propagating errors. *Nature Publishing Group*, 323(6088):533-536. https://doi.org/10.1038/323533a0

Rumelhart, D.E., Durbin, R., Golden, R. & Chauvin, Y. 1995. Backpropagation: The basic theory. In: Chauvin., Y. & Rumelhart., D.E., eds. Hillsdale, New Jersey: Lawrence Erlbaum Associates, Inc. pp. 1-34.

Ruslan, F.A., Samad, A.M., Zain, Z.M. & Adnan, R. 2013. *Modelling flood prediction using Radial Basis Function Neural Network (RBFNN) and inverse model: A comparative study*. Paper presented at the 2013 IEEE International Conference on Control System, Computing and Engineering, Penang, Malaysia 29 Nov.-1 Dec. 2013. https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6720031 Date of access: 13 Sep. 2021.

Rykiel, E.J. 1996. Testing ecological models: the meaning of validation. *Ecological Modelling*, 90(3):229-244. http://www.sciencedirect.com/science/article/pii/0304380095001522 https://doi.org/10.1016/0304-3800(95)00152-2

Sargent, R.G. 1984. *A tutorial on verification and validation of simulation models*. Paper presented at the 16th conference on Winter simulation (WSC 84), Dallas TX. https://repository.lib.ncsu.edu/bitstream/handle/1840.4/4929/1984_0017.pdf?sequence=1 Date of access: 15 Mar. 2021.

Scheidegger, A. & Maurer, M. 2012. Identifying biases in deterioration models using synthetic sewer data. *Water Science and Technology*, 66(11):2363-2369. https://doi.org/10.2166/wst.2012.471

Shakya, S., Hongchun, Y., Xinjun, C. & Liming, S. 2011. *Application of radial basis Function Neural Network for fishery forecasting*. Paper presented at the 2011 IEEE International Conference on Computer Science and Automation Engineering, Shanghai, 10-12 June 2011. https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=5952682 Date of access: 20 Jan. 2021.

Shanmugamani, R. 2018. *Deep Learning for Computer Vision: Expert techniques to train advanced neural networks using TensorFlow and Keras*. Packt Publishing Ltd.

Sherchan, W., Nepal, S. & Bouguettaya, A. 2011. *A Trust Prediction Model for Service Web*. Paper presented at the 2011 IEEE 10th International Conference on Trust, Security and Privacy in Computing and Communications, Changsha, China, 16-18 Nov. 2011. https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6120827 Date of access: 13 Sep. 2020.

Shi, J., Bochmann, G.v. & Adams, C. 2005. A trust model with statistical foundation. In: Dimitrakos, T. & Martinelli, F., eds. *Conference proceedings*. IFIP World Computer Congress, TC 1 (IFIP WCC TC1 2004) Formal Aspects in Security and Trust International Federation for Information Processing, Toulouse, France. Boston, MA: Springer. pp. 145-158.

Shukla, N. & Fricklas, K. 2018. *Machine learning with TensorFlow*. Manning Publications. Available from Livebook manning: https://livebook.manning.com/book/machine-learning-with-tensorflow/chapter-1/ Date of access: 19 Sep. 2021.

Sierra, C. & Debenham, J. 2005. *An information-based model for trust*. Paper presented at the Fourth international joint conference on Autonomous agents and multiagent systems (AAMAS05), Netherlands. https://dl.acm.org/doi/pdf/10.1145/1082473.1082549 Date of access: 12 Jan. 2021.

SimilarWeb. 2016. *SimilarWeb*. https://www.similarweb.com/website/nwu.ac.za Date of access: 2 Nov. 2017.

Singal, H. & Kohli, S. 2016. Trust Necessitated through Metrics: Estimating the Trustworthiness of Websites. *Procedia Computer Science*, 85:133-140. http://nwulib.nwu.ac.za/login?url=http://search.ebscohost.com/login.aspx?direct=true&db=edselp&AN=S1877050916305476&site=eds-live https://doi.org/10.1016/j.procs.2016.05.199

Society, I.C. 1998. *IEEE Standard for Software Verification and Validation*. (IEEE Std 1012-1998). https://ieeexplore.ieee.org/servlet/opac?punumber=5672 Date of access: 18 Apr. 2021.

Sossa, H., Griselda, C. & Guevara, E. 2014. New Radial Basis Function Neural Network Architecture for Pattern Classification: First Results. In: Eduardo, B.-C. & Edwin, H., eds. *Conference proceedings*. 19th Iberoamerican Congress (CIARP 2014), Puerto Vallarta, Mexico. Cham Heidelberg New York: Springer. pp. 706-713.

Strawn, G. 2014. Masterminds of the Arpanet. *IT Professional*, 16(3):66-68. https://doi.org/10.1109/MITP.2014.32

Sukhan, L. 1988. *Multilayer feedforward potential function network*. Paper presented at the IEEE 1988 International Conference on Neural Networks, San Diego, CA, USA https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=23844 Date of access: 10 Jan. 2021.

Surendra, H.S. & Mohan, H.S. 2017. A Review Of Synthetic Data Generation Methods For Privacy Preserving Data Publishing. *International Journal of Scientific & Technology Research*, 6(3):95-101. https://www.ijstr.org/final-print/mar2017/A-Review-Of-Synthetic-Data-Generation-Methods-For-Privacy-Preserving-Data-Publishing.pdf

Tahta, U.E., Sen, S. & Can, A.B. 2015. GenTrust: A genetic trust management model for peer-to-peer systems. *Applied Soft Computing*, 34(C):693-704. http://www.sciencedirect.com/science/article/pii/S156849461500280X http://dx.doi.org/10.1016/j.asoc.2015.04.053

Tappert, C.C. 2019. *Who Is the Father of Deep Learning?* Paper presented at the International Conference on Computational Science and Computational Intelligence (CSCI 2019), Las

Vegas, NV, USA, 5-7 Dec. 2019. http://dx.doi.org/10.1109/CSCI49370.2019.00067 Date of access: 11 Apr. 2021.

Team, R.C. 2016. *The R Project for Statistical Computing*. https://www.r-project.org/ Date of access: 2 Nov. 2017.

Thacker, B.H., Doebling, S.W., Hemez, F.M., Anderson, M.C., Pepin, J.E. & Rodriguez, E.A. 2004. *Concepts of Model Verification and Validation*. https://inis.iaea.org/collection/NCLCollectionStore/_Public/36/030/36030870.pdf?r=1 Date of access: 20 Jan. 2019.

Tran, H., Hitchens, M., Varadharajan, V. & Watters, P. 2005. *A trust based access control framework for P2P file-sharing systems*. Paper presented at the 38th Annual Hawaii International Conference on System Sciences, Big Island, HI, USA. https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=1385863 Date of access: 10 Aug. 2020.

Tsvetovat, M. & Carley, K.M. 2005. *Generation of realistic social network datasets for testing of analysis and simulation tools*. (CMU-ISRI-05-130). http://reports-archive.adm.cs.cmu.edu/anon/isri2005/CMU-ISRI-05-130.pdf Date of access: 14 Jun. 2020.

United States General Accounting, O. 2001. *Record linkage and privacy: Issues in creating new federal research and statistical information*. Washington, D.C.: U.S. General Accounting Office.

Van Paassen, A.H. & Luo, Q.X. 2002. Weather data generator to study climate change on buildings. *Building Services Engineering Research and Technology*, 23(4):251-258. Date of access: 09 Sep. 2019. https://doi.org/10.1191/0143624402bt048oa

Vega-Corona, A., Zárate-Banda, M., Barrón-Adame, J.M., Martínez-Celorio, R.A. & Andina, D. 2008. Design of the Approximation Function of a Pedometer Based on Artificial Neural Network for the Healthy Life Style Promotion in Diabetic Patients. In. 2008 Seventh Mexican International Conference on Artificial Intelligence. pp. 325-329.

W3C. 2002. *Security*. https://www.w3.org/standards/xml/security Date of access: 3 Nov. 2017.

Walonoski, J., Kramer, M., Nichols, J., Quina, A., Moesel, C., Hall, D., ... McLachlan, S. 2018. Synthea: An approach, method, and software mechanism for generating synthetic patients and the synthetic electronic health care record. *Journal of the American Medical Informatics Association*, 25(3):230-238. https://doi.org/10.1093/jamia/ocx079

Wang, C. & Chi, C.h. 2006. *Quantitative Trust Based on Actions*. Paper presented at the IEEE International Conference on Web Services (ICWS'06), Chicago, IL, USA. https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=4032026 Date of access: 10 Sep. 2021.

Wang, H., Zou, B., Guo, G., Zhang, J. & Yang, Z. 2015. *Optimal and Effective Web Service Composition with Trust and User Preference*. Paper presented at the IEEE International Conference on Web Services (ICWS 2015), New York, NY, USA June 27 2015-July 2 2015. https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=7195586 Date of access: 10 Apr. 2021.

Wang, S., Zhang, L. & Wang, S. 2008. *A Quantitative Evaluation Approach of Subjective Trust for E-Commerce*. Paper presented at the International Conference on Computational Intelligence for Modelling Control & Automation (CIMCA 2008), Vienna, Austria, 10-12 Dec. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5172721 Date of access: 10 Sep. 2020.

Wang, Y. & Vassileva, J. 2003. *Trust and reputation model in peer-to-peer networks*. Paper presented at the International Conference on Peer-to-Peer Computing (P2P 2003), Linköping, Sweden 1-3 Sept. 2003. https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=1231515 Date of access: 10 Aug. 2019.

Weston, J., Bordes, A., Chopra, S., M. Rush, A., Van Merriënboer, B., Joulin, A. & Mikolov, T. 2015. *Towards AI-Complete Question Answering: A Set of Prerequisite Toy Tasks*. Paper presented at the International Conference on Learning Representations (ICLR 2016), San Juan, Puerto Rico. https://arxiv.org/abs/1502.05698 Date of access: 20 Mar. 2020.

Wettschereck, D. & Dietterich, T. 1992. *Improving the performance of radial basis function networks by learning center locations*. Paper presented at the Advances in Neural Information Processing Systems 4, (NIPS 91'), Denver, Colorado, USA. https://dl.acm.org/doi/10.5555/2986916.2987056 Date of access: 25 Mar. 2021.

Witkowski, M. & Pitt, J. 2000. *Objective trust-based agents: Trust and Trustworthiness in a Multi-agent Trading Society*. Paper presented at the MultiAgent Systems, 2000. Proceedings. Fourth International Conference on MultiAgent Systems, Boston, MA, USA, 10-12 July 2000 https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=858526 Date of access: 10 Sep. 2021.

Wu, J., Chen, X.-Y., Zhang, H., Xiong, L.-D., Lei, H. & Deng, S.-H. 2019. Hyperparameter Optimization for Machine Learning Models Based on Bayesian Optimizationb. *Journal of Electronic Science and Technology*, 17(1):26-40.

https://www.sciencedirect.com/science/article/pii/S1674862X19300047
https://doi.org/10.11989/JEST.1674-862X.80904120

Xie, T., Yu, H. & Wilamowski, B. 2011. *Comparison between Traditional Neural Networks and Radial Basis Function Networks* Paper presented at the 2011 IEEE International Symposium on Industrial Electronics, Gdansk, Poland. https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=5984328 Date of access: 20 Aug. 2019.

Yan, X.-B., Wang, Z., Yu, S.-H. & Li, Y.-J. 2005. *Time series forecasting with RBF neural network*. Paper presented at the Fourth International Conference on Machine Learning and Cybernetics, Guangzhou, China https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1527764 Date of access: 10 Aug. 2021.

Yao, Y., Lian, Z., Hou, Z. & Liu, W. 2006. An innovative air-conditioning load forecasting model based on RBF neural network and combined residual error correction. *International Journal of Refrigeration*, 29(4):528-538. https://doi.org/10.1016/j.ijrefrig.2005.10.008

Yong-sheng, Z. & Ying, W. 2010. *Research on Trust-Authorization-Based Access Control Model for Web Services*. Paper presented at the Second International Conference on Networks Security, Wireless Communications and Trusted Computing (NSWCTC 2010), Wuhan, China, 24-25 April 2010. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5480833 Date of access: 10 Mar. 2020.

Yu, B. & Singh, M.P. 2002. *An evidential model of distributed reputation management*. Paper presented at the Proceedings of the First International Joint Conference on Autonomous Agents and Multi-agent Systems: Part 1 (AAMAS '02), Bologna, Italy. https://dl.acm.org/doi/pdf/10.1145/544741.544809 Date of access: 18 Mar. 2019.

Yu, H., Xie, T., Paszczynski, S. & Wilamowski, B.M. 2011. Advantages of Radial Basis Function Networks for Dynamic System Design. *IEEE Transactions on Industrial Electronics*, 58(12):5438-5450. https://doi.org/10.1109/TIE.2011.2164773

Zacaria, S., du Toit, T. & Venter, L. 2022. An Alternative Trust Calculation Method Using Radial Basis Function Neural Networks. In: Smuts, M. & Moorcroft, R., eds. *Conference Proceedings*. Southern Africa Telecommunication Networks and Applications Conference (SATNAC) 2022, Fancourt, George, Western Cape, South Africa. Telkom. pp. 159-164.

Zarei, R., Monemi, A. & Marsono, M. 2015. Automated Dataset Generation for Training Peer-to-Peer Machine Learning Classifiers. *Journal of Network and Systems Management*, 23(1):89-

110.
https://nwulib.nwu.ac.za/login?url=https://search.ebscohost.com/login.aspx?direct=true&db=aph&AN=100255290&site=eds-live https://doi.org/10.1007/s10922-013-9279-z

Zhou, Q. & Ooka, R. 2021. Influence of data preprocessing on neural network performance for reproducing CFD simulations of non-isothermal indoor airflow distribution. *Energy and Buildings*, 230:110525. https://www.sciencedirect.com/science/article/pii/S0378778820317783 https://doi.org/10.1016/j.enbuild.2020.110525

APPENDIX A. PROGRAM CODE

A.1 2021PeerTrustSetupSameInputP2P

```
use [2020PeerTrustNewP2P]
if exists (select 1 from sysobjects where type ='p'and name ='2021PeerTrustSetupSameInputP2P')
drop procedure [2021PeerTrustSetupSameInputP2P]
go
create proc [2021PeerTrustSetupSameInputP2P]
@PeerCount numeric(30),
@InteractionCount numeric(30)
as
declare @PeerNames nvarchar(50)
declare @Count nvarchar(18)
declare @InteractingPeer nvarchar(50)
declare @WithPeer nvarchar(50)
declare @TransactionSize nvarchar(50)
declare @TransactionSizeValue float
declare @NumberOfPeersInTheNetwork numeric
DECLARE @TId numeric
/*****************************************************************************************
Procedure Name: 2021PeerTrustSetupSameInputP2P
Created By: Smitha Zacaria
Updated Date: June 20 2017,Jan 09 2019,July 22 2021
*****************************************************************************************/
begin
--Deleting values in the tables
  if exists (select 1 from Peer)
```

```
begin
  delete from Peer
end
if exists (select 1 from Interaction)
begin
  delete from Interaction
end
if exists (select 1 from InteractionSet)
begin
  delete from InteractionSet
end
if exists (select 1 from InteractionSetP2P)
begin
  delete from InteractionSetP2P
end
if exists (select 1 from PeerInitial)
begin
  delete from PeerInitial
end
if exists (select 1 from InteractionInitials)
begin
  delete from InteractionInitials
end
```

```
insert into InteractionInitials (NumberOfPeers,NumberOfInteractions) VALUES (
@PeerCount,@InteractionCount )
```

```
--Creating peers and giving initial trust values

  set @NumberOfPeersInTheNetwork = @PeerCount

  set @Count = 1

  while @PeerCount >0

  begin

    set @PeerNames =  'Peer'+@Count


    if exists (select 1 from sysobjects where type ='u'and name ='Peer')

    begin


      insert into Peer (PeerName) VALUES ( @PeerNames )

       update Peer set CurrentTrustValue =  RAND()*(1-0)+0 ,TotalNumberOfInteractionMade= 0 where
PeerName=@PeerNames


    end

    set @Count = @Count+1

    set @PeerCount= @PeerCount-1

  end


  insert into PeerInitial(PeerName,InitialTrustValue)  ( select PeerName,CurrentTrustValue from Peer)


--Determining transactions, transaction size and transaction size values that needs to happen


  while (@InteractionCount>0)

  begin

    set @InteractingPeer = ( select   top 1 PeerName   from Peer order by newid())
```

```
set @WithPeer = ( select top 1 PeerName from Peer where PeerName != @InteractingPeer order by newid())

set @TransactionSize = ( select top 1 Size from TransactionSize order by newid())


if @TransactionSize ='Extra Large'

  set @TransactionSizeValue = (SELECT RAND()*(0.75-1)+1 )

else if @TransactionSize ='Large'

  set @TransactionSizeValue = (SELECT RAND()*(0.5-0.75)+0.75 )

else if @TransactionSize ='Medium'

  set @TransactionSizeValue = (SELECT RAND()*(0.25-0.5)+0.5 )

else if @TransactionSize ='Small'

  set @TransactionSizeValue = (SELECT RAND()*(0-0.25)+0.25 )


set @TransactionSizeValue = ROUND(@TransactionSizeValue, 3, 0);



if exists (select 1 from sysobjects where type ='u'and name ='Interaction')

  begin

      insert into InteractionSet (InteractingPeer,WithPeer,TransactionSize,TransactionSizeValue) VALUES ( @InteractingPeer,@WithPeer,@TransactionSize,@TransactionSizeValue )

    SET @TId = (SELECT MAX(Id) from InteractionSet)

                                    insert into InteractionSetP2P (TId,InteractingPeer,WithPeer,TransactionSize,TransactionSizeValue) VALUES (@TId, @InteractingPeer,@WithPeer,@TransactionSize,@TransactionSizeValue )

                                    insert into InteractionSetP2P (TId,InteractingPeer,WithPeer,TransactionSize,TransactionSizeValue) VALUES (@TId, @WithPeer,@InteractingPeer,@TransactionSize,@TransactionSizeValue )

  end
```

```
    SET @InteractionCount= @InteractionCount -1
  end


end
```

A.2 2021PeerTrustNewSameInputP2P

```
use [2020PeerTrustNewP2P]
if exists (select 1 from sysobjects where type ='p'and name ='2021PeerTrustNewSameInputP2P')
drop procedure [2021PeerTrustNewSameInputP2P]
go
create proc [2021PeerTrustNewSameInputP2P]
@PeerCount numeric(30),
@InteractionCount numeric(30),
@TransactionContextFlag numeric,
@CommunityContextFlag numeric
/*******************************************************************************************
@TransactionContextFlag = 0 to off,1 to on,
@CommunityContextFlag  = 0 to off,1 to on
OCT 03 2020 : If Transaction context flag is on alpha should be equal to one and transaction context factor value should be transaction size value. else 1 for all.
*******************************************************************************************/
as
declare @InteractingPeer nvarchar(50)
declare @WithPeer nvarchar(50)
declare @TransactionSize nvarchar(50)
declare @TransactionSizeValue float
declare @CurrentTrustValueOfInteractingPeer float
```

```
declare @CurrentTrustValueOfWithPeer float

declare @TrustValueOfWithPeer float

declare @NC_RC float

declare @C_RC float

declare @NC_RW float

declare @C_RW float

declare @CountOfMaxProbalility numeric

declare @CountOfMinProbalility numeric

declare @MaximumFieldValue float

declare @MinimumFieldValue float

declare @ComplaintStatus nvarchar(50)

declare @Feedback  numeric

declare @SatisfactionReceivedFromWithPeer float

declare @TotalInteractionbyInteractingPeer numeric(30)

declare @Interaction_cursor_rowcount int

declare @Id numeric

declare @TransactionContextFactor float

declare @CommunityContextFactor float

declare @ProductOfSatisfactionCredibilityTransactinFactor float

declare @TotalNoFeedbackInteractingPeerFiled numeric

declare @TotalTrustValueOfWithPeers float

declare @TotalInteractionDoneSofar numeric

declare @TotalInteraction numeric

declare @NewTrustValueofInteractingPeer float

declare @SumProductOfSatisfactionCredibilityTransactinFactor float

declare @Alphaconstant float

declare @BetaConstant float
```

```
declare @TotalNoInteractionDoneByOtherPeerWithInteractingPeer numeric

declare @MinId int

declare @MaxId int

declare @TId int

declare @Productvalue float

/*********************************************************************************************

Procedure Name:      2021PeerTrustNewSameInputP2P

Created By:       Smitha Zacaria

Date:          June 20 2017,Jan 09 2019, July 21 2021

Journal:       PeerTrust: Supporting Reputation-Based Trust for Peer-to-Peer Electronic Communities

Reference:        (Li & Ling, 2004)

For turning off transaction context factor

set @Alphaconstant = 1 and @TransactionContextFactor = 1


For turning off Community context

set @BetaConstant = 0

*********************************************************************************************/


begin

--Deleting values from the tables.

  if exists (select 1 from Peer)

  begin

    delete from Peer

          insert into Peer (PeerName,CurrentTrustValue,TotalNumberOfInteractionMade)( select
PeerName,InitialTrustValue,0 from PeerInitial)

  end

  if exists (select 1 from Interaction)
```

```
  begin
    delete from Interaction
  end
  if exists (select 1 from P2PInteraction)
  begin
    delete from P2PInteraction
  end

  if exists (select 1 from PeerInitial)
  begin
                                    update PeerInitial set FinalTrustValue
=NULL,TotalNumberOfInteractionMade=NULL,TotalSatisfactionReceived=NULL,TotalTransactionSizeVal
ue=NULL,NumberFeedbackGiven=NULL
  end

  set @MaxId =( select Max(Id) from InteractionSetP2P)
  set @MinId = ( select Min(Id) from InteractionSetP2P)
  set @TId = 0
--Each interactions are happening
  while (@MinId<=@MaxId)
  begin
    set @TId = @TId+1

    set @InteractingPeer = ( select  InteractingPeer  from InteractionSetP2P WHERE Id =@MinId )
    set @WithPeer = ( select  WithPeer  from InteractionSetP2P WHERE Id =@MinId)
    set @TransactionSize =  ( select  TransactionSize  from InteractionSetP2P WHERE Id =@MinId)
     set @TransactionSizeValue = (select  TransactionSizeValue  from InteractionSetP2P WHERE Id
=@MinId)
```

```
set @CurrentTrustValueOfInteractingPeer = (select CurrentTrustValue from Peer where PeerName = @InteractingPeer)

set @CurrentTrustValueOfWithPeer = (select CurrentTrustValue from Peer where PeerName = @WithPeer)


if exists (select 1 from sysobjects where type ='u'and name ='Interaction')

begin

insert into Interaction (InteractingPeer,WithPeer,TransactionSize,TransactionSizeValue,CurrentTrustValueOfWithPeer,CurrentTrustValueofInteractingPeer,InteractionStatus) VALUES ( @InteractingPeer,@WithPeer,@TransactionSize,@TransactionSizeValue,@CurrentTrustValueOfWithPeer,@CurrentTrustValueOfInteractingPeer,'Busy' )

end

--Obtaining feedback regarding current interaction.

set @Feedback =  FLOOR(RAND()*(1-0+1))+0

if @Feedback = 0

begin

  set @SatisfactionReceivedFromWithPeer  = 0

end

else

begin

  set @SatisfactionReceivedFromWithPeer = RAND()*(1-0)+0

end


update Interaction set ComplaintStatus = @ComplaintStatus,Feedback =@Feedback ,SatisfactionReceivedFromWithPeer= @SatisfactionReceivedFromWithPeer where InteractingPeer = @InteractingPeer and WithPeer=@WithPeer and InteractionStatus ='Busy'

set @TotalInteractionbyInteractingPeer = (select count(Id) from Interaction where InteractingPeer = @InteractingPeer)

set @TotalTrustValueOfWithPeers = (select sum(CurrentTrustValueOfWithPeer) from Interaction where InteractingPeer = @InteractingPeer )
```

```
        update Interaction set TotalInteractionDoneSofar =@TotalInteractionbyInteractingPeer where InteractingPeer = @InteractingPeer and InteractionStatus ='Busy'

    set @TotalInteraction = (select count(*) from interaction where InteractingPeer = @InteractingPeer)

--Implementing transaction context and community context factors.

    if @TransactionContextFlag =0

    begin

      set @Alphaconstant = 1

      set @TransactionContextFactor = 1

            update Interaction set TransactionContextFactor =@TransactionContextFactor where InteractingPeer = @InteractingPeer and InteractionStatus ='Busy'



    end

    if @TransactionContextFlag =1

    begin

      set @Alphaconstant = 0.5

      set @TransactionContextFactor = @TransactionSizeValue

            update Interaction set TransactionContextFactor =@TransactionContextFactor where InteractingPeer = @InteractingPeer and InteractionStatus ='Busy'

    end

    if @CommunityContextFlag = 0

    begin

      set @BetaConstant = 0

      set @CommunityContextFactor = 0

            update Interaction set CommunityContextFactor =@CommunityContextFactor where InteractingPeer = @InteractingPeer and InteractionStatus ='Busy'

    end

    if @CommunityContextFlag = 1

    begin
```

```
set @BetaConstant = 0.5

set @CommunityContextFactor = 1

update Interaction set CommunityContextFactor =@CommunityContextFactor where InteractingPeer = @InteractingPeer and InteractionStatus ='Busy'

end

--Calculating trust values of peers


CREATE TABLE #cursortable2 (Id numeric,InteractingPeer nvarchar(50),WithPeer nvarchar(50),TransactionSizeValue float,Feedback numeric,SatisfactionReceivedFromWithPeer float,TrustValueOfWithPeer float,TransactionContextFactor float,ProductOfSatisfactionCredibilityTransactinFactor float,TotalInteractionDoneSofar Numeric)


declare Interaction_cursor cursor for select Id, InteractingPeer,WithPeer,TransactionSizeValue,Feedback,SatisfactionReceivedFromWithPeer,CurrentTrustValueOfWithPeer,TransactionContextFactor,TotalInteractionDoneSofar from Interaction where InteractingPeer = @InteractingPeer order by Id

open Interaction_cursor

fetch next from Interaction_cursor into @Id ,@InteractingPeer,@WithPeer,@TransactionSizeValue,@Feedback,@SatisfactionReceivedFromWithPeer,@TrustValueOfWithPeer,@TransactionContextFactor,@TotalInteractionDoneSofar

set @Interaction_cursor_rowcount = @@FETCH_STATUS

while @Interaction_cursor_rowcount = 0

begin

insert #cursortable2 (Id,InteractingPeer,WithPeer,TransactionSizeValue,Feedback,SatisfactionReceivedFromWithPeer,TrustValueOfWithPeer,TransactionContextFactor,TotalInteractionDoneSofar)VALUES ( @Id,@InteractingPeer,@WithPeer,@TransactionSizeValue,@Feedback,@SatisfactionReceivedFromWithPeer,@TrustValueOfWithPeer,@TransactionContextFactor,@TotalInteractionDoneSofar )


select @Productvalue = @SatisfactionReceivedFromWithPeer*@TrustValueOfWithPeer*@TransactionContextFactor
```

```
        if @Productvalue = 0

        begin

          set @ProductOfSatisfactionCredibilityTransactinFactor = 0

        end

        else

        begin

                                        set @ProductOfSatisfactionCredibilityTransactinFactor =
(@Productvalue)/@TotalTrustValueOfWithPeers



        end

                    update #cursortable2 set TransactionContextFactor=@TransactionContextFactor,
ProductOfSatisfactionCredibilityTransactinFactor = @ProductOfSatisfactionCredibilityTransactinFactor
where InteractingPeer = @InteractingPeer and WithPeer=@WithPeer and Id = @Id

                                    fetch next from Interaction_cursor into @Id
,@InteractingPeer,@WithPeer,@TransactionSizeValue,@Feedback,@SatisfactionReceivedFromWithPe
er,@TrustValueOfWithPeer,@TransactionContextFactor,@TotalInteractionDoneSofar

        set @Interaction_cursor_rowcount = @@FETCH_STATUS



    end

                        set @SumProductOfSatisfactionCredibilityTransactinFactor = (select
sum(ProductOfSatisfactionCredibilityTransactinFactor) from #cursortable2)



     set @TotalInteractionbyInteractingPeer = (select count(*) from Interaction where InteractingPeer =
@InteractingPeer)

      set @TotalNoInteractionDoneByOtherPeerWithInteractingPeer = (select count(*) from Interaction
where WithPeer = @InteractingPeer)



    close Interaction_cursor

    deallocate Interaction_cursor
```

```
DROP TABLE #cursortable2



IF @TId = 1

begin

set @TransactionSize = ( select TransactionSize from Interaction WHERE InteractionStatus ='Busy')

set @TransactionSizeValue = (select TransactionSizeValue from Interaction WHERE InteractionStatus ='Busy')

set @SatisfactionReceivedFromWithPeer = (select SatisfactionReceivedFromWithPeer from Interaction WHERE InteractionStatus ='Busy')

set @CurrentTrustValueOfInteractingPeer = (select CurrentTrustValueOfInteractingPeer from Interaction where InteractionStatus ='Busy')

set @CurrentTrustValueOfWithPeer = (select CurrentTrustValueOfWithPeer from Interaction where InteractionStatus ='Busy')

set @TransactionContextFactor = (select TransactionContextFactor from Interaction where InteractionStatus ='Busy')

set @CommunityContextFactor = (select CommunityContextFactor from Interaction where InteractionStatus ='Busy')



set @Feedback = (select Feedback from Interaction where InteractionStatus ='Busy')



insert into P2PInteraction (InteractingPeer,WithPeer,NewTrustValueofInteractingPeer,InteractionStatus ) VALUES ( @InteractingPeer,@WithPeer ,@NewTrustValueofInteractingPeer,'Busy')



update P2PInteraction set TransactionSize = @TransactionSize ,TransactionSizeValue =@TransactionSizeValue, TransactionContextFactor= @TransactionContextFactor,CommunityContextFactor=@CommunityContextFactor,SatisfactionReceived FromWithPeer = @SatisfactionReceivedFromWithPeer,CurrentTrustValueOfInteractingPeer=@CurrentTrustValueOfIntera
```

```
ctingPeer,CurrentTrustValueOfWithPeer=@CurrentTrustValueOfWithPeer where InteractingPeer =
@InteractingPeer and InteractionStatus ='Busy'



        update P2PInteraction set FeedbackGivenByWithpeer = @Feedback where InteractingPeer =
@InteractingPeer and InteractionStatus ='Busy'



         update Interaction set Alphaconstant = @Alphaconstant, BetaConstant = @BetaConstant,
SumProductOfSatisfactionCredibilityTransactinFactor
=@SumProductOfSatisfactionCredibilityTransactinFactor,TotalInteractionbyInteractingPeer
=@TotalInteractionbyInteractingPeer

       ,InteractionStatus ='BusyBusy1' where  InteractionStatus ='Busy'




    end

    if @TId = 2

    begin


        set @SatisfactionReceivedFromWithPeer = (select SatisfactionReceivedFromWithPeer from
Interaction WHERE InteractionStatus ='Busy')



                    update P2PInteraction set SatisfactionReceivedFromInteractingPeer =
@SatisfactionReceivedFromWithPeer where InteractionStatus ='Busy'



       set @Feedback =  (select Feedback from Interaction where InteractionStatus ='Busy')

        update P2PInteraction set FeedbackGivenByInteractingPeer = @Feedback where WithPeer =
@InteractingPeer and InteractionStatus ='Busy'



         set @InteractingPeer = (select InteractingPeer from Interaction where InteractionStatus
='BusyBusy1')
```

```
set @TotalNoFeedbackInteractingPeerFiled = (select count(*) from Interaction where InteractingPeer = @InteractingPeer and Feedback != 0)-- check

set @TotalNoFeedbackInteractingPeerFiled = (select count(*) from P2PInteraction where InteractingPeer = @InteractingPeer and FeedbackGivenByInteractingPeer != 0)+(select count(*) from P2PInteraction where WithPeer = @InteractingPeer and FeedbackGivenByWithpeer != 0)-- check

update P2PInteraction set TotalNoFeedbackInteractingPeerFiled = @TotalNoFeedbackInteractingPeerFiled where InteractionStatus ='Busy'

update Interaction set TotalNoFeedbackInteractingPeerFiled = @TotalNoFeedbackInteractingPeerFiled where InteractionStatus ='BusyBusy1'

update Interaction set NewTrustValueofInteractingPeer = (Alphaconstant*SumProductOfSatisfactionCredibilityTransactinFactor) + (BetaConstant*(TotalNoFeedbackInteractingPeerFiled/TotalInteractionbyInteractingPeer)) where InteractionStatus ='BusyBusy1'

update P2PInteraction set NewTrustValueofInteractingPeer =(select NewTrustValueofInteractingPeer from Interaction where InteractionStatus ='BusyBusy1') where InteractionStatus ='Busy'

update Interaction set InteractionStatus ='Done' where InteractionStatus ='BusyBusy1'

update Interaction set Alphaconstant = @Alphaconstant, BetaConstant = @BetaConstant, SumProductOfSatisfactionCredibilityTransactinFactor
=@SumProductOfSatisfactionCredibilityTransactinFactor,TotalInteractionbyInteractingPeer
=@TotalInteractionbyInteractingPeer

,InteractionStatus ='BusyBusy2' where  InteractionStatus ='Busy'
```

```
set @InteractingPeer = (select InteractingPeer from Interaction where InteractionStatus ='BusyBusy2')

set @TotalNoFeedbackInteractingPeerFiled = (select count(*) from Interaction where InteractingPeer = @InteractingPeer and Feedback != 0)

set @TotalNoFeedbackInteractingPeerFiled = (select count(*) from P2PInteraction where InteractingPeer = @InteractingPeer and FeedbackGivenByInteractingPeer != 0)+(select count(*) from P2PInteraction where WithPeer = @InteractingPeer and FeedbackGivenByWithpeer != 0)-- check

update P2PInteraction set TotalNoFeedbackWithPeerFiled = @TotalNoFeedbackInteractingPeerFiled where InteractionStatus ='Busy'

update Interaction set TotalNoFeedbackInteractingPeerFiled = @TotalNoFeedbackInteractingPeerFiled where InteractionStatus ='BusyBusy2'

update Interaction set NewTrustValueofInteractingPeer = (Alphaconstant*SumProductOfSatisfactionCredibilityTransactinFactor) + (BetaConstant*(TotalNoFeedbackInteractingPeerFiled/TotalInteractionbyInteractingPeer)) where InteractionStatus ='BusyBusy2'

update P2PInteraction set NewTrustValueofWithPeer =(select NewTrustValueofInteractingPeer from Interaction where InteractionStatus ='BusyBusy2')where InteractionStatus ='Busy'

update Interaction set InteractionStatus ='Done' where InteractionStatus ='BusyBusy2'

--Updating new trust values of peers

update Peer set Peer.CurrentTrustValue = P2PInteraction.NewTrustValueofInteractingPeer

from Peer ,P2PInteraction

where Peer.PeerName= P2PInteraction.InteractingPeer

and P2PInteraction.InteractionStatus = 'Busy'
```

```
        update Peer set Peer.CurrentTrustValue = P2PInteraction.NewTrustValueofWithPeer
        from Peer ,P2PInteraction
        where Peer.PeerName= P2PInteraction.WithPeer
        and P2PInteraction.InteractionStatus = 'Busy'

        update P2PInteraction set InteractionStatus ='Done' where  InteractionStatus ='Busy'

        set @TId = 0
      end

      SET @MinId = @MinId+1

   end
   update PeerInitial set PeerInitial.FinalTrustValue = Peer.CurrentTrustValue
   from PeerInitial ,Peer
   where PeerInitial.PeerName= Peer.PeerName

    update PeerInitial set PeerInitial.TotalNumberOfInteractionMade = (select count(Interaction.Id) from
Interaction where PeerInitial.PeerName= Interaction.InteractingPeer)
   from PeerInitial ,Interaction
   where PeerInitial.PeerName= Interaction.InteractingPeer

   update PeerInitial
     set TotalSatisfactionReceived = (select sum(Interaction.SatisfactionReceivedFromWithPeer)from
Interaction where PeerInitial.PeerName= Interaction.InteractingPeer)
   from PeerInitial ,Interaction
   where PeerInitial.PeerName= Interaction.InteractingPeer
```

```
  update PeerInitial

   set NumberFeedbackGiven = (select count(*)from Interaction where PeerInitial.PeerName= Interaction.WithPeer and Interaction.Feedback != 0)

  from PeerInitial ,Interaction

  where PeerInitial.PeerName= Interaction.InteractingPeer



  update PeerInitial

   set TotalTransactionSizeValue = (select sum(Interaction.TransactionSizeValue)from Interaction where PeerInitial.PeerName= Interaction.InteractingPeer)

  from PeerInitial ,Interaction

  where PeerInitial.PeerName= Interaction.InteractingPeer



--Displaying all the details of the transactions that nees for the data set.

   select Id , ROW_NUMBER()  OVER (ORDER BY Id)  AS TransactionCount,InteractingPeer ,WithPeer as OtherPeer,TransactionSize,TransactionSizeValue,FeedbackGivenByWithpeer as FeedbackGivenByOtherPeer,SatisfactionReceivedFromWithPeer as SatisfactionReceivedFromOtherPeer,FeedbackGivenByInteractingPeer,SatisfactionReceivedFromInteractingPeer,CurrentTrustValueofInteractingPeer,CurrentTrustValueOfWithPeer as CurrentTrustValueOfOtherPeer,TotalNoFeedbackInteractingPeerFiled,TotalNoFeedbackWithPeerFiled as TotalNoFeedbackOtherPeerFiled,NewTrustValueofInteractingPeer,NewTrustValueofWithPeer as NewTrustValueofOtherPeer from P2PInteraction -- ----2021 May 26



end
```

A.3 2021PeerTrust_Do_BinaryNorm

```
use [2020PeerTrustNewP2P]

if exists (select 1 from sysobjects where type ='p'and name ='2021PeerTrust_Do_BinaryNorm')

drop procedure [2021PeerTrust_Do_BinaryNorm]

go
```

```
create proc [2021PeerTrust_Do_BinaryNorm]

@NumberOfPeers numeric(30)

as

declare @BinaryTable nvarchar(50)

declare @MinTranId numeric(18,0)

declare @MaxTranId numeric(18,0)

declare @InputCount numeric(18,0)

declare @min as float

declare @max as float

declare @OutputTable nvarchar(50)

declare @FieldName nvarchar(50)

declare @OutputCount numeric(30)

declare @WindowCount numeric(30)

declare @Count numeric(30)

declare @FieldValue nvarchar(50)

declare @WindowInputCount numeric(30)

declare @RowCount numeric(30)

declare @RowNumber numeric(30)

declare @InputFieldName nvarchar(50)



Begin

        set @BinaryTable = 'BinaryOutput'
```

```
if exists (select 1 from sys.Tables where type ='u'and name = 'BinaryOutput')

begin

        EXEC ('drop table BinaryOutput' )

end



EXEC ('create table BinaryOutput(RowNumber Numeric(18,0))')



Set @InputCount =@NumberOfPeers

while @InputCount >= 1

begin

        set @FieldName= 'InteractingPeerBinary'+convert(nvarchar(50),@InputCount)

        EXEC('alter table BinaryOutput add "'+@FieldName+'" nvarchar(max) ')


        Set @InputCount =@InputCount-1

end
```

```
Set @InputCount =@NumberOfPeers

while @InputCount >= 1

begin

        set @FieldName= 'OtherPeerBinary'+convert(nvarchar(50),@InputCount)

        EXEC('alter table BinaryOutput add "'+@FieldName+'" nvarchar(max) ')


        Set @InputCount =@InputCount-1

end




Set @InputCount =4

while @InputCount >= 1

begin

        set @FieldName= 'TransactionSizeBinary'+convert(nvarchar(50),@InputCount)

        EXEC('alter table BinaryOutput add "'+@FieldName+'" nvarchar(max) ')


        Set @InputCount =@InputCount-1

end




EXEC ('alter table BinaryOutput add [Id] [numeric](18, 0)  NULL,

[InteractingPeer] [nvarchar](50) NULL,

[OtherPeer] [nvarchar](50) NULL,

[TransactionSize] [nvarchar](max) NULL,
```

[TransactionSizeValue] [float] NULL,

[NormTransactionSizeValue] [float] NULL,

[FeedbackGivenByOtherpeer] [numeric](18, 0) NULL,

[SatisfactionReceivedFromOtherPeer] [float] NULL,

[NormSatisfactionReceivedFromOtherPeer] [float] NULL,

[FeedbackGivenByInteractingPeer] [numeric](18, 0) NULL,

[SatisfactionReceivedFromInteractingPeer] [float] NULL,

[NormSatisfactionReceivedFromInteractingPeer] [float] NULL,

[CurrentTrustValueofInteractingPeer] [float] NULL,

[NormCurrentTrustValueofInteractingPeer] [float] NULL,

[CurrentTrustValueOfOtherPeer] [float] NULL,

[NormCurrentTrustValueOfOtherPeer] [float] NULL,

[TotalNoFeedbackInteractingPeerFiled] [numeric](18, 0) NULL,

[NormTotalNoFeedbackInteractingPeerFiled][float] NULL,

[TotalNoFeedbackOtherPeerFiled] [numeric](18, 0) NULL,

[NormTotalNoFeedbackOtherPeerFiled] [float] NULL,

[NewTrustValueofInteractingPeer] [float] NULL,

[NewTrustValueofOtherPeer] [float] NULL,

[InteractingPeerNameNumber] [numeric](18, 0) NULL,

[OtherPeerNameNumber] [numeric](18, 0) NULL')

```
insert into  BinaryOutput (
RowNumber,
Id,
InteractingPeer,
OtherPeer,
TransactionSize,
TransactionSizeValue,
FeedbackGivenByOtherpeer,
SatisfactionReceivedFromOtherPeer,
FeedbackGivenByInteractingPeer,
SatisfactionReceivedFromInteractingPeer,
CurrentTrustValueofInteractingPeer,
CurrentTrustValueOfOtherPeer,
TotalNoFeedbackInteractingPeerFiled,
TotalNoFeedbackOtherPeerFiled,
NewTrustValueofInteractingPeer,
NewTrustValueofOtherPeer


)
select ROW_NUMBER()  OVER (ORDER BY Id),
            Id,
InteractingPeer,
WithPeer,
TransactionSize,
```

```
TransactionSizeValue,

FeedbackGivenByWithpeer,

SatisfactionReceivedFromWithPeer,

FeedbackGivenByInteractingPeer,

SatisfactionReceivedFromInteractingPeer,

CurrentTrustValueofInteractingPeer,

CurrentTrustValueOfWithPeer,

TotalNoFeedbackInteractingPeerFiled,

TotalNoFeedbackWithPeerFiled,

NewTrustValueofInteractingPeer,

NewTrustValueofWithPeer

from UsedP2PInteraction
```

```
update BinaryOutput set TransactionSizeBinary4=0,TransactionSizeBinary3=0,TransactionSizeBinary2=0,TransactionSizeBinary1 = 1 where TransactionSize ='Small'
```

```
update BinaryOutput set TransactionSizeBinary4=0,TransactionSizeBinary3=0,TransactionSizeBinary2=1,TransactionSizeBinary1 = 0 where TransactionSize ='Medium'

update BinaryOutput set TransactionSizeBinary4=0,TransactionSizeBinary3=1,TransactionSizeBinary2=0,TransactionSizeBinary1 = 0 where TransactionSize ='Large'

update BinaryOutput set TransactionSizeBinary4=1,TransactionSizeBinary3=0,TransactionSizeBinary2=0,TransactionSizeBinary1 = 0 where TransactionSize ='Extra Large'
```

```
update BinaryOutput set InteractingPeerNameNumber = REPLACE(InteractingPeer, 'Peer', '')

update BinaryOutput set OtherPeerNameNumber = REPLACE(OtherPeer, 'Peer', '')


        Set @InputCount =@NumberOfPeers

        while @InputCount >= 1

        begin

                set @FieldName= 'InteractingPeerBinary'+convert(nvarchar(50),@InputCount)

                EXEC('update  BinaryOutput set '''+@FieldName+''' =0 ')


                set @FieldName= 'OtherPeerBinary'+convert(nvarchar(50),@InputCount)

                EXEC('update  BinaryOutput set '''+@FieldName+''' =0 ')


                Set @InputCount =@InputCount-1

        end




set @MinTranId = (select min(Id) from BinaryOutput)

set @MaxTranId = (select max(Id) from BinaryOutput)



Set @InputCount =@MinTranId

        while @InputCount <= @MaxTranId
```

```
        begin

                set     @FieldName=     'InteractingPeerBinary'+        convert(nvarchar(50),(select
InteractingPeerNameNumber from BinaryOutput where  Id =@InputCount))


                EXEC('update    BinaryOutput   set   '''+@FieldName+'''   =1   where    BinaryOutput.Id
='+@InputCount+'')


                set         @FieldName=         'OtherPeerBinary'+        convert(nvarchar(50),(select
OtherPeerNameNumber from BinaryOutput where  Id =@InputCount))

                EXEC('update    BinaryOutput   set   '''+@FieldName+'''   =1   where    BinaryOutput.Id
='+@InputCount+'')


                Set @InputCount =@InputCount+1

        end


-------------------------------NormTotalNoFeedbackInteractingPeerFiled--------------------


Set @min =( select min(TotalNoFeedbackInteractingPeerFiled) from BinaryOutput)

Set @max =(select max(TotalNoFeedbackInteractingPeerFiled) from BinaryOutput)


update BinaryOutput

set NormTotalNoFeedbackInteractingPeerFiled = (TotalNoFeedbackInteractingPeerFiled -@min)/(@max-
@min)
```

```
---------------------------------NormTotalNoFeedbackOtherPeerFiled----------------------

Set @min =( select min(TotalNoFeedbackOtherPeerFiled) from BinaryOutput)

Set @max =(select max(TotalNoFeedbackOtherPeerFiled) from BinaryOutput)




update BinaryOutput

set NormTotalNoFeedbackOtherPeerFiled = (TotalNoFeedbackOtherPeerFiled -@min)/(@max-@min)




-------------------------------------TransactionSizeValue---------------------------------------

Set @min =( select min(TransactionSizeValue) from BinaryOutput)

Set @max =(select max(TransactionSizeValue) from BinaryOutput)




update BinaryOutput

set NormTransactionSizeValue = (TransactionSizeValue -@min)/(@max-@min)


-------------------------------SatisfactionReceivedFromOtherPeer-------------------------

Set @min =( select min(SatisfactionReceivedFromOtherPeer) from BinaryOutput)

Set @max =(select max(SatisfactionReceivedFromOtherPeer) from BinaryOutput)
```

```
update BinaryOutput

set NormSatisfactionReceivedFromOtherPeer = (SatisfactionReceivedFromOtherPeer -@min)/(@max-@min)


----------SatisfactionReceivedFromInteractingPeer---------


Set @min =( select min(SatisfactionReceivedFromInteractingPeer) from BinaryOutput)

Set @max =(select max(SatisfactionReceivedFromInteractingPeer) from BinaryOutput)




update BinaryOutput

set NormSatisfactionReceivedFromInteractingPeer = (SatisfactionReceivedFromInteractingPeer -@min)/(@max-@min)


----------CurrentTrustValueofInteractingPeer------


Set @min =( select min(CurrentTrustValueofInteractingPeer) from BinaryOutput)

Set @max =(select max(CurrentTrustValueofInteractingPeer) from BinaryOutput)
```

```
update BinaryOutput

set NormCurrentTrustValueofInteractingPeer = (CurrentTrustValueofInteractingPeer -@min)/(@max-@min)



----------CurrentTrustValueOfOtherPeer----------

Set @min =( select min(CurrentTrustValueOfOtherPeer) from BinaryOutput)

Set @max =(select max(CurrentTrustValueOfOtherPeer) from BinaryOutput)



update BinaryOutput

set NormCurrentTrustValueOfOtherPeer = (CurrentTrustValueOfOtherPeer -@min)/(@max-@min)

--------------------------------------------------------------------------------



select * from BinaryOutput



Select * into TempBinaryOutput from BinaryOutput order by RowNumber ASC
```

```
ALTER TABLE TempBinaryOutput DROP COLUMN Id,InteractingPeer,OtherPeer,TransactionSize,TransactionSizeValue,TotalNoFeedbackInteractingPeerFiled,TotalNoFeedbackOtherPeerFiled,InteractingPeerNameNumber,OtherPeerNameNumber,SatisfactionReceivedFromOtherPeer,SatisfactionReceivedFromInteractingPeer,CurrentTrustValueofInteractingPeer,CurrentTrustValueOfOtherPeer



select * from TempBinaryOutput order by RowNumber ASC


if exists (select 1 from sys.Tables where type ='u'and name = 'UsedBinaryOutput')
        begin
                EXEC ('delete from UsedBinaryOutput' )
        end


insert into UsedBinaryOutput select * from TempBinaryOutput     order by RowNumber ASC


 DROP TABLE TempBinaryOutput


end
```

A.4 Python Program for PeerTrustRBFNN

1 TrustRBFNN.py

```
import pyodbc
import csv
```

```
import pandas as pd

import os

import os.path


import datetime

import numpy as np


from numpy import loadtxt


import pandas as pd


import math


import shutil

import tensorflow


from tensorflow import keras

from tensorflow.keras import layers

from tensorflow.keras.layers import Activation, Dense, BatchNormalization,Input

from tensorflow.keras.models import Sequential

#from tensorflow.keras.callbacks import LearningRateScheduler

import random as rn


from rbflayer import RBFLayer, InitCentersRandom

from kmeans_initializer import InitCentersKMeans

from initializer import InitFromFile
```

```
from matplotlib import pyplot


#setting seed value to 42 just before  building the RBFNN

seed_value= 42

os.environ['PYTHONHASHSEED']=str(seed_value)

np.random.seed(seed_value)

tensorflow.random.set_seed(seed_value)

rn.seed(None)


####################Set the input values

beta_min = 0

window_size_min = 1

hidden_nodes_min= 1


beta_max = 2

window_size_max = 15

hidden_nodes_max= 200

learning_rates = [0.1, 0.01, 0.001, 0.0001, 0.00001, 0.000001]

hours_to_run=24

epochs_value = 100


##########################################


##Delete old mse_results.csv file and add new file with heading only and remove all old plyplots#############################################################


os.remove('D:/RBFNN_code/mse_results.csv')#Delete old mse_results.csv
```

```
os.remove('D:/RBFNN_code/mse_min.csv')#Delete old mse_min.csv

os.remove('D:/RBFNN_code/BestTrustRBFNNmodel_mse_results.csv')#Delete old mse_min.csv

shutil.rmtree('D:/RBFNN_code/Pyplots/') # remove all old plyplots

shutil.rmtree('D:/RBFNN_code/BestModelPyplots/') # remove all old BestModelPyplots


shutil.copyfile('D:/RBFNN_code/NewFiles/mse_results.csv', 'D:/RBFNN_code/mse_results.csv')#add new file with heading only

shutil.copyfile('D:/RBFNN_code/NewFiles/mse_min.csv', 'D:/RBFNN_code/mse_min.csv')#add new file with heading only

shutil.copyfile('D:/RBFNN_code/NewFiles/BestTrustRBFNNmodel_mse_results.csv',
'D:/RBFNN_code/BestTrustRBFNNmodel_mse_results.csv')#add new file with heading only

os.mkdir('D:/RBFNN_code/Pyplots')#Creating new directory for plyplots.

os.mkdir('D:/RBFNN_code/BestModelPyplots')#Creating new directory for BestModelPyplots.


########################################## Time loop startts here

end_time = datetime.datetime.now() + datetime.timedelta(hours=hours_to_run)

Experiment_Number = 1

while end_time.timestamp() - datetime.datetime.now().timestamp() > 0:

  #randomnly select hyperparameters #########

  rn.seed(None)

  betas_value = rn.uniform(beta_min, beta_max)

  WindowSize = rn.randint(window_size_min, window_size_max)

  NumberOfHiddenNodes = rn.randint(hidden_nodes_min, hidden_nodes_max)

  #learning rate selection and applying it to compile process

  learning_rate = rn.choice(learning_rates)

  opt=keras.optimizers.RMSprop(learning_rate)


  ##########################data gen#################################################
```

```
                cnxn = pyodbc.connect('DRIVER={SQL Server};SERVER=P25380249;DATABASE=Standards;Trusted_Connection=yes;')

    cursor = cnxn.cursor()

    #data=cursor.execute('Proc_test2 @Rowcount=?',(6))

     data=cursor.execute('update [WindowInitials] set StartingWindowSize=?,WindowCountIncrementValue=?,NextWindowSize=?',(WindowSize,0,WindowSize))

    cnxn.commit()

    cursor.close()

    cnxn.close()


    #seting seed value to 42 just before data generation with window size

    seed_value= 42

    os.environ['PYTHONHASHSEED']=str(seed_value)

    np.random.seed(seed_value)

    tensorflow.random.set_seed(seed_value)

    rn.seed(None)

    #checking if the window data is already created.

    if os.path.isfile('D:/RBFNN_code/UsedWindowDataTiles/%s.csv'% str(WindowSize)):

        print ("Window size data file exist-")

        print(WindowSize)

                shutil.copyfile('D:/RBFNN_code/UsedWindowDataTiles/%s.csv'% str(WindowSize),'D:/RBFNN_code/Trust2021.csv' )#copying from  backups


    else:

        print(WindowSize)

        print ("Window size file not exist")


        os.system("D:\RBFNN_code\WindowInputData\Run_window_program.bat")
```

```
                                        cnxn                =                pyodbc.connect('DRIVER={SQL
Server};SERVER=P25380249;DATABASE=Standards;Trusted_Connection=yes;')

    cursor = cnxn.cursor()

    cursor.execute('SELECT * FROM [dbo].[OutputWindow] order by [RowNumber] asc')



    data = cursor.fetchall()

    #print (data)

    with open('D:\RBFNN_code\Trust2021.csv', 'w', newline='') as f_handle:

      writer = csv.writer(f_handle)

      for row in data:

        writer.writerow(row)

    cursor.close()

    cnxn.close()

      shutil.copyfile('D:/RBFNN_code/Trust2021.csv',  'D:/RBFNN_code/UsedWindowDataTiles/%s.csv'%
str(WindowSize))#keeping a backup of all data

  ##############################################################################

  trust_df = pd.read_csv("Trust2021.csv")

  print(trust_df.describe())

  # load the dataset

  Trustdataset = loadtxt('Trust2021.csv', delimiter=',',skiprows=1)# skip first row as first row is headings.

  ######################################################

  #Set the NumPy random seed to 42

  #Shuffle the data before you split the data into a training, validation and test dataset.

  np.random.seed(42)

  np.random.shuffle(Trustdataset)

##############################################################################

   TotalNumberOfInputOutputFields = len( Trustdataset[0] )# Must reduce first column as first column is
row count.
```

```
TotalNumberOfInputFields = TotalNumberOfInputOutputFields - 2-1 #  Must remove first column as first column is row count and 2 outputcolumns.

TotalNumberOfDataRows = len(list(Trustdataset))# Window 1 rows = 10000.

TrainDataRowCountMax = int((TotalNumberOfDataRows*70)/100 )# 70% of total data is for training.

ValDataRowCountMax = int((TotalNumberOfDataRows*90)/100 )# 20% of remaining 30 % data is for validating.

print('TotalNumberOfInputFields =', TotalNumberOfInputFields)

# split into input (X) and output (y) variables

# 70% of total data is for training.

# in WindowSize 1, 63 input 2 output total 66 columns in the file including first column(row count).

X_train = Trustdataset[0:TrainDataRowCountMax,1:TotalNumberOfInputOutputFields-2]

Y_train = Trustdataset[0:TrainDataRowCountMax,TotalNumberOfInputOutputFields-2:TotalNumberOfInputOutputFields]



# 20% of remaining 30 % data is for validating.

X_val = Trustdataset[TrainDataRowCountMax:ValDataRowCountMax,1:TotalNumberOfInputOutputFields-2]

Y_val = Trustdataset[TrainDataRowCountMax:ValDataRowCountMax,TotalNumberOfInputOutputFields-2:TotalNumberOfInputOutputFields]



# Remaining data is for testing.(10% of 30 % data is for testing.

X_test = Trustdataset[ValDataRowCountMax:TotalNumberOfDataRows,1:TotalNumberOfInputOutputFields-2]

Y_test = Trustdataset[ValDataRowCountMax:TotalNumberOfDataRows,TotalNumberOfInputOutputFields-2:TotalNumberOfInputOutputFields]



TotalNumberOfOutputFields = 2



TrustRBFNNmodel = Sequential()
```

```
        TrustRBFLayer = RBFLayer(NumberOfHiddenNodes, betas=betas_value, input_shape=(TotalNumberOfInputFields,))

    TrustRBFNNmodel.add(TrustRBFLayer)


    OutputLayer = Dense(TotalNumberOfOutputFields, activation='sigmoid')

    TrustRBFNNmodel.add(OutputLayer)


    TrustRBFNNmodel.summary()


    #setting seed value to 0 just before learning rate selection

      # seed_value= 0

      #np.random.seed(seed_value)

      #tensorflow.random.set_seed(seed_value)

      # rn.seed(seed_value)

    ###############################


    ################################################


    # compile the keras model

    TrustRBFNNmodel.compile(loss='mse', optimizer=opt, metrics=['mse'])


    # fit the keras model (train)on the dataset

      TrustRBFNNmodelHistory = TrustRBFNNmodel.fit(X_train, Y_train, epochs=epochs_value , batch_size=10, validation_data=(X_val,Y_val))


    # plot metrics

    pyplot.plot(TrustRBFNNmodelHistory.history['mse'])
```

```
    pyplot.plot(TrustRBFNNmodelHistory.history['val_mse'])


     # While you are searching for the best hyperparameters, you must evaluate your model on the validation set

    # Only after many experiments and after you have found the best hyperparameters, you evaluate the model on the test set


    # Determine validation MSE. This is the last training validation value

    validation_mse = TrustRBFNNmodelHistory.history['val_mse'][-1]


    # Save the MSE value to a csv file

    output_file = open('mse_results.csv', 'a')

     text = str(Experiment_Number) + ', ' + str(NumberOfHiddenNodes) + ', ' + str(WindowSize) + ', ' + str(betas_value)+', '+str(learning_rate) + ','+ str(epochs_value)+ ',' + str(validation_mse)

    output_file.write(text + '\n')

    output_file.close()

                    pyplot.savefig('D:/RBFNN_code/Pyplots/%s.png'%                    ("Experiment_Number-"+str(Experiment_Number)+"-Window-"+str(WindowSize)+"-Hiddennodes-"+str(NumberOfHiddenNodes)))

    Experiment_Number =Experiment_Number +1

    print ((end_time.timestamp() - datetime.datetime.now().timestamp())/60)

# Finding minimum value of validation_mse


df = pd.read_csv("D:/RBFNN_code/mse_results.csv")

row_count = df.shape[0]


min_validation_mse = df.at[0, df.columns.values[6]]


for i in range(0,row_count,1):
```

```
    if min_validation_mse > df.at[i, df.columns.values[6]]:
        min_validation_mse =df.at[i, df.columns.values[6]]


for i in range(0,row_count,1):
    if min_validation_mse == df.at[i, df.columns.values[6]]:
        min_validation_mse =df.at[i, df.columns.values[6]]
        Experiment_Number =df.at[i, df.columns.values[0]]
        NumberOfHiddenNodes=df.at[i, df.columns.values[1]]
        WindowSize=df.at[i, df.columns.values[2]]
        betas_value=df.at[i, df.columns.values[3]]
        learning_rate=df.at[i, df.columns.values[4]]
        epochs_value = df.at[i, df.columns.values[5]]
        output_file = open('D:/RBFNN_code/mse_min.csv', 'a')
        text = str(Experiment_Number) + ', ' + str(NumberOfHiddenNodes) + ', ' + str(WindowSize) + ', ' +
str(betas_value)+', '+str(learning_rate) + ','+ str(epochs_value) +',' + str(min_validation_mse)
        output_file.write(text + '\n')
        output_file.close()


# Printing the details of experiments having minimum value of validation_mse


print("=================================================================")
print("Details of experiment having min_validation_mse : ",min_validation_mse)
print("=================================================================")
df = pd.read_csv("D:/RBFNN_code/mse_min.csv")
row_count = df.shape[0]
for i in range(row_count):
    min_validation_mse =df.at[i, df.columns.values[6]]
```

```
    Experiment_Number = df.at[i, df.columns.values[0]]
    NumberOfHiddenNodes=df.at[i, df.columns.values[1]]
    WindowSize=df.at[i, df.columns.values[2]]
    betas_value=df.at[i, df.columns.values[3]]
    learning_rate=df.at[i, df.columns.values[4]]
    epochs_value = df.at[i, df.columns.values[5]]
    print("Experiment_Number =",Experiment_Number)
    print("NumberOfHiddenNodes =",NumberOfHiddenNodes)
    print("WindowSize =",WindowSize)
    print("betas_value =",betas_value)
    print("learning_rate =",learning_rate)
    print("epochs_valu =",epochs_value)
    print("min_validation_mse =",min_validation_mse)
    print("-------------------------------------")


#After training has stopped (after the 10 hours), do the following:
# Create the train, validate and test datasets by using the window size of the best model found.
# Train the best model found again on the training set.
#Evaluate this model on the test set and report the MSE on the training, validation and test sets.
##########SET opt using best model learning rate


opt=keras.optimizers.RMSprop(learning_rate)


# Create the train, validate and test datasets by using the window size of the best model found.
shutil.copyfile('D:/RBFNN_code/UsedWindowDataTiles/%s.csv'%
str(WindowSize),'D:/RBFNN_code/Trust2021.csv' )#copying from  backups
Trustdataset = loadtxt('Trust2021.csv', delimiter=',',skiprows=1)# skip first row as first row is headings.
```

```
np.random.seed(42)

np.random.shuffle(Trustdataset)

#############################################################################

TotalNumberOfInputOutputFields = len( Trustdataset[0] )# Must reduce first column as first column is row count.

TotalNumberOfInputFields = TotalNumberOfInputOutputFields - 2-1 #  Must remove first column as first column is row count and 2 outputcolumns.

TotalNumberOfDataRows = len(list(Trustdataset))# Window 1 rows = 10000.

TrainDataRowCountMax = int((TotalNumberOfDataRows*70)/100 )# 70% of total data is for training.

ValDataRowCountMax = int((TotalNumberOfDataRows*90)/100 )# 20% of remaining 30 % data is for validating.

print('TotalNumberOfInputFields =', TotalNumberOfInputFields)

# split into input (X) and output (y) variables

# 70% of total data is for training.

# in WindowSize 1, 63 input 2 output total 66 columns in the file including first column(row count).

X_train = Trustdataset[0:TrainDataRowCountMax,1:TotalNumberOfInputOutputFields-2]

Y_train = Trustdataset[0:TrainDataRowCountMax,TotalNumberOfInputOutputFields-2:TotalNumberOfInputOutputFields]


# 20% of remaining 30 % data is for validating.

X_val = Trustdataset[TrainDataRowCountMax:ValDataRowCountMax,1:TotalNumberOfInputOutputFields-2]

Y_val = Trustdataset[TrainDataRowCountMax:ValDataRowCountMax,TotalNumberOfInputOutputFields-2:TotalNumberOfInputOutputFields]


# Remaining data is for testing.(10% of 30 % data is for testing.

X_test = Trustdataset[ValDataRowCountMax:TotalNumberOfDataRows,1:TotalNumberOfInputOutputFields-2]
```

```
Y_test = Trustdataset[ValDataRowCountMax:TotalNumberOfDataRows,TotalNumberOfInputOutputFields-2:TotalNumberOfInputOutputFields]


TotalNumberOfOutputFields = 2

BestTrustRBFNNmodel = Sequential()

TrustRBFLayer = RBFLayer(NumberOfHiddenNodes, betas=betas_value, input_shape=(TotalNumberOfInputFields,))

BestTrustRBFNNmodel.add(TrustRBFLayer)

OutputLayer = Dense(TotalNumberOfOutputFields, activation='sigmoid')

BestTrustRBFNNmodel.add(OutputLayer)

BestTrustRBFNNmodel.summary()


# compile the best model

BestTrustRBFNNmodel.compile(loss='mse', optimizer=opt, metrics=['mse'])


# fit the keras model (train)on the dataset

BestTrustRBFNNmodelHistory = BestTrustRBFNNmodel.fit(X_train, Y_train, epochs=epochs_value , batch_size=10, validation_data=(X_val,Y_val))

pyplot.plot(BestTrustRBFNNmodelHistory.history['mse'])

pyplot.plot(BestTrustRBFNNmodelHistory.history['val_mse'])


#While you are searching for the best hyperparameters, you must evaluate your model on the validation set

# Only after many experiments and after you have found the best hyperparameters, you evaluate the model on the test set

# Determine validation MSE. This is the last training validation value

training_mse = BestTrustRBFNNmodelHistory.history['mse'][-1]

validation_mse = BestTrustRBFNNmodelHistory.history['val_mse'][-1]
```

```
pyplot.savefig('D:/RBFNN_code/BestModelPyplots/%s.png'%                ("Window-"+str(WindowSize)+"-
Hiddennodes-"+str(NumberOfHiddenNodes)))


#Evaluate the model using test set

Test_DataSet_Evaluation = BestTrustRBFNNmodel.evaluate(X_test, Y_test,  batch_size=10)

Test_DataSet_mse = Test_DataSet_Evaluation[1]


#Evaluate the model using validation data set

Validation_DataSet_Evaluation = BestTrustRBFNNmodel.evaluate(X_val, Y_val,  batch_size=10)

Validation_DataSet_mse = Validation_DataSet_Evaluation[1]


# Save the MSE values of Best model  to a csv file

output_file = open('BestTrustRBFNNmodel_mse_results.csv', 'a')

text =  str(NumberOfHiddenNodes) + ', ' + str(WindowSize) + ', ' + str(betas_value)+', '+str(learning_rate)
+ ','+ str(epochs_value)+ ','+ str(training_mse) + ','+str(validation_mse) +','+ str(Test_DataSet_mse) +','+
str(Validation_DataSet_mse)

output_file.write(text + '\n')

output_file.close()
```

2 rbflayer.py

```
import tensorflow as tf

from tensorflow.keras.layers import Layer

from tensorflow.keras.initializers import RandomUniform, Initializer, Constant

import numpy as np


class InitCentersRandom(Initializer):

  """ Initializer for initialization of centers of RBF network

    as random samples from the given data set.
```

```
    # Arguments
        X: matrix, dataset to choose the centers from (random rows
          are taken as centers)
    """

    def __init__(self, X):
        self.X = X
        super().__init__()

    def __call__(self, shape, dtype=None):
        assert shape[1:] == self.X.shape[1:]  # check dimension

        # np.random.randint returns ints from [low, high) !
        idx = np.random.randint(self.X.shape[0], size=shape[0])

        return self.X[idx, :]

class RBFLayer(Layer):
    """ Layer of Gaussian RBF units.

    # Example

    ```python
        model = Sequential()
        model.add(RBFLayer(10,
                           initializer=InitCentersRandom(X),
                                                          ___
```
```

```
                betas=1.0,
                input_shape=(1,)))
  model.add(Dense(1))
```

Arguments
  output_dim: number of hidden units (i.e. number of outputs of the
            layer)
  initializer: instance of initiliazer to initialize centers
  betas: float, initial value for betas

 """

```
def __init__(self, output_dim, initializer=None, betas=1.0, **kwargs):

  self.output_dim = output_dim

  # betas is either initializer object or float
  if isinstance(betas, Initializer):
    self.betas_initializer = betas
  else:
    self.betas_initializer = Constant(value=betas)

  self.initializer = initializer if initializer else RandomUniform(
    0.0, 1.0)

   super().__init__(**kwargs)
```

```
def build(self, input_shape):

    self.centers = self.add_weight(name='centers',
                                   shape=(self.output_dim, input_shape[1]),
                                   initializer=self.initializer,
                                   trainable=True)
    self.betas = self.add_weight(name='betas',
                                 shape=(self.output_dim,),
                                 initializer=self.betas_initializer,
                                 # initializer='ones',
                                 trainable=True)

    super().build(input_shape)

def call(self, x):

    C = tf.expand_dims(self.centers, -1)  # inserts a dimension of 1
    H = tf.transpose(C-tf.transpose(x))  # matrix of differences
    return tf.exp(-self.betas * tf.math.reduce_sum(H**2, axis=1))

def compute_output_shape(self, input_shape):
    return (input_shape[0], self.output_dim)

def get_config(self):
    # have to define get_config to be able to use model_from_json
    config = {
```

```
        'output_dim': self.output_dim
    }
    base_config = super().get_config()
    return dict(list(base_config.items()) + list(config.items()))
```

3 kmeans_initializer.py

```
from tensorflow.keras.initializers import Initializer
from sklearn.cluster import KMeans

class InitCentersKMeans(Initializer):
  """ Initializer for initialization of centers of RBF network
    by clustering the given data set.
  # Arguments
    X: matrix, dataset
  """

  def __init__(self, X, max_iter=100):
    self.X = X
    self.max_iter = max_iter
    super().__init__()

  def __call__(self, shape, dtype=None):
    assert shape[1:] == self.X.shape[1:]

    n_centers = shape[0]
    km = KMeans(n_clusters=n_centers, max_iter=self.max_iter, verbose=0)
```

```
    km.fit(self.X)
    return km.cluster_centers_
```

4 initializer.py

```
from tensorflow.keras.initializers import Initializer
import numpy as np

class InitFromFile(Initializer):
  """ Initialize the weights by loading from file.

  # Arguments
    filename: name of file, should by .npy file
  """
  def __init__(self, filename):
    self.filename = filename
    super().__init__()

  def __call__(self, shape, dtype=None):
    with open(self.filename, "rb") as f:
      X = np.load(f, allow_pickle=True) # fails without allow_pickle
    assert tuple(shape) == tuple(X.shape)
    return X

  def get_config(self):
    return {
```

```
    'filename': self.filename
  }
```

5 Run_window_program.bat

```
@echo Running sql batch file

SQLCMD -s P25380249 -d Standards -E -i "D:\RBFNN_code\WindowInputData\Exe_smitha_proc_standards_do_smartwidows2021.sql"
```

6 Exe_smitha_proc_standards_do_smartwidows2021.sql

```
use Standards
exec smitha_proc_standards_do_smartwidows2021
```

7 smitha_proc_standards_do_smartwidows2021

```
use Standards

if exists (select 1 from sysobjects where type ='p'and name ='smitha_proc_standards_do_smartwidows2021')

drop procedure smitha_proc_standards_do_smartwidows2021

go

create proc smitha_proc_standards_do_smartwidows2021


as

declare @InputTable nvarchar(50)

declare @OutputTable nvarchar(50)

declare @InputCount numeric(30)

declare @FieldName nvarchar(50)

declare @OutputCount numeric(30)

declare @WindowCount numeric(30)

declare @Count numeric(30)
```

```
declare @FieldValue nvarchar(50)
declare @WindowInputCount numeric(30)
declare @RowCount numeric(30)
declare @RowNumber numeric(30)
declare @InputFieldName nvarchar(50)
declare @NumberOfInputs numeric(30)
declare @NumberOfOutputs numeric(30)
declare @WindowSize numeric(30)
/*****************************************************************************************
Procedure Name: smitha_proc_standards_do_smartwidows2021
Created By: Smitha Zacaria
Date: June 20 2020
Use: To do sliding window for the PeerTrust data
*****************************************************************************************/
Begin

  set @NumberOfInputs =63;
  set @NumberOfOutputs =2;
  set @WindowSize = (select NextWindowSize from WindowInitials);
  set @InputTable = 'InputWindow'
  set @OutputTable = 'OutputWindow'
--Removing tables and creating tables.
  if exists (select 1 from sys.Tables where type ='u'and name = 'InputWindow')
  begin
    exec ('drop table InputWindow' )
  end
  if exists (select 1 from sys.Tables where type ='u'and name = 'OutputWindow')
```

```
  begin
    exec ('drop table OutputWindow' )
  end

  exec ('create table InputWindow(RowNumber Numeric(18,0))')

  set @InputCount =1
  while @InputCount <= @NumberOfInputs
  begin
    set @FieldName= 'In'+convert(nvarchar(50),@InputCount)
    exec('alter table InputWindow add "'+@FieldName+'" varchar(12) ')
    set @InputCount =@InputCount+1
  end

  set @OutputCount =1
  while @OutputCount <= @NumberOfOutputs
  begin
    set @FieldName= 'Out'+convert(nvarchar(50),@OutputCount)
    exec('alter table InputWindow add "'+@FieldName+'" varchar(12) ')
    set @OutputCount =@OutputCount+1
  end
--Obtaining input data to do sliding window.
  bulk insert  InputWindow  from 'D:\RBFNN_code\WindowInputData\Windowinput.txt'
  delete from InputWindow where RowNumber is NULL

  exec ('create table OutputWindow(RowNumber Numeric(18,0))')
----------2021 june 14----------
```

```
  delete from InputWindow where RowNumber = 0

  exec ('insert into OutputWindow ( RowNumber )values( -1)')

------------------------2021 june 14-------------------------

--Creating fields to table according to window size to store sliding windo data.

  set @InputCount =1;

  while @InputCount <= (@NumberOfInputs*@WindowSize)

  begin

    set @FieldName= 'In'+convert(nvarchar(50),@InputCount);

    exec('alter table OutputWindow add "'+@FieldName+'" varchar(12) ');

    Set @InputCount =@InputCount+1;

  end


  set @OutputCount =1

  while @OutputCount <= @NumberOfOutputs

  begin

    set @FieldName= 'Out'+convert(nvarchar(50),@OutputCount)

    exec('alter table OutputWindow add "'+@FieldName+'" varchar(12) ')

    set @OutputCount =@OutputCount+1

  end


  select @RowCount=count(*) from InputWindow


   insert into OutputWindow (RowNumber)select RowNumber from InputWindow where RowNumber <=
@RowCount -( @WindowSize-1)


  set @WindowCount = 1

  set @InputCount = 1
```

```
--Windowing the input data and storing

  while @WindowCount <= @WindowSize


  begin

    set @WindowInputCount=1

    while @WindowInputCount <= @NumberOfInputs


    begin

      set @FieldName= 'In'+convert(nvarchar(50),@InputCount)

      set @InputFieldName= 'In'+convert(nvarchar(50),@WindowInputCount)

       exec('update OutputWindow set ''' + @FieldName +'''  = (Select ''' + @InputFieldName +''' from
InputWindow where InputWindow.RowNumber=OutputWindow.RowNumber+'+@WindowCount+'-1)')

      set @WindowInputCount=@WindowInputCount+1

      set @InputCount =@InputCount+1

    end

    set @WindowCount = @WindowCount+1

  end


  set @Count=1

  while @Count <= @NumberOfOutputs


  begin

    set @FieldName= 'Out'+convert(nvarchar(50),@Count)

       exec('update OutputWindow set ''' + @FieldName +'''  = (Select ''' + @FieldName +''' from
InputWindow where InputWindow.RowNumber=OutputWindow.RowNumber+'+@WindowSize+'-1)')


    set @Count =@Count+1

  end
```

```
----------2021 june 14----------
set @InputCount =1
while @InputCount <= @NumberOfInputs*@WindowSize
begin
  set @FieldName= 'In'+convert(nvarchar(50),@InputCount)
  exec('update OutputWindow set "'+@FieldName+'"  = "'+@FieldName+'"  where RowNumber=-1')

  set @InputCount =@InputCount+1
end

set @OutputCount =1
while @OutputCount <= @NumberOfOutputs
begin
  set @FieldName= 'Out'+convert(nvarchar(50),@OutputCount)
  exec('update OutputWindow set "'+@FieldName+'"  = "'+@FieldName+'"  where RowNumber=-1')

  set @OutputCount =@OutputCount+1
end
----------2021 june 14----------
        exec('UPDATE WindowInitials SET NextWindowSize = NextWindowSize+WindowCountIncrementValue')
end
```

A.5 2021ARDSGenerateMonths

```
use ARDS
if exists (select 1 from sysobjects where type ='p'and name ='2021ARDSGenerateMonths')
```

```
drop procedure [2021ARDSGenerateMonths]

go

create proc [2021ARDSGenerateMonths]

@MonthCount numeric(30),

@MaxNumberOfPeersPerMonth numeric(30),

@TrustContext nvarchar(max),

@FeatureCount int,

@PercentageUptime float


as

/*****************************************************************************************

Procedure Name: 2021ARDSGenerateMonths

Created By: Smitha Zacaria

Date: June 20 2020,January 20 2021

Use: To generate given number of months and features for ARDS trust calculation

*****************************************************************************************/


declare @Count nvarchar(50)


begin

--Deleting data from tables

  if exists (select 1 from InputValues)

     begin

       delete from InputValues

     end
```

```
insert into InputValues (MonthCount,MaxNumberOfPeerPerMonth,TrustContext,FeatureCount,PercentageUptime)values(@MonthCount,@MaxNumberOfPeersPerMonth,@TrustContext,@FeatureCount,@PercentageUptime)

  if exists (select 1 from FeatureCount)
    begin
      delete from FeatureCount
    end


  if exists (select 1 from MonthlyData)
    begin
      delete from MonthlyData
    end
--Generating given number months for the ARDS trust calculation.
  set @Count = 1
  while @Count<= @MonthCount
  begin
insert into MonthlyData (id,MonthName,ExpectingMonthlyPercentageUptimeStatus,NumberOfPeers)values(@Count,'Month'+@Count,'0',FLOOR(RAND()*(@MaxNumberOfPeersPerMonth-0+1)+0))
    set @Count = @Count+1
  end
  --Generating given number of features for the ARDS trust calculation.
  set @Count = 1
  while @Count<= @FeatureCount
  begin
    insert into FeatureCount (CountNumber,NumberStatus)values(@Count,'0')
    set @Count = @Count+1
  end
```

```
  select * from InputValues

  if exists (select 1 from MonthlyTransaction)
  begin
    delete from MonthlyTransaction
  end
  if exists (select 1 from TransactionDetails)
  begin
    delete from TransactionDetails
  end

end
```

A.6 2021ARDSDoSingleMonthlyuptimeCreditbackTransactionsRandom

```
use ARDS
if exists (select 1 from sysobjects where type ='p'and name ='2021ARDSDoSingleMonthlyuptimeCreditbackTransactionsRandom')
drop procedure [2021ARDSDoSingleMonthlyuptimeCreditbackTransactionsRandom]
go
create proc [2021ARDSDoSingleMonthlyuptimeCreditbackTransactionsRandom]
 @Gamma1 float,
 @Gamma2 float,
 @Gamma3 float,
 @Gamma4 float,
 @Alpha float,
```

```
 @Beta float

as

declare @MonthCount numeric(30)

declare @MaxNumberOfPeersPerMonth numeric(30)

declare @TrustContext nvarchar(max)

declare @FeatureCount int

declare @PercentageUptime float

declare @MonthlyRandomPercentageUptime float

declare @RandomTrustStatus nvarchar(50)

declare @Month_cursor_Count nvarchar(50)

declare @Count int

declare @Count3 int

Declare @ParaCount nvarchar(18)

declare @FieldName nvarchar(50)

declare @Month_cursor_rowcount int

declare @MonthName nvarchar(50)

declare @NumberOfPeers numeric

declare @FeatureValue REAL

declare @ExpectingMonthlyPercentageUptimeStatus nvarchar(50)

declare @Transaction_cursor_rowcount int

declare @TransactionCount numeric

declare @TotalPeersParticipatedPerMonth int

declare @RandomExpectingTrustvalue real

declare @TotalRandomMonthlyExpectingTrustsum real
```

```
declare @RandonAvailabilityPercentage real

declare @FeatureSum float

declare @Commentpart nvarchar(max)

Declare @ParaCountNumber nvarchar(18)

declare @TransactionCountSoFar numeric

declare @CurrentMonthlyTransactionTrustValueSofar real

declare @CreditReturnFactor nvarchar(50)

declare @CreditReturnSuccess int

declare @TransactionExpectingTrustStatus nvarchar(max)

declare @MonthlyCreditreturnFailedNumber int

declare @CreditreturnFailedNumberSofar int

declare @CurrentTrustValueSofar float

declare @TrustValueOfARDS float

declare @MonthlyAvailabilitypercentageSofar real

declare @FeatureAvailability float

declare @MonthlyCreditReturnFactor nvarchar(50)

declare @PeerName nvarchar(50)

declare @AverageFeatureAvailabilitySoFar float

declare @NoOfCreditreturnRequiredSofar int

declare @TrustOfAvailabilityPart float

declare @TrustDeductionAmount float


Begin

        if exists (select 1 from InputValues)
```

```
		begin
			select	@MonthCount =MonthCount,
				@MaxNumberOfPeersPerMonth =MaxNumberOfPeerPerMonth,
				@TrustContext =TrustContext,
				@FeatureCount =FeatureCount,
				@PercentageUptime =PercentageUptime
			from InputValues

		end


	if exists (select 1 from MonthlyTransaction)
		begin
			delete from MonthlyTransaction
		end

	insert into MonthlyTransaction (id,MonthName,ExpectingMonthlyPercentageUptimeStatus,NumberOfPeers)select id, MonthName,ExpectingMonthlyPercentageUptimeStatus,NumberOfPeers from MonthlyData order by id

	if exists (select 1 from sys.Tables where type ='u'and name = 'TransactionDetails')
			begin
				EXEC ('drop table TransactionDetails')
			end
```

```
EXEC ('create table TransactionDetails(TransactionCountSoFar Numeric(18,0),MonthName nvarchar(max),TotalPeersParticipatedPerMonth Numeric(18,0),PeerName nvarchar(50))')

Set @ParaCount =1
while @ParaCount <= @FeatureCount
begin
        set @FieldName= 'AvailabilityFeature'+@ParaCount
        EXEC('alter table TransactionDetails add '"'+@FieldName+'" real ')

        Set @ParaCount =@ParaCount+1
end
EXEC('alter table TransactionDetails add FeatureAvailability float ')
        EXEC('alter table TransactionDetails add AverageFeatureAvailabilitySoFar float ')

EXEC('alter table TransactionDetails add MonthlyCreditReturnFactor nvarchar(50) ')
EXEC('alter table TransactionDetails add CreditReturnSuccess int ')
EXEC('alter table TransactionDetails add CreditreturnFailedNumberSofar int ')
EXEC('alter table TransactionDetails add TrustValueOfARDS float ')
EXEC('alter table TransactionDetails add TrustOfAvailabilityPart float ')
EXEC('alter table TransactionDetails add TrustDeductionAmount float ')

EXEC('alter table TransactionDetails add NoOfCreditreturnRequiredSofar int ')
EXEC('alter table TransactionDetails add MonthlyAvailabilitypercentageSofar real ')
EXEC('alter table TransactionDetails add ExpectingMonthlyPercentageUptimeStatus nvarchar(max) ')
```

```
EXEC('alter table TransactionDetails add OptionCode nvarchar(max)  ')

EXEC('alter table TransactionDetails add Comments nvarchar(max) ')

        EXEC('alter table TransactionDetails add Availabilitypercentage real ')



set @TransactionCountSoFar = 1



declare Month_cursor cursor for select MonthName,ExpectingMonthlyPercentageUptimeStatus,NumberOfPeers from MonthlyTransaction order by id

open Month_cursor


fetch next from Month_cursor into @MonthName,@ExpectingMonthlyPercentageUptimeStatus,@NumberOfPeers

        set @Month_cursor_rowcount = @@FETCH_STATUS

        while @Month_cursor_rowcount = 0

        begin

                set @TotalPeersParticipatedPerMonth = @NumberOfPeers


                set @Count =1

                while @NumberOfPeers >0

                begin

                        set @FeatureAvailability = 0
```

```
insert into TransactionDetails (TransactionCountSoFar,MonthName,TotalPeersParticipatedPerMonth,PeerName)

values(@TransactionCountSoFar,@MonthName,@TotalPeersParticipatedPerMonth,'Peer'+CAST(@Count AS varchar))

set @PeerName = 'Peer'+CAST(@Count AS varchar)

Set @ParaCount =1

while @ParaCount <= @FeatureCount

begin

set @FieldName= 'AvailabilityFeature'+@ParaCount

set @FeatureValue = (select RAND()*(1-0)+0)

exec('update TransactionDetails set "'+@FieldName+'" = "'+@FeatureValue+'" WHERE MonthName ="'+@MonthName+'"and PeerName="'+@PeerName+"" )

if @ParaCount = 1

begin

set @FeatureAvailability = @FeatureAvailability +(@FeatureValue*@Gamma1)

end

else if @ParaCount=2

begin
```

```
                    set     @FeatureAvailability    =    @FeatureAvailability
+(@FeatureValue*@Gamma2)

                end

                else if @ParaCount=3

                begin

                    set     @FeatureAvailability    =    @FeatureAvailability
+(@FeatureValue*@Gamma3)

                end

                else if @ParaCount=4

                begin

                    set     @FeatureAvailability    =    @FeatureAvailability
+(@FeatureValue*@Gamma4)

                end

                Set @ParaCount =@ParaCount+1

            end

            if @FeatureAvailability < 0.9995

            begin

             set @MonthlyCreditReturnFactor ='Yes'

             set @CreditReturnSuccess =( select FLOOR(RAND()*(1-0+1))+0)

            end

            else

            begin

                set @MonthlyCreditReturnFactor ='No'

                set @CreditReturnSuccess = 1

            end
```

```
update TransactionDetails set FeatureAvailability =@FeatureAvailability,
MonthlyCreditReturnFactor = @MonthlyCreditReturnFactor,
CreditReturnSuccess=@CreditReturnSuccess where
TransactionCountSoFar=@TransactionCountSoFar and MonthName =@MonthName

set @CreditreturnFailedNumberSofar = (select
count(CreditReturnSuccess) from TransactionDetails where CreditReturnSuccess=0)

update TransactionDetails set CreditreturnFailedNumberSofar =
@CreditreturnFailedNumberSofar where TransactionCountSoFar=@TransactionCountSoFar and
MonthName =@MonthName

set @AverageFeatureAvailabilitySoFar = (select(
sum(FeatureAvailability)/ @TransactionCountSoFar )from TransactionDetails )

set @NoOfCreditreturnRequiredSofar = (select
count(MonthlyCreditReturnFactor) from TransactionDetails where MonthlyCreditReturnFactor ='Yes')

set @TrustOfAvailabilityPart = @Alpha
*(@AverageFeatureAvailabilitySoFar)

if @NoOfCreditreturnRequiredSofar >0

begin

set @TrustDeductionAmount = @Beta
*(CAST(@CreditreturnFailedNumberSofar AS float)/ CAST(@NoOfCreditreturnRequiredSofar AS float))

end

else begin
```

```
set @TrustDeductionAmount = 0

end

set @TrustValueOfARDS = @TrustOfAvailabilityPart - @TrustDeductionAmount

if @TrustValueOfARDS < 0

begin

set @TrustValueOfARDS = 0

end

update TransactionDetails set AverageFeatureAvailabilitySoFar = @AverageFeatureAvailabilitySoFar,

TrustOfAvailabilityPart= @TrustOfAvailabilityPart,

TrustValueOfARDS = @TrustValueOfARDS ,

TrustDeductionAmount =@TrustDeductionAmount,

NoOfCreditreturnRequiredSofar=@NoOfCreditreturnRequiredSofar

where TransactionCountSoFar=@TransactionCountSoFar

and MonthName =@MonthName

set @TransactionCountSoFar = (select (max(TransactionCountSoFar)+1) from TransactionDetails)

set @NumberOfPeers = @NumberOfPeers-1

set @Count = @Count+1
```

```
                end



                NextMonth:fetch next from Month_cursor into
@MonthName,@ExpectingMonthlyPercentageUptimeStatus,@NumberOfPeers

                set @Month_cursor_rowcount = @@FETCH_STATUS

                set @TotalPeersParticipatedPerMonth =@NumberOfPeers

            end


        close Month_cursor

        deallocate Month_cursor




         SELECT [TransactionCountSoFar]

    ,[MonthName]

    ,[PeerName]

    ,[AvailabilityFeature1] as 'AvailabilityOfFeature1'

    ,[AvailabilityFeature2] as 'AvailabilityOfFeature2'

    ,[AvailabilityFeature3] as  'AvailabilityOfFeature3'

          ,[AvailabilityFeature4] as  'AvailabilityOfFeature4'

    ,[FeatureAvailability]
```

```
      ,[MonthlyCreditReturnFactor]


      ,[CreditReturnSuccess]

      ,[CreditReturnFailedNumberSofar]

            ,[NoOfCreditreturnRequiredSofar]

      ,[TrustValueOfARDS]


       from TransactionDetails




end
```

A.7 2021ARDSTrust_Do_BinaryNorm

```
use [ARDS]

if exists (select 1 from sysobjects where type ='p'and name ='2021ARDSTrust_Do_BinaryNorm')

drop procedure [2021ARDSTrust_Do_BinaryNorm]

go

create proc [2021ARDSTrust_Do_BinaryNorm]


as

declare @BinaryTable nvarchar(50)

declare @MinTranId numeric(18,0)

declare @MaxTranId numeric(18,0)

declare @InputCount numeric(18,0)
```

```
declare @min as float

declare @max as float

                                        declare @OutputTable nvarchar(50)


                                        declare @FieldName nvarchar(50)

                                        declare @OutputCount numeric(30)

                                        declare @WindowCount numeric(30)

                                        declare @Count numeric(30)

                                        declare @FieldValue nvarchar(50)

                                        declare @WindowInputCount numeric(30)

                                        declare @RowCount numeric(30)

                                        declare @RowNumber numeric(30)

                                        declare @InputFieldName nvarchar(50)



Begin

        set @BinaryTable = 'BinaryOutput'



        if exists (select 1 from sys.Tables where type ='u'and name = 'BinaryOutput')

        begin

                EXEC ('drop table BinaryOutput' )
```

```
end



EXEC ('create table BinaryOutput(RowNumber Numeric(18,0),

TransactionCountSoFar numeric(18, 0) NULL,


MonthName nvarchar(50) NULL)')



EXEC ('alter table BinaryOutput add

[PeerName] [nvarchar](50) NULL,

[AvailabilityOfFeature1] [real]  NULL,

        [NormAvailabilityOfFeature1] [float]  NULL,

[AvailabilityOfFeature2] [real]  NULL,

        [NormAvailabilityOfFeature2] [float]  NULL,

[AvailabilityOfFeature3] [real]  NULL,

        [NormAvailabilityOfFeature3] [float]  NULL,

[AvailabilityOfFeature4] [real]  NULL,

        [NormAvailabilityOfFeature4] [float]  NULL,


[MonthlyCreditReturnFactor] [nvarchar](50) NULL,

        [NormMonthlyCreditReturnFactor] [float]  NULL,

[CreditReturnSuccess] [real] NULL,

        [NormCreditReturnSuccess] [float]  NULL,
```

```
[CreditreturnFailedNumberSofar] [real] NULL,

[NoOfCreditreturnRequiredSofar] [real] NULL,

[TrustValueOfARDS] [float]  NULL')

insert into  BinaryOutput (
RowNumber,
TransactionCountSoFar,
MonthName,
PeerName,
AvailabilityOfFeature1,
AvailabilityOfFeature2,
AvailabilityOfFeature3,
AvailabilityOfFeature4,

MonthlyCreditReturnFactor,
CreditReturnSuccess,
CreditreturnFailedNumberSofar,
NoOfCreditreturnRequiredSofar,
TrustValueOfARDS

)
select ROW_NUMBER()  OVER (ORDER BY TransactionCountSoFar),
```

```
TransactionCountSoFar,

MonthName,

PeerName,

AvailabilityFeature1,

AvailabilityFeature2,

AvailabilityFeature3,

AvailabilityFeature4,



MonthlyCreditReturnFactor,

CreditReturnSuccess,

CreditreturnFailedNumberSofar,

convert(float,NoOfCreditreturnRequiredSofar),

TrustValueOfARDS

from UsedARDSInteraction



----------NormAvailabilityOfFeature1----------

Set @min =( select min(AvailabilityOfFeature1) from BinaryOutput)

Set @max =(select max(AvailabilityOfFeature1) from BinaryOutput)



update BinaryOutput

set NormAvailabilityOfFeature1 = (AvailabilityOfFeature1 -@min)/(@max-@min)
```

```
-------------------------------------NormAvailabilityOfFeature2-------------------------------------

Set @min =( select min(AvailabilityOfFeature2) from BinaryOutput)

Set @max =(select max(AvailabilityOfFeature2) from BinaryOutput)




update BinaryOutput

set NormAvailabilityOfFeature2 = (AvailabilityOfFeature2 -@min)/(@max-@min)



-------------------------------------NormAvailabilityOfFeature3-------------------------------------

Set @min =( select min(AvailabilityOfFeature3) from BinaryOutput)

Set @max =(select max(AvailabilityOfFeature3) from BinaryOutput)




update BinaryOutput

set NormAvailabilityOfFeature3 = (AvailabilityOfFeature3 -@min)/(@max-@min)



-------------------------------------NormAvailabilityOfFeature4-------------------------------------

Set @min =( select min(AvailabilityOfFeature4) from BinaryOutput)

Set @max =(select max(AvailabilityOfFeature4) from BinaryOutput)
```

```
update BinaryOutput

set NormAvailabilityOfFeature4 = (AvailabilityOfFeature4 -@min)/(@max-@min)



----------NormMonthlyCreditReturnFactor----------

Set @min =( select min(MonthlyCreditReturnFactor) from BinaryOutput)

Set @max =(select max(MonthlyCreditReturnFactor) from BinaryOutput)



update BinaryOutput

set NormMonthlyCreditReturnFactor = (MonthlyCreditReturnFactor -@min)/(@max-@min)

----------NormCreditReturnSuccess----------

Set @min =( select min(CreditReturnSuccess) from BinaryOutput)

Set @max =(select max(CreditReturnSuccess) from BinaryOutput)
```

```
update BinaryOutput

set NormCreditReturnSuccess = (CreditReturnSuccess -@min)/(@max-@min)


select * from BinaryOutput



Select * into TempBinaryOutput from BinaryOutput          order by RowNumber ASC


ALTER TABLE TempBinaryOutput DROP COLUMN

TransactionCountSoFar,

MonthName,

PeerName,

AvailabilityOfFeature1,

AvailabilityOfFeature2,

AvailabilityOfFeature3,

AvailabilityOfFeature4,


MonthlyCreditReturnFactor,

CreditReturnSuccess,

CreditreturnFailedNumberSofar,

NoOfCreditreturnRequiredSofar
```

```
select * from TempBinaryOutput order by RowNumber ASC


if exists (select 1 from sys.Tables where type ='u'and name = 'UsedBinaryOutput')
        begin
                EXEC ('delete from UsedBinaryOutput' )
        end


insert into UsedBinaryOutput select * from TempBinaryOutput     order by RowNumber ASC


 DROP TABLE TempBinaryOutput


end
```

A.8 Python Program for ARDSTrustRBFNN

1. ARDSTrustRBFNN.py

```
import pyodbc
import csv
import pandas as pd
import os
import os.path

import datetime
import numpy as np
```

```
from numpy import loadtxt

import pandas as pd

import math

import shutil
import tensorflow

from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.layers import Activation, Dense, BatchNormalization,Input,Dropout
from tensorflow.keras.models import Sequential
#from tensorflow.keras.callbacks import LearningRateScheduler
import random as rn

from rbflayer import RBFLayer, InitCentersRandom
from kmeans_initializer import InitCentersKMeans
from initializer import InitFromFile

from matplotlib import pyplot

#setting seed value to 42 just before  building the RBFNN
seed_value= 42
os.environ['PYTHONHASHSEED']=str(seed_value)
np.random.seed(seed_value)
tensorflow.random.set_seed(seed_value)
```

```
rn.seed(None)


####################Set the input values

beta_min = 0

window_size_min = 1

hidden_nodes_min= 1


beta_max = 2

window_size_max =15

hidden_nodes_max= 200

learning_rates = [0.1, 0.01, 0.001, 0.0001, 0.00001, 0.000001]

hours_to_run= 3

epochs_value = 128


##########################################

Experiment_Number = 4470

Experiment_Number_Nan =174

##########################################


##Delete old mse_results.csv file and add new file with heading only and remove all old plyplots#################################################################


os.remove('mse_results.csv')#Delete old mse_results.csv

os.remove('mse_min.csv')#Delete old mse_min.csv

os.remove('BestARDSTrustRBFNNmodel_mse_results.csv')#Delete old mse_min.csv

os.remove('nan_mse_results.csv')#Delete old nan_mse_results.csv

shutil.rmtree('Pyplots/') # remove all old plyplots
```

```
shutil.rmtree('BestModelPyplots/') # remove all old BestModelPyplots


shutil.copyfile('NewFiles/mse_results.csv', 'mse_results.csv')#add new file with heading only

shutil.copyfile('NewFiles/mse_min.csv', 'mse_min.csv')#add new file with heading only

shutil.copyfile('NewFiles/BestARDSTrustRBFNNmodel_mse_results.csv',
'BestARDSTrustRBFNNmodel_mse_results.csv')#add new file with heading only

shutil.copyfile('NewFiles/nan_mse_results.csv', 'nan_mse_results.csv')#add new file with heading only

os.mkdir('Pyplots')#Creating new directory for plyplots.

os.mkdir('BestModelPyplots')#Creating new directory for BestModelPyplots.


############################################## Time loop startts here

end_time = datetime.datetime.now() + datetime.timedelta(hours=hours_to_run)


while end_time.timestamp() - datetime.datetime.now().timestamp() > 0:

  #randomnly select hyperparameters #########

  rn.seed(None)

  betas_value = rn.uniform(beta_min, beta_max)

  WindowSize = rn.randint(window_size_min, window_size_max)

  NumberOfHiddenNodes = rn.randint(hidden_nodes_min, hidden_nodes_max)

  #learning rate selection and applying it to compile process

  learning_rate = rn.choice(learning_rates)

  opt=keras.optimizers.RMSprop(learning_rate)


#Remove SQL cOMMENT

  ############################data gen################################################


                cnxn                =                pyodbc.connect('DRIVER={SQL
Server};SERVER=P25380249;DATABASE=ARDSWindows;Trusted_Connection=yes;')
```

```
    cursor = cnxn.cursor()

    #data=cursor.execute('Proc_test2 @Rowcount=?',(6))

     data=cursor.execute('update [WindowInitials] set StartingWindowSize=?,WindowCountIncrementValue
=?,NextWindowSize=?',(WindowSize,0,WindowSize))

    cnxn.commit()

    cursor.close()

    cnxn.close()


#REMOVE SQL COMMENT


    #seting seed value to 42 just before data generation with window size

    seed_value= 42

    os.environ['PYTHONHASHSEED']=str(seed_value)

    np.random.seed(seed_value)

    tensorflow.random.set_seed(seed_value)

    rn.seed(None)

    #checking if the window data is already created.

    if os.path.isfile('UsedWindowDataFiles/%s.csv'% str(WindowSize)):

        print ("Window size data file exist-")

        print(WindowSize)

            shutil.copyfile('UsedWindowDataFiles/%s.csv'%  str(WindowSize),'ARDSTrust2021.csv'  )#copying
from  backups


    else:

        print(WindowSize)

        print ("Window size file not exist")


        os.system("WindowInputData\Run_window_program.bat")
```

```
                                        cnxn = pyodbc.connect('DRIVER={SQL Server};SERVER=P25380249;DATABASE=ARDSWindows;Trusted_Connection=yes;')
    cursor = cnxn.cursor()
    cursor.execute('SELECT * FROM [dbo].[OutputWindow] order by [RowNumber] asc')

    data = cursor.fetchall()
    #print (data)
    with open('ARDSTrust2021.csv', 'w', newline='') as f_handle:
      writer = csv.writer(f_handle)

      for row in data:
          writer.writerow(row)
    cursor.close()
    cnxn.close()
     shutil.copyfile('ARDSTrust2021.csv', 'UsedWindowDataFiles/%s.csv'% str(WindowSize))#keeping a backup of all data

  ###############################################################################
  trust_df = pd.read_csv("ARDSTrust2021.csv")
  print(trust_df.describe())
  # load the dataset
   Trustdataset = loadtxt('ARDSTrust2021.csv', delimiter=',',skiprows=1)# skip first row as first row is headings.
  ###################################################
  #Set the NumPy random seed to 42
  #Shuffle the data before you split the data into a training, validation and test dataset.
  np.random.seed(42)
  np.random.shuffle(Trustdataset)
```

```
#############################################################################

   TotalNumberOfInputOutputFields = len( Trustdataset[0] )# Must reduce first column as first column is row count.

   TotalNumberOfInputFields = TotalNumberOfInputOutputFields - 1-1 #  Must remove first column as first column is row count and 1 outputcolumns.

   TotalNumberOfDataRows = len(list(Trustdataset))# Window 1 rows = 13727.

   TrainDataRowCountMax = int((TotalNumberOfDataRows*70)/100 )# 70% of total data is for training.

   ValDataRowCountMax = int((TotalNumberOfDataRows*90)/100 )# 20% of remaining 30 % data is for validating.

   print('TotalNumberOfInputFields =', TotalNumberOfInputFields)

   # split into input (X) and output (y) variables

   # 70% of total data is for training.

   # in WindowSize 1, 63 input 2 output total 66 columns in the file including first column(row count).

   X_train = Trustdataset[0:TrainDataRowCountMax,1:TotalNumberOfInputOutputFields-1]

         Y_train      =      Trustdataset[0:TrainDataRowCountMax,TotalNumberOfInputOutputFields-1:TotalNumberOfInputOutputFields]



   # 20% of remaining 30 % data is for validating.

                                                X_val                                                =  Trustdataset[TrainDataRowCountMax:ValDataRowCountMax,1:TotalNumberOfInputOutputFields-1]

                                                Y_val                                                =  Trustdataset[TrainDataRowCountMax:ValDataRowCountMax,TotalNumberOfInputOutputFields-1:TotalNumberOfInputOutputFields]



   # Remaining data is for testing.(10% of 30 % data is for testing.

                                                X_test                                               =  Trustdataset[ValDataRowCountMax:TotalNumberOfDataRows,1:TotalNumberOfInputOutputFields-1]

                                                Y_test                                               =  Trustdataset[ValDataRowCountMax:TotalNumberOfDataRows,TotalNumberOfInputOutputFields-1:TotalNumberOfInputOutputFields]

   #print(np.any(np.isnan(Trustdataset)))
```

```
# print(np.any(np.isnan(Y_test)))

# print("above")

TotalNumberOfOutputFields = 1


TrustRBFNNmodel = Sequential()


        TrustRBFLayer = RBFLayer(NumberOfHiddenNodes, betas=betas_value, input_shape=(TotalNumberOfInputFields,))

TrustRBFNNmodel.add(TrustRBFLayer)

TrustRBFNNmodel.add(Dropout(0.5))

OutputLayer = Dense(TotalNumberOfOutputFields, activation='sigmoid')

TrustRBFNNmodel.add(OutputLayer)


TrustRBFNNmodel.summary()

  ##############################


##############################################


# compile the keras model

TrustRBFNNmodel.compile(loss='mse', optimizer=opt, metrics=['mse'])


# fit the keras model (train)on the dataset

   TrustRBFNNmodelHistory = TrustRBFNNmodel.fit(X_train, Y_train, epochs=epochs_value , batch_size=10, validation_data=(X_val,Y_val))


# plot metrics

pyplot.plot(TrustRBFNNmodelHistory.history['mse'])

pyplot.plot(TrustRBFNNmodelHistory.history['val_mse'])
```

```
        # While you are searching for the best hyperparameters, you must evaluate your model on the validation set

        # Only after many experiments and after you have found the best hyperparameters, you evaluate the model on the test set



        # Determine validation MSE. This is the last training validation value

    validation_mse = TrustRBFNNmodelHistory.history['val_mse'][-1]



        # Save the MSE value to a csv file if it is not a nan

     print(validation_mse)

     if str(validation_mse) != "nan":

        output_file = open('mse_results.csv', 'a')

         text = str(Experiment_Number) + ', ' + str(NumberOfHiddenNodes) + ', ' + str(WindowSize) + ', ' + str(betas_value)+', '+str(learning_rate) + ','+ str(epochs_value)+ ',' + str(validation_mse)

        output_file.write(text + '\n')

        output_file.close()

                pyplot.savefig('Pyplots/%s.png'%    ("Experiment_Number-"+str(Experiment_Number)+"-Window-"+str(WindowSize)+"-Hiddennodes-"+str(NumberOfHiddenNodes)))

        Experiment_Number =Experiment_Number +1

     else :

        output_file = open('nan_mse_results.csv', 'a')

        text = str(Experiment_Number_Nan) + ', ' + str(NumberOfHiddenNodes) + ', ' + str(WindowSize) + ', ' + str(betas_value)+', '+str(learning_rate) + ','+ str(epochs_value)+ ',' + str(validation_mse)

        output_file.write(text + '\n')

        output_file.close()

        Experiment_Number_Nan =Experiment_Number_Nan +1

     print ((end_time.timestamp() - datetime.datetime.now().timestamp())/60)

# Finding minimum value of validation_mse
```

```
df = pd.read_csv("mse_results.csv")

row_count = df.shape[0]


min_validation_mse = df.at[0, df.columns.values[6]]


for i in range(0,row_count,1):
  if min_validation_mse > df.at[i, df.columns.values[6]]:
    min_validation_mse =df.at[i, df.columns.values[6]]


for i in range(0,row_count,1):
  if min_validation_mse == df.at[i, df.columns.values[6]]:
    min_validation_mse =df.at[i, df.columns.values[6]]
    Experiment_Number =df.at[i, df.columns.values[0]]
    NumberOfHiddenNodes=df.at[i, df.columns.values[1]]
    WindowSize=df.at[i, df.columns.values[2]]
    betas_value=df.at[i, df.columns.values[3]]
    learning_rate=df.at[i, df.columns.values[4]]
    epochs_value = df.at[i, df.columns.values[5]]
    output_file = open('mse_min.csv', 'a')
     text = str(Experiment_Number) + ', ' + str(NumberOfHiddenNodes) + ', ' + str(WindowSize) + ', ' +
str(betas_value)+', '+str(learning_rate) + ','+ str(epochs_value) +',' + str(min_validation_mse)
    output_file.write(text + '\n')
    output_file.close()


# Printing the details of experiments having minimum value of validation_mse

print("=================================================================")
```

```
print("Details of experiment having min_validation_mse : ",min_validation_mse)
print("================================================================================")
df = pd.read_csv("mse_min.csv")
row_count = df.shape[0]
for i in range(row_count):
  min_validation_mse =df.at[i, df.columns.values[6]]
  Experiment_Number = df.at[i, df.columns.values[0]]
  NumberOfHiddenNodes=df.at[i, df.columns.values[1]]
  WindowSize=df.at[i, df.columns.values[2]]
  betas_value=df.at[i, df.columns.values[3]]
  learning_rate=df.at[i, df.columns.values[4]]
  epochs_value = df.at[i, df.columns.values[5]]
  print("Experiment_Number =",Experiment_Number)
  print("NumberOfHiddenNodes =",NumberOfHiddenNodes)
  print("WindowSize =",WindowSize)
  print("betas_value =",betas_value)
  print("learning_rate =",learning_rate)
  print("epochs_valu =",epochs_value)
  print("min_validation_mse =",min_validation_mse)
  print("-----------------------------------")

#After training has stopped (after the 10 hours), do the following:
# Create the train, validate and test datasets by using the window size of the best model found.
# Train the best model found again on the training set.
#Evaluate this model on the test set and report the MSE on the training, validation and test sets.
##########SET opt using best model learning rate
```

```
opt=keras.optimizers.RMSprop(learning_rate)


# Create the train, validate and test datasets by using the window size of the best model found.

shutil.copyfile('UsedWindowDataFiles/%s.csv'% str(WindowSize),'ARDSTrust2021.csv' )#copying from  backups

Trustdataset = loadtxt('ARDSTrust2021.csv', delimiter=',',skiprows=1)# skip first row as first row is headings.


np.random.seed(42)

np.random.shuffle(Trustdataset)

#############################################################################

TotalNumberOfInputOutputFields = len( Trustdataset[0] )# Must reduce first column as first column is row count.

TotalNumberOfInputFields = TotalNumberOfInputOutputFields - 1-1 #  Must remove first column as first column is row count and 2 outputcolumns.

TotalNumberOfDataRows = len(list(Trustdataset))# Window 1 rows = 10000.

TrainDataRowCountMax = int((TotalNumberOfDataRows*70)/100 )# 70% of total data is for training.

ValDataRowCountMax = int((TotalNumberOfDataRows*90)/100 )# 20% of remaining 30 % data is for validating.

print('TotalNumberOfInputFields =', TotalNumberOfInputFields)

# split into input (X) and output (y) variables

# 70% of total data is for training.

# in WindowSize 1, 63 input 2 output total 66 columns in the file including first column(row count).

X_train = Trustdataset[0:TrainDataRowCountMax,1:TotalNumberOfInputOutputFields-1]

Y_train = Trustdataset[0:TrainDataRowCountMax,TotalNumberOfInputOutputFields-1:TotalNumberOfInputOutputFields]


# 20% of remaining 30 % data is for validating.

X_val = Trustdataset[TrainDataRowCountMax:ValDataRowCountMax,1:TotalNumberOfInputOutputFields-1]
```

```
Y_val = Trustdataset[TrainDataRowCountMax:ValDataRowCountMax,TotalNumberOfInputOutputFields-1:TotalNumberOfInputOutputFields]


# Remaining data is for testing.(10% of 30 % data is for testing.

X_test = Trustdataset[ValDataRowCountMax:TotalNumberOfDataRows,1:TotalNumberOfInputOutputFields-1]

Y_test = Trustdataset[ValDataRowCountMax:TotalNumberOfDataRows,TotalNumberOfInputOutputFields-1:TotalNumberOfInputOutputFields]


TotalNumberOfOutputFields = 1


BestTrustRBFNNmodel = Sequential()


TrustRBFLayer = RBFLayer(NumberOfHiddenNodes, betas=betas_value, input_shape=(TotalNumberOfInputFields,))

BestTrustRBFNNmodel.add(TrustRBFLayer)

BestTrustRBFNNmodel.add(Dropout(0.5))

OutputLayer = Dense(TotalNumberOfOutputFields, activation='sigmoid')

BestTrustRBFNNmodel.add(OutputLayer)


BestTrustRBFNNmodel.summary()


# compile the best model

BestTrustRBFNNmodel.compile(loss='mse', optimizer=opt, metrics=['mse'])


# fit the keras model (train)on the dataset

BestTrustRBFNNmodelHistory = BestTrustRBFNNmodel.fit(X_train, Y_train, epochs=epochs_value , batch_size=10, validation_data=(X_val,Y_val))
```

```
# plot metrics

pyplot.plot(BestTrustRBFNNmodelHistory.history['mse'])

pyplot.plot(BestTrustRBFNNmodelHistory.history['val_mse'])


#While you are searching for the best hyperparameters, you must evaluate your model on the validation set

# Only after many experiments and after you have found the best hyperparameters, you evaluate the model on the test set


# Determine validation MSE. This is the last training validation value

training_mse = BestTrustRBFNNmodelHistory.history['mse'][-1]

validation_mse = BestTrustRBFNNmodelHistory.history['val_mse'][-1]

pyplot.savefig('BestModelPyplots/%s.png'%                    ("Window-"+str(WindowSize)+"-Hiddennodes-"+str(NumberOfHiddenNodes)))


#Evaluate the model using test set

Test_DataSet_Evaluation = BestTrustRBFNNmodel.evaluate(X_test, Y_test,  batch_size=10)

Test_DataSet_mse = Test_DataSet_Evaluation[1]


#Evaluate the model using validation data set

Validation_DataSet_Evaluation = BestTrustRBFNNmodel.evaluate(X_val, Y_val,  batch_size=10)

Validation_DataSet_mse = Validation_DataSet_Evaluation[1]


# Save the MSE values of Best model  to a csv file

output_file = open('BestARDSTrustRBFNNmodel_mse_results.csv', 'a')

text =  str(NumberOfHiddenNodes) + ', ' + str(WindowSize) + ', ' + str(betas_value)+', '+str(learning_rate) + ','+ str(epochs_value)+ ','+ str(training_mse) + ','+str(validation_mse) +','+ str(Test_DataSet_mse) +','+ str(Validation_DataSet_mse)
```

```
output_file.write(text + '\n')

output_file.close()
```

APPENDIX B. PSTDG – ARDS MODEL VALIDATION DETAILS

1. What-if Scenario 1 (Expert knowledge 1)

1. 1 Compile

What is the effect on the **trust value** of the ARDS if the **satisfaction** (feedback regarding the percentage availability of features) **decreases**? The satisfaction for percentage availability of features starts at a maximum value of 1 and declines with a random value lying between 0 and 0.001 in a series of feedback submissions.

A series of feedback submissions from Peers (customers) will be done. The satisfaction level received from peers (customers) will be decreased with a random value between 0 and 0.001 for each submission. After each submission, the trust value of the ARDS will be calculated.

1. 2 Generate and plot

The following parameters will be fixed in this scenario.

- The total number of months: 1
- The total number of peers (customers): 1 000 (Peer1 to Peer1000)
- Trust context: availability and durability
- Feature count: 4
- Percentage uptime offered by the ARDS: 99.95%
- The weight factor for the first feature: 0.25
- The weight factor for the second feature: 0.25
- The weight factor for the third feature: 0.25
- The weight factor for the fifth feature: 0.25
- The weight factors α and β: 0.75 and 0.25, respectively

The new trust value of the ARDS will be plotted on the y-axis against the satisfaction (feedback regarding the percentage availability) received from the peers (customers) on the x-axis for a series of 1 000 transactions in Figure B-1.

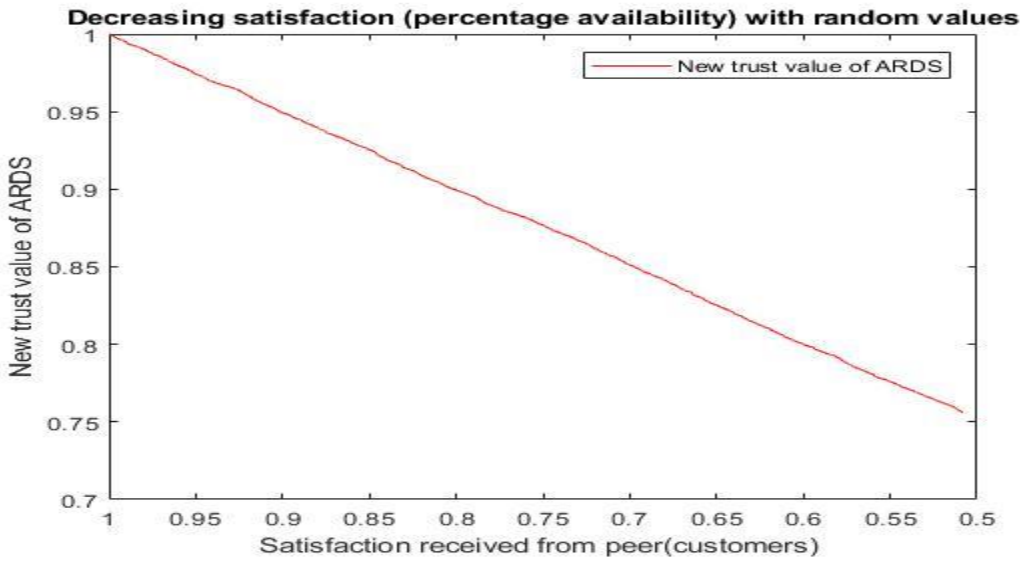


Figure B-1: Trust value of 1000 feedback when the satisfaction (percentage availability) received is decreasing with random values

The new trust value of the ARDS will be plotted on the y-axis against the satisfaction received from peers (customers) on the x-axis for a series of the first 25 transactions in Figure B-2.

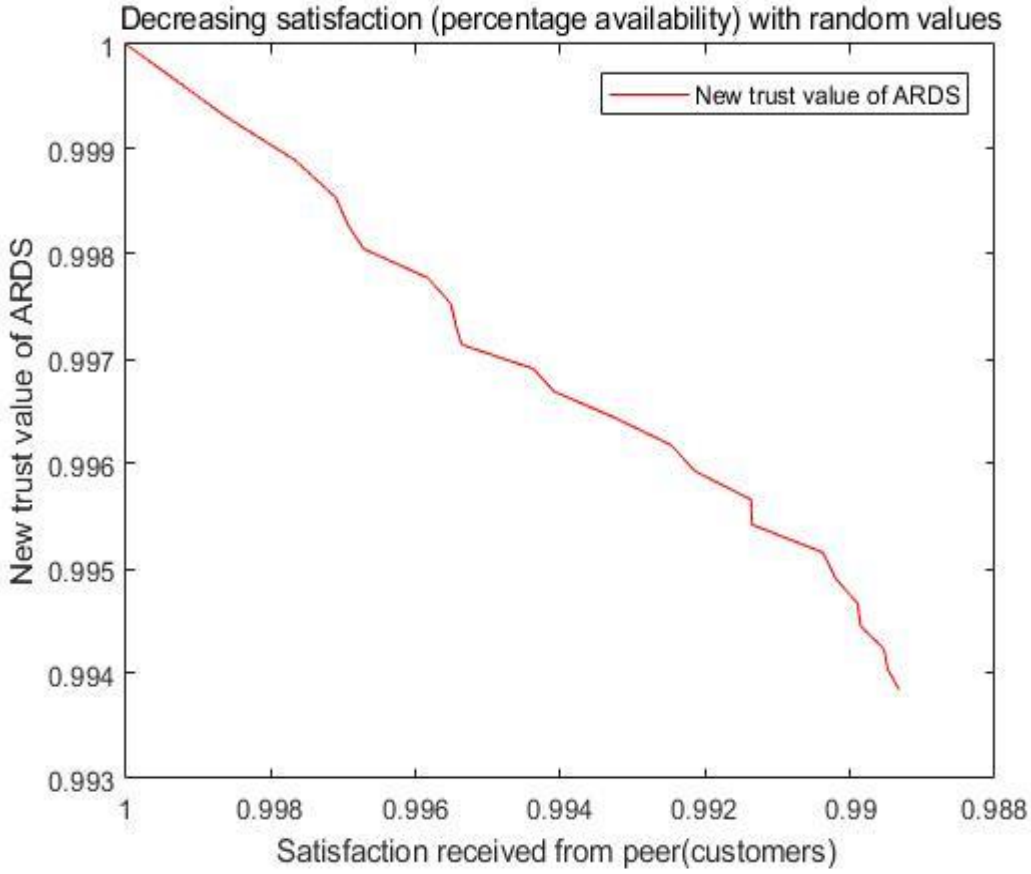


Figure B-2: Trust value of first 25 feedback when the satisfaction (percentage availability) received is decreasing with random values

1. 3 Analysis

Figure B-1 shows the graph of the new trust value of the ARDS during 1 000 transactions when the satisfaction (feedback regarding the percentage availability) received from the peers changes from maximum value (1) to a lower value, keeping all other parameters constant. Figure B-2 shows the graph of the new trust value of the ARDS during the first 25 transactions. The plot (Figure B-1) for the larger sample looks like a straight line but the plot (Figure B-2) for the smaller sample is not a straight line. This difference in shape is due to the visual scale. However, both graphs' trend is exactly as expected.

According to Figure B-1 and Figure B-2, the parameter 'satisfaction (feedback regarding the percentage availability) received from the peer (or customers)' has an impact on the new trust value of the ARDS, in such a way that the trust value of the ARDS drops as the 'satisfaction received from the peer' decreases. The plots show a very clear (nearly) linear relationship between the parameters 'satisfaction received from the peer' and 'new trust value of the ARDS'.

For other scenarios, plots for the larger sample will be plotted. Since the randomised change will not be visible in the plot for the larger sample, due to the scale problem, linear change in values will be used.

1. 4 Conclusion

The plots in Figure B-1 and Figure B-2 show the new trust value is directly proportional to the satisfaction received and therefore the data satisfies Expert knowledge 1.

2. What-if Scenario 2 (Expert knowledge 1)

2. 1 Compile

What will happen to the **trust valu**e of ARDS when the **satisfaction** (feedback regarding the percentage availability of features) received from peers (customers) **increases** linearly from a minimum value (0) to a maximum value (0.999) in a series of transactions?

After each transaction, the new trust value of the ARDS will be calculated when the satisfaction (feedback regarding the percentage availability) level received from the peers (customers) increases linearly and keeps all other parameters constant.

2. 2 Generate and plot

The following will be the fixed parameters in this scenario.

- The total number of months: 1
- The total number of peers (customers): 1 000 (Peer1 to Peer1000)
- Trust context: availability and durability
- Feature count: 4
- Percentage uptime offered by the ARDS: 99.95%
- The weight factor for the first feature: 0.25
- The weight factor for the second feature: 0.25
- The weight factor for the third feature: 0.25
- The weight factor for the fifth feature: 0.25
- The weight factors α and β: 0.75 and 0.25, respectively

The new trust value of the ARDS will be plotted on the y-axis against the satisfaction (feedback regarding the percentage availability) received from other peers on the x-axis for a series of 1 000 transactions in Figure B-3.

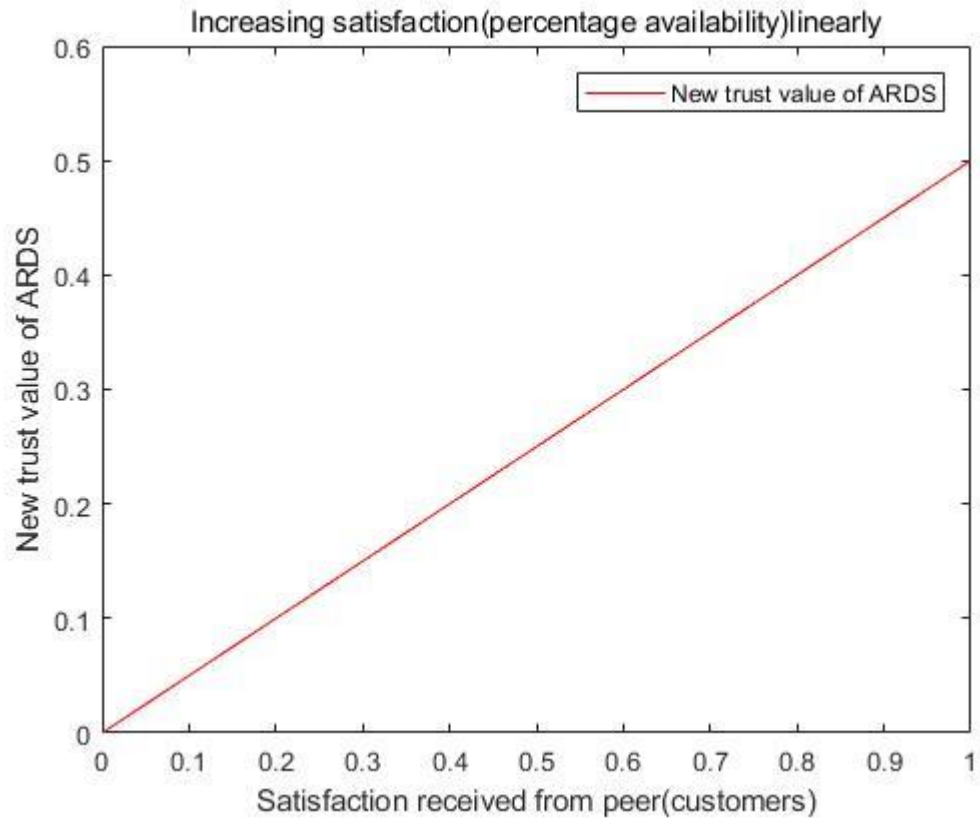


Figure B-3: Trust value when the satisfaction received is increasing linearly

2. 3 Analysis

Figure B-3 shows that the ARDS's trust value changes from 0 to 0.499 when 'satisfaction received from the peer' changes from 0 to 0.999. When the satisfaction received from the peer

is 0, the new trust value of the ARDS is 0. This is because the trust value part due to the availability context factor becomes 0 as the satisfaction received from the peer is 0. The plots show a very clear (nearly) linear relationship between the parameters ‘satisfaction received from the peer’ and ‘new trust value of the ARDS’ as the trust value of the ARDS increases when ‘satisfaction received from the peer’ increases.

2. 4 Conclusion

The plot in Figure B-3 shows the new trust value is directly proportional to the satisfaction (feedback regarding the percentage availability) received and therefore the data satisfies Expert knowledge 1.

3. What-if Scenario 3 (Expert knowledge 2)

3. 1 Compile

What will happen to the **trust value** of the ARDS when the ARDS receives feedback from other peers (or customers), but the trust value calculation **does not include all previous feedback** from all customers or peers?

After each feedback submission, the new trust value for the ARDS will be calculated. Here the ARDS will be receiving feedback from peers or customers in a series of 500 transactions.

The satisfaction (feedback regarding the percentage availability) received from the 1st peer (or customer) will be set to a minimum value of 0 with a failure in credit return success. The satisfaction (feedback regarding the percentage availability) received from the next 448 peers will be set to a maximum of value 1 each. The satisfaction (feedback regarding the percentage availability) received from the 500th peer (or customer) will also be set to a minimum value of 0 with a failure in credit return success. After each transaction, the trust value of the ARDS will be calculated, keeping all other parameters constant.

3. 2 Generate and plot

The following will be the fixed parameters in this scenario.

- The total number of months: 1
- The total number of peers (customers): 500 (Peer1 to Peer500)
- Trust context: availability and durability
- Feature count: 4
- Percentage uptime offered by the ARDS: 99.95%

- The weight factor for the first feature: 0.25
- The weight factor for the second feature: 0.25
- The weight factor for the third feature: 0.25
- The weight factor for the fifth feature: 0.25
- The weight factors α and β: 0.75 and 0.25, respectively

The feedback regarding the percentage availability of features of the ARDS will be plotted on the y-axis against the transaction count (feedback count) on the x-axis for the 500 transactions in Figure B-4.

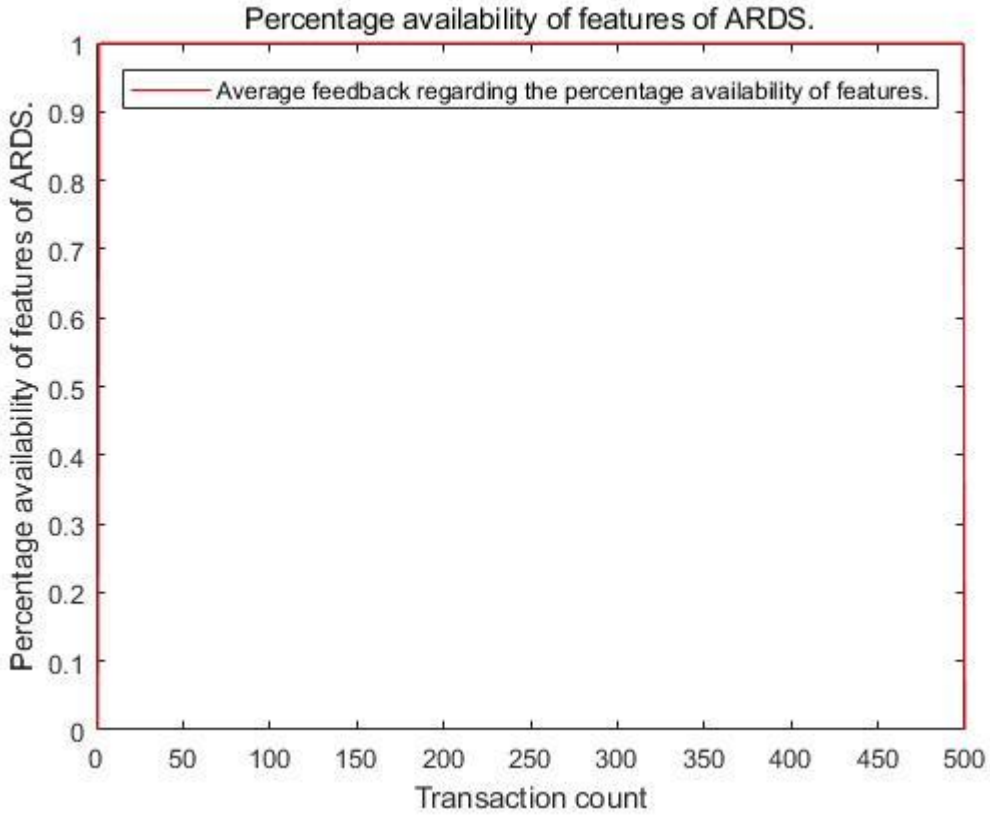


Figure B-4: Percentage availability of features of ARDS

The new trust value of the ARDS will be plotted on the y-axis against the series of transactions (feedback count) on the x-axis in Figure B-5.

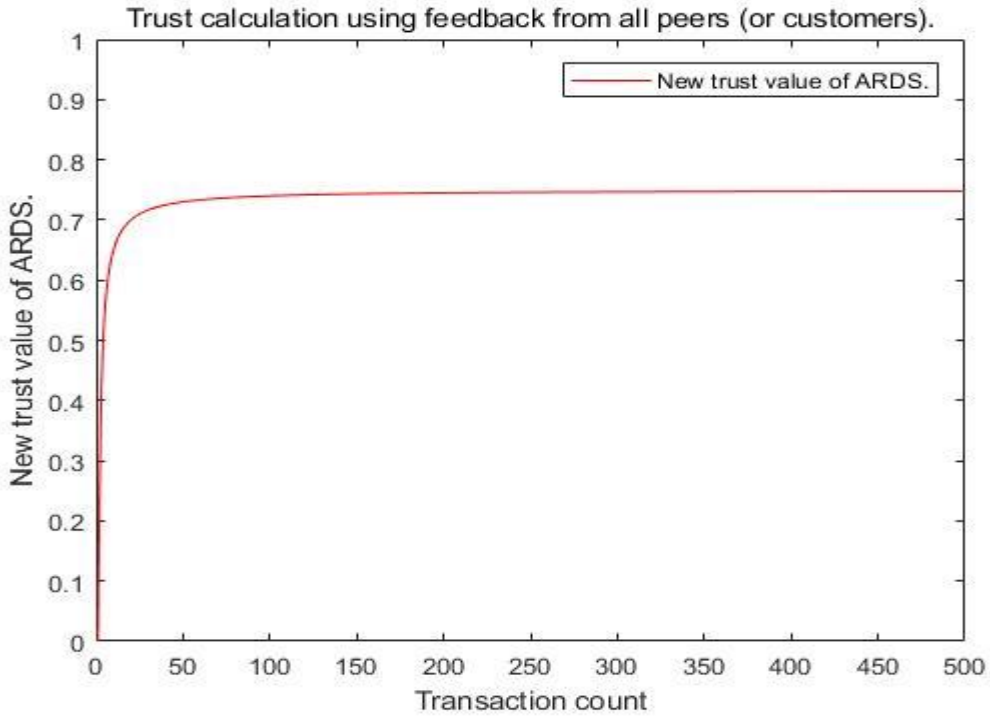


Figure B-5: Trust value of ARDS

3. 3 Analysis

The ARDS's trust value is 0 when the ARDS receives feedback regarding the percentage availability from the 1st peer (or customer). However, the ARDS's trust value becomes 0.746 in Figure B-5 when the ARDS receives satisfaction (feedback regarding the percentage availability) from the 500th peer (or customer). In both cases, the amount of satisfaction submitted was the same. The only difference was, in Case 2, the ARDS received feedback from 499 peers before receiving it from Peer500. The credit return success of Peer500 and Peer1 was a failure for both cases. The **satisfaction level** received from Peer500 and Peer1 was a very low value of 0 for both cases. However, the ARDS is having a high trust value in Case 2 and a lower trust value in Case 1. Figure B-5 also shows that the trust value of the ARDS is stabilising. This shows that in the long run, the ARDS has a high trust value, which remains stable when calculating the trust value. The trust value is calculated using all the feedback received from all peers or customers.

3. 4 Conclusion

Figure B-5 shows that the trust value is calculated using all the feedback received from all peers during the time. Hence the generated data satisfies the Expert knowledge 2.

4. What-if Scenario 4 (Expert knowledge 2)

4. 1 Compile

What will happen to the **trust value** of the ARDS when the ARDS receives feedback from other peers (or customers), but the trust value calculation **does not include feedback** regarding all features or sub-features?

The new trust value for the ARDS will be calculated after each feedback submission. Here the ARDS will be receiving feedback from peers or customers in a series of 1 000 transactions.

A series of feedback submissions from Peers (customers) will be done. The satisfaction level received from peers (customers) regarding Feature4 will be decreased with a random value between 0 and 0.001 for each submission. After each submission, the trust value of ARDS will be calculated.

During the first 250 feedback submissions, the satisfaction level received from peers (customers) regarding all four features will be a maximum value of 1. For the next 250 feedback submissions the satisfaction level received from peers (customers) regarding three features will be a maximum value of 1 and for the fourth feature will be 0. For the next 250 feedback submissions, the satisfaction level received from peers (customers) regarding all four features will again be a maximum value of 1. From 501 to 750th feedback the 4th feature will again be receiving a satisfaction level of 0. For the last 250 feedback submissions, all features will again receive a satisfaction level of 1.

After each feedback submission, the trust value of the ARDS will be calculated, keeping all other parameters constant.

4. 2 Generate and plot

The following will be the fixed parameters in this scenario.

- The total number of months: 1
- The total number of peers (customers): 1 000 (Peer1 to Peer1000)
- Trust context: availability and durability
- Feature count: 4
- Percentage uptime offered by the ARDS: 99.95%
- The weight factor for the first feature: 0.25
- The weight factor for the second feature: 0.25
- The weight factor for the third feature: 0.25

- The weight factor for the fifth feature: 0.25
- The weight factors α and β: 0.75 and 0.25, respectively

The feedback regarding the percentage availability of Feature4 will be plotted on the y-axis against the transaction count (feedback count) on the x-axis for the 1 000 transactions in Figure B-6.

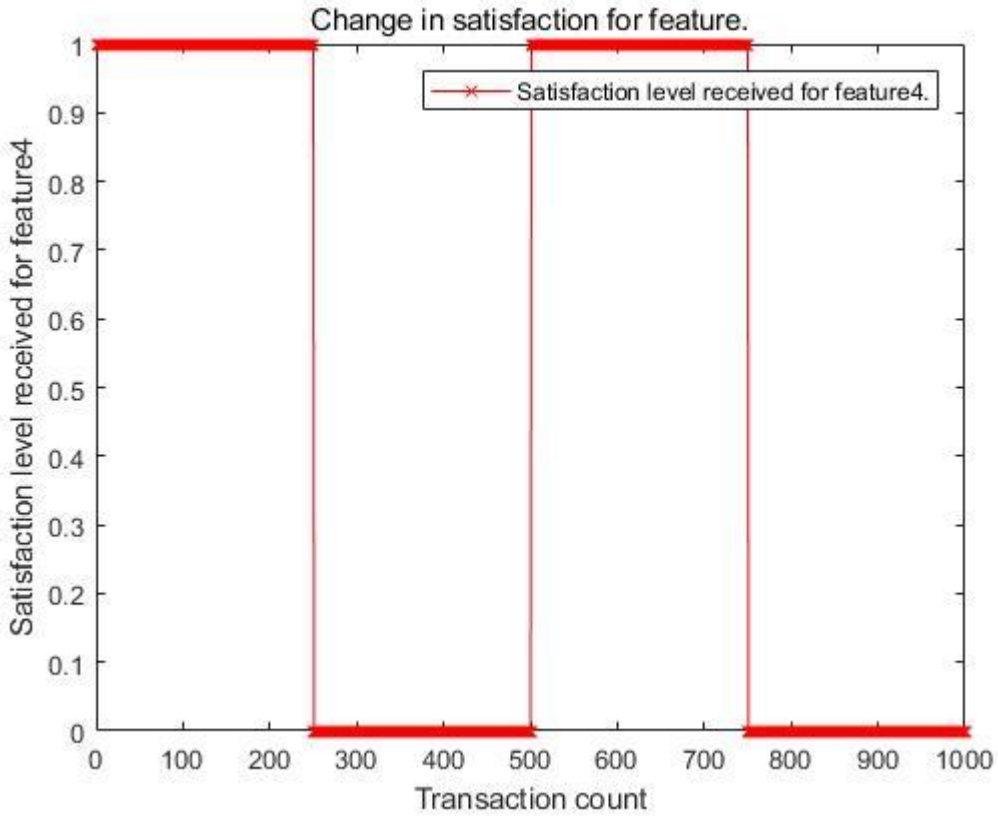


Figure B-6: Percentage availability of Feature4

The new trust value of the ARDS will be plotted on the y-axis against the series of transactions (feedback count) on the x-axis in Figure B-7.

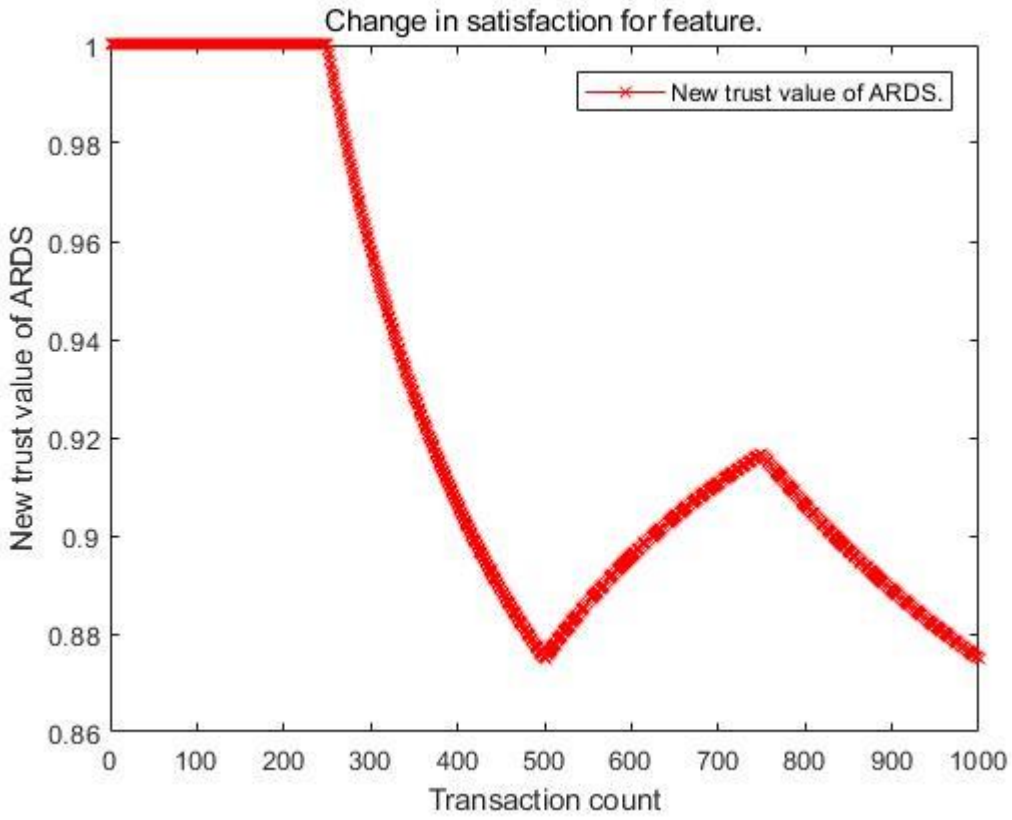


Figure B-7: Trust value of ARDS

4. 3 Analysis

The ARDS's trust value is 1 during the first 250 feedbacks. The trust value of the ARDS started decreasing during the next 250 feedbacks. During the next 250 feedbacks, there is an increase in the trust value of ARDS. During the last 250 feedbacks, the trust value of ARDS is again dropping. In all these cases, the only difference was in Cases 2 and 4, where the satisfaction received for Feature4 was decreasing. This shows that the trust value is calculated using all the feedback received for all features.

4. 4 Conclusion

Figure B-6 and Figure B-7 show that the trust value is calculated using all the feedback received for all features during the time. Hence the generated data satisfies the Expert knowledge 2.

5. What-if Scenario 5 (Expert knowledge 3)

5. 1 Compile

What will happen to the **trust value** of the ARDS when the ARDS **fails** to give **credit return** during a set of transactions, keeping all other parameters constant in a series of transactions?

The new trust value of the ARDS will be calculated after each feedback submission. There will be 1 000 transactions (or feedback submissions). During the first 250 feedback submissions, the ARDS will give credit return to the peers (or customers). For the next 250 feedback submissions, ARDS will not give credit return to the peers (or customers), then it will start giving credit return for the next 250 cases and for the last 250 cases, the ARDS will not give credit return to the peers (or customers), keeping all other parameters constant.

5. 2 Generate and plot

The following will be the fixed parameters in this scenario.

- The total number of months: 1
- The total number of peers (customers): 1 000 (Peer1 to Peer1000)
- Trust context: availability and durability
- Feature count: 4
- Percentage uptime offered by the ARDS: 99.95%
- The weight factor for the first feature: 0.25
- The weight factor for the second feature: 0.25
- The weight factor for the third feature: 0.25
- The weight factor for the fifth feature: 0.25
- The weight factors α and β: 0.75 and 0.25, respectively
- Satisfaction (feedback regarding the percentage availability): 0.99

In this scenario, two graphs will be plotted using the generated data set. The first graph is to visualise the change in credit return success status of the ARDS during each transaction (or feedback submissions) and the second graph is to visualise the change in trust value after each case. The transaction count will be on the x-axis for both graphs. A change in trust value due to the change in credit return success status (the ARDS fails to give credit return) can be analysed by comparing the two graphs.

The status of the credit return (whether given or not) given by the ARDS will be plotted on the y-axis against the series of transactions (transaction count) on the x-axis in Figure B-8.

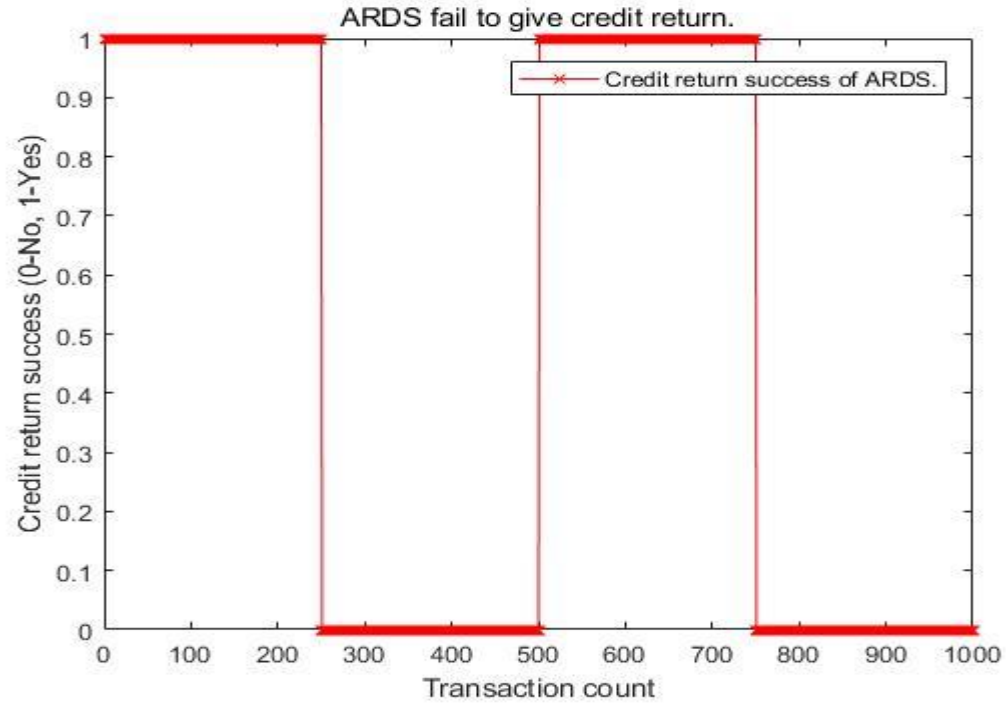


Figure B-8: ARDS fail to give credit return

The new trust value of the ARDS will be plotted on the y-axis against the series of transactions (transaction count) on the x-axis in Figure B-9.

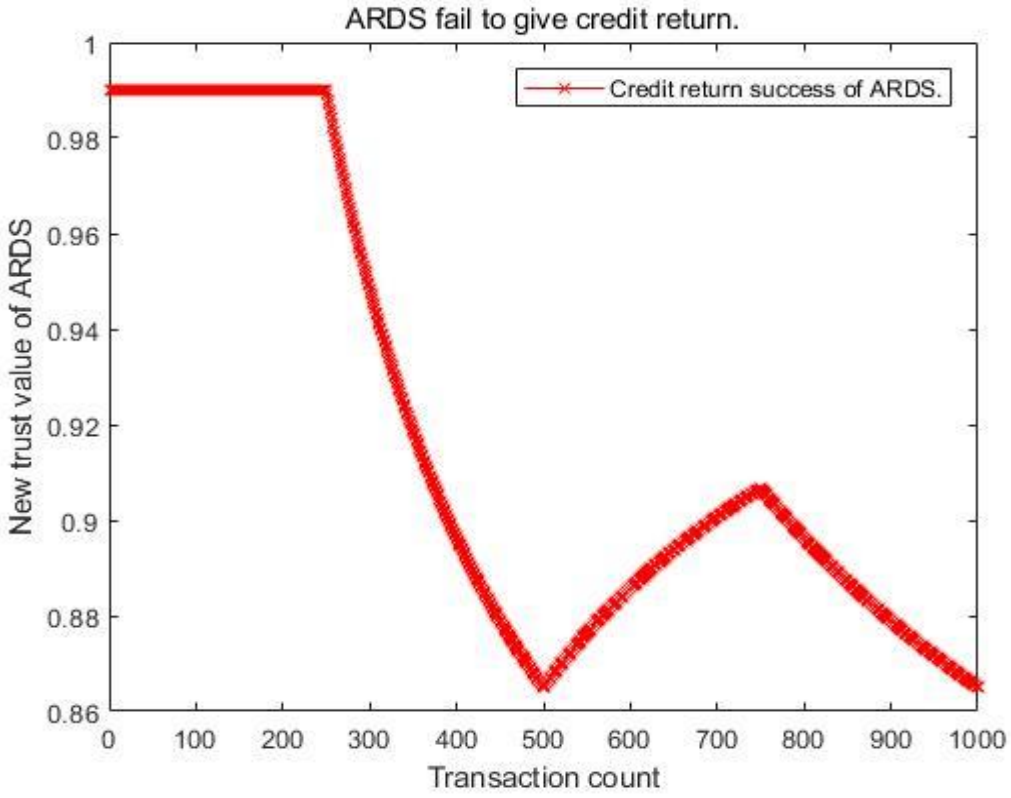


Figure B-9: Trust value when the ARDS fail to give credit return

5. 3 Analysis

Figure B-8 and Figure B-9 show what will happen when the ARDS fails to give credit return to the peers (or customers), keeping all other parameters constant. According to Figure B-8, the ARDS gave credit return from transaction count 1 to 250 and from transaction count 501 to 750. During all other cases, the ARDS failed to provide credit returns to the peers (or customers). Figure B-9 shows the ARDS's trust value decreases slowly from transaction count 251 to 500, then the trust value of the ARDS increases up to transaction count 750. The trust value of the ARDS is again declining slowly from transaction count 751 to 1 000. This shows that the ARDS's trust value depends upon the credit return success context factor. The amount of the reduction of trust value is increasing with every credit return failure.

5. 4 Conclusion

Figure B-8 and Figure B-9 show that the ARDS's trust value depends upon the community credit return success context factor. The generated data satisfies Expert knowledge 3.

6. Conclusion

The above graphs and their analysis give the following indications.

1. The trust value of the ARDS is directly proportional to the success of the MUP (Expert knowledge 1) according to the What-if Scenarios 1 and 2 and the conclusions in Section 1. 4 and Section 2. 4.
2. The trust value of the ARDS is calculated using all the feedback received regarding all the features from all peers or customers (Expert knowledge 2) according to the What-if Scenarios in Sections 3 and 4 and the conclusions in Sections 3. 4 and 4. 4.
3. As per the What-if Scenario in Section 5 and its conclusion in Section 5. 4, the trust value depends upon the success or failure of the credit return and the amount of the reduction of trust value is directly proportional to the failure of credit return (Expert knowledge 3).

The above indications show that the PSTDG-ARDS model generates a data set that satisfies all the expert knowledge identified in 7.2.2. The developed PSTDG-ARDS model can be used to generate a valid trust data set for creating the ARDSTrustRBFNN model.

APPENDIX C. PUBLISHED PEER-REVIEWED CONFERENCE ARTICLE

During this study, a peer-reviewed conference paper entitled *An Alternative Trust Calculation Method Using Radial Basis Function Neural Networks* (Zacaria *et al.*, 2022) was accepted, published and presented at the Southern Africa Telecommunication Networks and Applications Conference (SATNAC) 2022 held at George in the Western Cape, South Africa. This paper presents an alternative trust calculation method using an RBFNN which was discussed in this research.

An Alternative Trust Calculation Method Using Radial Basis Function Neural Networks

Smitha Zacaria[1], Tiny du Toit[2], Lucas Venter[3]

[1,2,3]*School of Computer Science and Information Systems*

North-West University, Potchefstroom Campus, South Africa

[1]25380249@nwu.ac.za

[2]Tiny.DuToit@nwu.ac.za

[3]Venter.Lucas@gmail.com

Abstract—**One of the most challenging problems in an electronic environment is the quantification of trust between electronic entities. Trust is a generic concept that can be effectively used in various contexts. However, most trust calculations are in the security context, and each electronic entity has its unique features and standards. The most important challenges identified in trust calculation are increased calculation complexity, calculation time, data storage, and access to large data sets for trust calculation. This study proposes an alternative trust calculation method using a radial basis function neural network (RBFNN) to solve some of these problems. An RBFNN is a feed-forward neural network used as a universal function approximator to solve nonlinear problems. Training an RBFNN to model trust values requires accurate and large data sets. Insufficient trust data sets were found due to privacy and data storage problems. Hence, a possible solution to trust data scarcity is synthetic data generation. However, inaccurate data can lead to an inaccurate RBFNN model to predict trust. Consequently, a seven-step framework named the Pure Synthetic Trust Data Generation Framework was developed to generate a valid and pure synthetic trust data set. A three-step process was developed to validate the trust data generation model. A four-step experimental design process was also introduced to build an RBFNN model to determine trust values between electronic entities. The positive results of this study on a theoretical and a real-world problem indicate that an alternative trust calculation method can be successfully developed using an RBFNN.**

Keywords—**Data generation, Electronic trust, Radial basis function neural network, RBFNN, Synthetic trust data, Trust.**

I. Introduction

In this digital age, trust is essential because electronic entities, primarily software entities, have a significant role in many parts of our lives. Day by day, electronic commerce, internet-based access to information, interpersonal communication through the internet, and peer-to-peer distributed computing are increasing. Businesses are moving from traditional platforms to digital platforms. Telecommunication networks are an integral part of this transformation. Hence electronic entity trust is more prevalent in this age. Trust is an essential factor in predicting the behaviour of any electronic or software system, as there will be concerns regarding the trustworthiness of each system. According to [1], interactions will be more reliable if based on trust.

Many researchers' definition of trust is relevant only in the context under which it is defined [2-5]. There is an assumption in these definitions that trust can be quantified somehow. Thus, trust can be calculated. However, no widely accepted definition of trust could be found in computer science and information technology. Therefore, there is a need for a definition of trust which can be used in general. Furthermore, most of the available trust calculations focus only on the information security context, whereas the specific trust value of an entity will be different in diverse contexts. Therefore, there is also a need to have a method to calculate trust in any context.

The remainder of the paper is structured as follows. Literature on trust, in general, is presented in Section II. In Section III, the structure, and favourable properties of a radial basis function neural network (RBFNN), which is proposed as an alternative trust calculation method, are discussed. Trust data generation is addressed in Section IV. In Section V, the experimental design is considered regarding developing a possible RBFNN model that can determine trust values between electronic entities. The results obtained from the experiments are addressed in Section VI. Finally, some concluding remarks are presented in Section VII.

II. Trust in General

Trust is context-dependent, and information security, service delivery, reliability, and credibility are examples of different trust contexts [2-5]. Some input information is required to calculate a trust value. The information needed can be obtained from various sources, such as recommendations, previous experience, feedback, and the presence of the World Wide Web Consortium (W3C) methods [2, 6-11]. Several existing trust models that quantify trust values can be found in the literature [12-17]. These models use different architectures for accessing information required for calculating and managing the trust values. The complexity of the trust value calculations increases as the complexity of the context increases. This complexity also increases when the complexity of the electronic entity networks increases. To calculate new trust values, some trust calculation algorithms use the previous trust value of the entity. Hence, all the previous transaction details and trust values of all entities must be stored and accessed each time to calculate the trust value of each entity concerning other entities. The fact that the trust value of an entity in one context, for example, the information security context, can be different in another context, for example, the service delivery context, also increases the complexity of the calculation. Another identified problem is the scarcity of real-world trust

data sets due to privacy concerns. Accordingly, there are several complications regarding the calculation of trust values and the management of trust data. Therefore, determining trust values can be complex and time-consuming due to information access, data storage, data management, network complexity, and context complexity factors.

Trust comprises three essential dimensions: the dimension of context, the dimension of calculation or quantification of the trust, and the dimension of information sources needed to perform the calculation. Based on these three dimensions, trust can be defined as a quantified belief or a probability of belief of an entity, which can be calculated by accessing or using information from different sources based on some set of standards or guidelines to have some desired property within a specified context.

The main aim of this study is to determine an alternative trust calculation method using an RBFNN that may solve some of the problems mentioned above. To achieve this goal, two objectives need to be reached as follows:

1) *Objective 1:* Data Generation

2) *Objective 2:* Construction of an RBFNN trust model

Once the first objective is successfully achieved, the problem of not having an extensive trust data set to use as training data will be solved. Therefore, constructing an RBFNN trust model will be the second objective. In the next section, the structure and properties of the proposed RBFNN model are considered.

III. RADIAL BASIS FUNCTION NEURAL NETWORKS

An RBFNN is a feed-forward neural network with a relatively simple three-layered structure [18]. The three layers are the input, hidden, and output layers [19]. RBFNNs have been widely used as a universal function approximator [18]. An RBFNN has several advantages, like a faster learning algorithm, support of incremental training and approximation capability that is higher than some other artificial neural networks [20-25]. In addition, an RBFNN performs a nonlinear transformation over the input vectors before they are classified. Hence, the RBFNN can convert a non-separable linear problem into a linearly separable problem.

Trust values and the number of parameters used to calculate trust depend on the context and the trust algorithm. The non-symmetric and non-transitive property of trust gives a broader context possibility for trust calculation, leading to the use of any number of parameters in the calculation of trust. Thus, trust can be calculated using any number of variables depending upon the context. The trust value will never be constantly increasing or decreasing as it can rely upon the level of past interactions. Since an RBFNN can solve complex pattern classification problems, it may be used in trust value classification. However, to train an RBFNN to calculate trust values between electronic entities in randomly varying situations requires large trust data sets. Synthetic trust data needs to be generated due to the scarcity of real-world trust data caused by privacy concerns, time in managing and accessing the trust data, and cost.

IV. TRUST DATA GENERATION

One solution to the scarcity of real-world training data is to generate synthetic data programmatically [26, 27]. A seven-step framework called the Pure Synthetic Trust Data Generation (PSTDG) Framework, as shown in Fig. 1, will be used to create pure synthetic data sets for trust calculation in this study. Consequently, the PSTDG Framework is a synthetic data generation lifecycle that provides a structured approach to generating valid and pure synthetic trust data sets. The PSTDG Framework includes the PSTDG model development, the PSTDG model validation, and the pure synthetic trust data generation using the validated PSTDG model. In Fig. 1, the four steps of the experimental design used in this study are also shown. This process will be addressed in Section V.

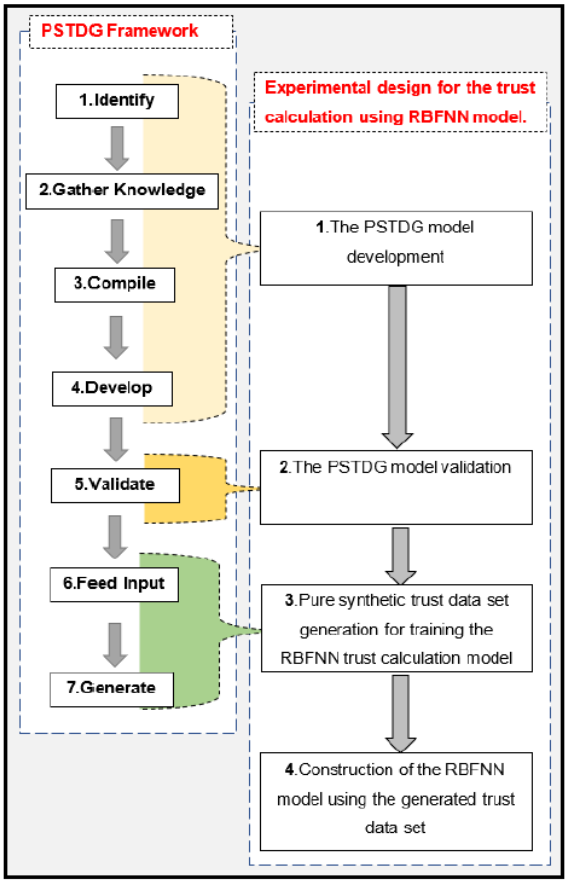


Figure 1: PSTDG Framework and experimental design of the RBFNN model for trust calculation

As a first step, the justification behind the pure synthetic trust data generation instead of using a real-life trust data set will be determined. All the knowledge, including expert knowledge regarding trust calculation, will be gathered as a second step. In the third step, constraints are compiled, which establish how the specialist expertise identified in the second step can be used to calculate the trust values. A set of rules or guidelines can be implemented as constraints in the trust data generation system. The trust data generation model or system

development will be performed in step four. An algorithm or program that can be used for the trust data generation process must be developed. Step five is added to validate the data generation model to determine whether the trust data generation model generates a valid trust data set. Validation can be defined as a process or method to provide evidence that can be used to determine whether the synthetic trust data generation model is generating a valid trust data set under its specific scenarios for its intended use. The steps below will be followed to conduct sensitivity and what-if analyses [28, 29] of the generated trust data set:

Step 1: A set of what-if scenarios is compiled using the available expert knowledge regarding the trust calculation.

Step 2: The trust data set is generated with scatter plots for each scenario.

Step 3: The plotted graphs are analysed to determine if the visual representation of the change (according to each scenario) in the generated trust data matches the expert knowledge.

If the plotted graphs affirm the expectation of the expert, the data generation model can be said to be valid for its intended use. Hence this process will be used as a validation method to show that the data generation model is valid according to the definition of validation given above.

The trust data generation model or system must be able to generate data according to specific situations or parameter values. Thus, the input parameter values that need to be used for the trust data set generation will be determined in the sixth step. The final step is where a trust data set will be generated. The constraints and restrictions or rules selected in the third step of the framework and the input parameter values provided in the sixth step will guide the generation of a synthetic trust data set. In the next section, the experimentation performed to determine if an RBFNN can be used as an alternative trust calculation method is presented.

V. EXPERIMENTAL DESIGN

As shown in Fig. 1, a four-step experimental design process was developed to build an RBFNN model to determine trust values between electronic entities. The PSTDG Framework forms the first three steps of this experimental design process. Two experiments were performed: the first used Li and Ling's PeerTrust theoretical trust calculation model [15], and the second was performed on a real-world problem. The purpose of the second experiment was to demonstrate that the proposed solution can also be applied to real-world problems. Hence, this experiment was done using the Amazon Relational Database Service (ARDS) [30].

The first four steps of the PSTDG Framework were followed to develop the two data generation models called *PSTDG-PeerTrust* and *PSTDG-ARDS*. First, both data generation models were created in Microsoft SQL. The two models were then validated using the definition of validation provided in Section IV. This validation was performed in the fifth step of the PSTDG Framework.

The first step in the validation process was the compilation of what-if scenarios regarding the calculation of trust values. A valid data set should satisfy all the identified expert knowledge. Hence all the what-if scenarios were compiled using the identified expert knowledge. According to Section IV, if the plotted graphs affirm the expectation of the identified expert knowledge, the data generation model can be valid for its intended use. Therefore, trust data with scatter plots were generated according to the compiled what-if scenarios. Then, the plotted graphs were analysed to determine if the generated trust data trends' visual representation matches the expert knowledge. An example of a what-if analysis of the PSTDG-PeerTrust model is given below.

Expert knowledge of Li and Ling's PeerTrust theoretical trust calculation model required that the new trust value of the interacting peer must depend upon the satisfaction received from the other peer after an interaction. Thus, the new trust value should be directly proportional to the satisfaction received. The model validation steps carried out were as follows:

Step 1: What is the effect on the trust value of Peer 1 if the satisfaction level received from all the other peers interacting with Peer 1 is randomly decreased? The satisfaction level starts at a maximum value of 1 and declines with a random value between 0 and 0.001 in a series of transactions. Consequently, a sequence of transactions between Peer 1 and the other peers was done. After each transaction, the trust value of Peer 1 was calculated.

Step 2: The following parameters was fixed in this scenario:

- The total number of transactions: 1000.
- The total number of peers: 1001 (Peer 1 to Peer 1001), with Peer 1 interacting with all the other peers.
- The transaction context value (transaction size value): 1.
- The initial trust value of all peers: 1.
- The weight factors α and β: 0.5.

The transaction size can be small, medium, large, or extra-large. A uniform random value inside the given ranges was assigned to each transaction according to the size. Extra-large transactions, for example, were assigned a random value between 0.75 and 1.0. The weight factors α and β are set to the same value, namely 0.5 so that all factors can contribute equally to the trust value.

The new trust value of the interacting peer, Peer 1, was plotted on the y-axis against the satisfaction received from the other peers on the x-axis for 1000 transactions in Fig. 2. The new trust value of the interacting peer, Peer 1, was also plotted on the y-axis against the satisfaction received from the other peers on the x-axis for a series of the first 25 transactions in Fig. 3.

Step 3: Fig. 2 shows the new trust value of the interacting peer during 1000 transactions when the satisfaction received from the other peers' changes from the maximum value (1) to a lower value keeping all other parameters constant. In Fig. 3, the new trust value of the interacting peer during the first 25 transactions is shown. The plot (Fig. 2) for the larger sample looks like a straight line, but the plot (Fig. 3) for the smaller sample is not a straight line. This difference in shape is due to the visual scale. However, both graphs' trend is precisely as expected.

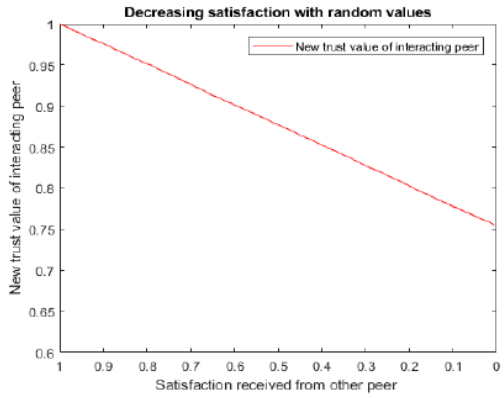


Figure 2: Trust value of 1000 transactions when the satisfaction received is decreasing with random values

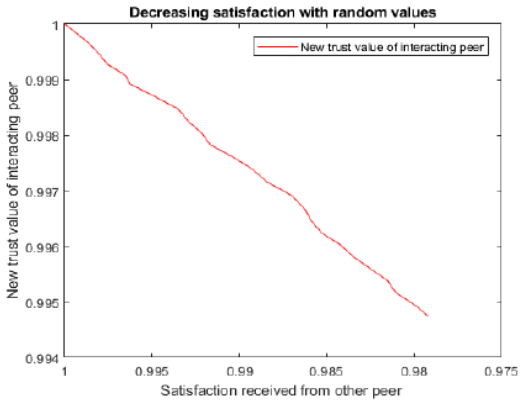


Figure 3: Trust value of first 25 transactions when the satisfaction received is decreasing with random values

According to Fig. 2 and Fig. 3, the satisfaction received from the other peer parameter impacts the new trust value of the interacting peer in such a way that the trust value of the interacting peer drops as the satisfaction received from the other peer decreases. The plots show a clear (nearly) linear relationship between the satisfaction received from the other peer and the new trust value of the interacting peer parameters. The Fig. 2 and Fig. 3 plots show that the new trust value is directly proportional to the satisfaction received. Therefore, the data satisfies the identified expert knowledge given above.

The synthetic trust data set generation was done in the sixth and seventh steps of the PSTDG Framework. In total, 10000 samples were generated by the PSTDG-PeerTrust model consisting of 14 input variables, while the PSTDG-ARDS model generated 13727 samples having 13 inputs. The 14 input variables generated by the PSTDG-PeerTrust model included the name of the interacting peer, the name of the other peer participating in the interaction, the size of the transaction, the transaction size value, an indication of whether the interacting peer received feedback from the other peer, the amount of satisfaction received from the other peer, an indication whether the other peer received feedback from the interacting peer, the amount of satisfaction received from the interacting peer, the current trust value of the interacting peer, the existing trust value of the other peer, the total number of feedback the interacting peer has given, the total number of feedback the other peer has given, the new trust value of the interacting peer after the transaction, and the new trust value of the other peer. In the PSTDG-ARDS model, the inputs are the number of transactions that happened so far, the name of the month, the name of the peer, the amount of satisfaction (percentage availability or percentage success regarding the first feature) received from peers or customers, the amount of satisfaction (percentage availability or percentage success regarding the second feature) received from peers or customers, the amount of satisfaction (percentage availability or percentage success regarding the third feature) received from peers or customers, the amount of satisfaction (percentage availability or percentage success regarding the fourth feature) received from peers or customers, the amount of satisfaction (percentage availability or percentage success regarding all of the four features) calculated using four weight factors, an indication whether the peer or customer is eligible for credit return from the ARDS, an indication whether the ARDS succeeded in credit return to eligible peers or customers, an indication of the total number of failures that happened in credit return so far, the total number of credit returns required so far and the new trust value of the ARDS after the transaction or submission of monthly feedback from each peer or customer.

The data were modelled as two time series using sliding windows. After the two synthetic trust data sets were developed, the two proposed RBFNN models were built. This process included data pre-processing, identifying the best RBFNN models, and evaluating the best models. Data pre-processing can improve the accuracy of an artificial neural network model, decrease the computational cost, and accelerate the learning process [31-33]. In addition, the pre-processing of the input variables helps to better match the predicted output [32, 33]. Some inputs were converted to one-hot encoded binary values as an artificial neural network can work only with numerical data. Other inputs were normalised to the range [0, 1] as the data must lie in the same range for the artificial neural network to treat the values equally [33]. After the pre-processing was performed, both synthetic trust data sets were randomly partitioned into training (70%), validation (20%) and test sets (10%).

To build an accurate RBFNN model that can determine trust values between electronic entities, suitable model hyperparameters were chosen to construct the two models. The hyperparameter optimisation system was developed in the Python 3.9.5 programming language with the TensorFlow 2.5 library and Keras 2.5 application program interface. Suitable model hyperparameters were chosen by training several candidate RBFNN models using uniform randomly sampled hyperparameters and selecting the best model. The random search performed is a widely used and efficient method for hyperparameter optimisation [34-36]. All experiments were performed on an Intel® Core™ i7-8550U CPU running at 1.80GHz with 8.00 GB of RAM.

A hyperparameter search space was defined before searching for the best hyperparameters. In TABLE I, the hyperparameter search space bounds for the PeerTrustRBFNN and ARDSTrustRBFNN models are shown. These bounds were determined by performing preliminary experiments.

TABLE I

HYPERPARAMETER SEARCH BOUNDS FOR THE PeerTrustRBFNN AND ARDSTrustRBFNN MODELS

Hyperparameter	Minimum value	Maximum value
β (bias of output layer)	0.0	2.0
Hidden nodes	1	200
Time series sliding window size	1	15
Learning rate	$10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}$, or 10^{-1}	

The search for the best hyperparameters was then performed for 24 hours, during which 162 candidate PeerTrustRBFNN models were evaluated and for 250 hours, during which 4531 candidate ARDSTrustRBFNN models were assessed. The difference in hyperparameter optimisation time of the two experiments was due to a good PeerTrustRBFNN model found early in the first experiment. The final evaluation of the RBFNN models was done by determining the mean square error (MSE) value on the test sets, where a value closer to zero denotes better model performance [37]. The results obtained from the two experiments are discussed in the next section.

VI. RESULTS AND DISCUSSION

In the first experiment, the focus during the development of the PeerTrustRBFNN model was mainly on validating the trust data generation model called the PSTDG-PeerTrust model. This was done because the PeerTrust model by Li and Ling [15] is a theoretical trust calculation model, and they provided the trust calculation equation. Even though the equation was available, the developed data generation model did not need to produce a valid trust data set as errors can happen in the model development process. Using what-if analysis and sensitivity analysis, the validation of the PSTDG-PeerTrust model was done using the definition provided in Section IV. Having a large trust data set was the most important challenge for the training of an RBFNN and was solved by developing the PSTDG-PeerTrust model. The hyperparameters of the best PeerTrustRBFNN model found are shown in TABLE II. The model trained for 100 epochs and had an MSE of $2.58 \cdot 10^{-4}$.

TABLE II

HYPERPARAMETERS OF THE BEST PeerTrustRBFNN MODEL

Hyperparameter	Value
β	$7.13 \cdot 10^{-2}$
Hidden nodes	72
Sliding window size	1
Learning rate	10^{-3}

In the second experiment, the pure synthetic trust data set generation model called the PSTDG-ARDS model was developed using the ARDS to show that the proposed solution can be applied to real-life problems. The focus was on model development and validation since no algorithm or equations were available to perform a trust calculation for the ARDS. Thus, it was necessary to show how expert knowledge can be obtained and how trust values can be calculated. The PSTDG-ARDS model validation was performed to ensure that the model can satisfy the identified expert knowledge. Determining the validity of the pure synthetic trust data set was done by validating the PSTDG-ARDS model. The validation of the PSTDG-ARDS model was done by what-if analysis and sensitivity analysis using the definition given in Section IV. The development of the PSTDG-ARDS model solved the problem of not having a large data set for the training of an RBFNN model to calculate trust values for the ARDS. After the trust data set was constructed, the best RBFNN model found had the hyperparameter values shown in TABLE III.

TABLE III

HYPERPARAMETERS OF THE BEST ARDSTrustRBFNN MODEL

Hyperparameter	Value
β	$7.71 \cdot 10^{-1}$
Hidden nodes	126
Time series sliding window size	14
Learning rate	10^{-4}

This model trained for 128 epochs and had an MSE value of $7.72 \cdot 10^{-6}$.

VII. CONCLUSION

Trust is a generic concept, and one of the most challenging problems in an electronic environment is quantifying trust between electronic entities. This study proposes an alternative trust calculation method using an RBFNN to solve some of the identified problems like increased calculation complexity, calculation time, data storage and access to large data sets for trust calculation. The scarcity of trust data is solved by synthetic trust data generation using the PSTDG Framework. This framework can be used to create a valid and pure synthetic trust data set. The three-step process developed to validate the trust data generation model enabled validation of the two developed models, namely the PSTDG-PeerTrust and PSTDG-ARDS. The four-step experimental design process proposed to build an RBFNN model to determine trust values between electronic entities was successfully demonstrated using two experiments. There is no need for large data storage and access by an RBFNN to calculate the trust values, reducing the trust calculation time. Trust calculation using an RBFNN will reduce the calculation complexity. Network complexity will never affect the trust calculation complexity if using an RBFNN. Using an RBFNN to calculate trust and an increased number of parameters to quantify trust will also not increase the complexity of the trust calculation. Finally, this study shows that an alternative trust calculation method can be successfully developed using an RBFNN.

REFERENCES

[1] Z. Yong-sheng and W. Ying, "Research on Trust-Authorization-Based Access Control Model for Web Services," in *Networks Security Wireless Communications and Trusted Computing (NSWCTC), 2010 Second International Conference on*, 24-25

April 2010 2010, vol. 1, pp. 454-457, doi: 10.1109/NSWCTC.2010.113.

[2] D. Gambetta, "Can We Trust Trust?," *Trust: Making and Breaking Cooperative Relations*Department of Sociology, University of Oxford: Blackwell, 1988, pp. 213-237.

[3] T. Grandison and M. Sloman, "Specifying and Analysing Trust for Internet Applications," presented at the Towards the Knowledge Society: eCommerce, eBusiness and eGovernment The Second IFIP Conference on E-Commerce, E-Business, E-Government (I3E 2002) Lisbon, Portugal, Oct 7–9, 2002.

[4] T. Grandison and M. Sloman, "A survey of trust in internet applications," *IEEE Communications Surveys & Tutorials,* vol. 3, no. 4, pp. 2-16, 2002, doi: 10.1109/COMST.2000.5340804.

[5] D. Olmedilla, O. F. Rana, B. Matthews, and W. Nejdl, "Security and trust issues in semantic grids," in *Dagstuhl Seminar Proceedings*, 2006: Schloss Dagstuhl-Leibniz-Zentrum für Informatik).

[6] D. Denning, "A New Paradigm for Trusted Systems," pp. 36-41, 01/01 1993, doi: 10.1145/283751.283772.

[7] L. Mui, M. Mohtashemi, and A. Halberstadt, "A computational model of trust and reputation," presented at the Proceedings of the 35th Annual Hawaii International Conference on System Sciences, Big Island, HI, USA 10-10 Jan, 2002.

[8] Y. Wang and J. Vassileva, "Trust and reputation model in peer-to-peer networks," in *Peer-to-Peer Computing, 2003. (P2P 2003). Proceedings. Third International Conference on Peer-to-Peer Computing* Linkoping, Sweden 1-3 Sept. 2003 2003: IEEE pp. 150-157, doi: 10.1109/PTP.2003.1231515.

[9] M. Witkowski and J. Pitt, "Objective trust-based agents: Trust and trustworthiness in a multi-agent trading society," in *MultiAgent Systems, 2000. Proceedings. Fourth International Conference on MultiAgent Systems*, Boston, MA, USA, 10-12 July 2000 2000: IEEE pp. 463-464, doi: 10.1109/ICMAS.2000.858526.

[10] C. W. Hang, A. K. Kalia, and M. P. Singh, "Behind the Curtain: Service Selection via Trust in Composite Services," presented at the 2012 IEEE 19th International Conference on Web Services, Honolulu, HI, USA 24-29 Jun., 2012.

[11] R. Au, M. Looi, and P. Ashley, "Automated cross-organisational trust establishment on extranets," in *Proceedings Workshop on Information Technology for Virtual Enterprises. ITVE 2001*, Gold Coast, Queensland, Australia, Australia 29-30 Jan. 2001 2001, pp. 3-11, doi: 10.1109/ITVE.2001.904483.

[12] K. Aberer and Z. Despotovic, "Managing trust in a peer-2-peer information system," presented at the Proceedings of the tenth international conference on Information and knowledge management, Atlanta, Georgia, USA, 2001.

[13] S. Daskapan, I. Nurtanti, and J. v. d. Berg, "Improving trust valuation for file sharing in P2P networks," presented at the 2008 IEEE International Conference on Systems, Man and Cybernetics, Singapore, 12-15 Oct, 2008.

[14] X. Li and L. Ling, "Building trust in decentralized peer-to-peer electronic communities," in *Fifth International Conference on Electronic Commerce Research (ICECR-5)*, Montreal, Canada, 23-27 October 2002, pp. 1-15.

[15] X. Li and L. Ling, "PeerTrust: supporting reputation-based trust for peer-to-peer electronic communities," *IEEE Transactions on Knowledge and Data Engineering,* vol. 16, no. 7, pp. 843-857, 2004, doi: 10.1109/TKDE.2004.1318566.

[16] H. Singal and S. Kohli, "Trust Necessitated through Metrics: Estimating the Trustworthiness of Websites," *Procedia Computer Science,* Article vol. 85, pp. 133-140, 1/1/2016 2016, doi: 10.1016/j.procs.2016.05.199.

[17] U. E. Tahta, S. Sen, and A. B. Can, "GenTrust: A genetic trust management model for peer-to-peer systems," *Applied Soft Computing,* vol. 34, pp. 693-704, 9// 2015, doi: http://dx.doi.org/10.1016/j.asoc.2015.04.053.

[18] F. A. Ruslan, A. M. Samad, Z. M. Zain, and R. Adnan, "Modelling flood prediction using Radial Basis Function Neural Network (RBFNN) and inverse model: A comparative study," in *2013 IEEE International Conference on Control System, Computing and Engineering*, 29 Nov.-1 Dec. 2013 2013, pp. 577-581, doi: 10.1109/ICCSCE.2013.6720031.

[19] M. Khazaei, H. Sadat-Hosseini, A. Marjaninejad, and S. Daneshvar, "A Radial Basis Function Neural Network approximator with fast terminal sliding mode-based learning algorithm and its application in control systems," in *2017 Iranian Conference on Electrical Engineering (ICEE)*, 2-4 May 2017 2017, pp. 812-816, doi: 10.1109/IranianCEE.2017.7985150.

[20] S. Chen, C. F. N. Cowan, and P. M. Grant, "Orthogonal least squares learning algorithm for radial basis function networks," *IEEE Transactions on Neural Networks,* vol. 2, no. 2, pp. 302-309, 1991, doi: 10.1109/72.80341.

[21] S. Qasem, S. M. Shamsuddin, and A. Zain, "Multi-objective hybrid evolutionary algorithms for radial basis function neural network design," vol. 27, pp. 475–497, March 2013, doi: 10.1016/j.knosys.2011.10.001.

[22] G.-F. Lin and M.-C. Wu, "An RBF network with a two-step learning algorithm for developing a reservoir inflow forecasting model," *Journal of Hydrology,* vol. 405, no. 3, pp. 439-450, 2011/08/05/ 2011, doi: https://doi.org/10.1016/j.jhydrol.2011.05.042.

[23] H. Yu, T. Xie, S. Paszczynski, and B. M. Wilamowski, "Advantages of Radial Basis Function Networks for Dynamic System Design," *IEEE Transactions on Industrial Electronics,* vol. 58, no. 12, pp. 5438-5450, 2011, doi: 10.1109/TIE.2011.2164773.

[24] J. Park and I. W. Sandberg, "Universal Approximation Using Radial-Basis-Function Networks," *Neural Computation,* vol. 3, no. 2, pp. 246-257, 1991, doi: 10.1162/neco.1991.3.2.246.

[25] R. Azmi, M. Hakimi, and Z. Bahmani, *Dynamic Reputation Based Trust Management Using Neural Network Approach*. 2011.

[26] J. W. Anderson, K. E. Kennedy, L. B. Ngo, A. Luckow, and A. W. Apon, "Synthetic data generation for the internet of things," in *2014 IEEE International Conference on Big Data (Big Data)*, 27-30 Oct. 2014 2014, pp. 171-176, doi: 10.1109/BigData.2014.7004228.

[27] J. Weston *et al.*, *Towards AI-Complete Question Answering: A Set of Prerequisite Toy Tasks.Under review as a conference paper at ICLR*. 2015.

[28] J. P. C. Kleijnen, "Sensitivity analysis and related analyses: A review of some statistical techniques," *Journal of Statistical Computation and Simulation,* vol. 57, no. 1-4, pp. 111-142, 1997/04/01 1997, doi: 10.1080/00949659708811805.

[29] Y. Chan, C. D. Correa, and K. Ma, "Flow-based scatterplots for sensitivity analysis," in *2010 IEEE Symposium on Visual Analytics Science and Technology*, 25-26 Oct. 2010 2010, pp. 43-50, doi: 10.1109/VAST.2010.5652460.

[30] Amazon. "Amazon RDS Service Level Agreement." https://aws.amazon.com/rds/sla/ (accessed September 23, 2011).

[31] N. Mohd Nawi, W. Atomi, and S. M. Rehman Gillani, *The Effect of Data Pre-processing on Optimized Training of Artificial Neural Networks*. 2013.

[32] K. Kuźniar and M. Zając, "Some methods of pre-processing input data for neural networks," *Computer Assisted Methods in Engineering and Science,* vol. 22, no. 2, pp. 141-151, 2017.

[33] S. I. Koval, "Data preparation for neural network data analysis," in *2018 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus)*, 2018: IEEE, pp. 898-901.

[34] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *Journal of machine learning research,* vol. 13, no. 2, 2012.

[35] W. Li, W. W. Ng, T. Wang, M. Pelillo, and S. Kwong, "HELP: An LSTM-based approach to hyperparameter exploration in neural network learning," *Neurocomputing,* vol. 442, pp. 161-172, 2021.

[36] J. Wu, X.-Y. Chen, H. Zhang, L.-D. Xiong, H. Lei, and S.-H. Deng, "Hyperparameter optimization for machine learning models based on Bayesian optimization," *Journal of Electronic Science and Technology,* vol. 17, no. 1, pp. 26-40, 2019.

[37] A. Elzwayie, A. El-Shafie, Z. M. Yaseen, H. A. Afan, and M. F. Allawi, "RBFNN-based model for heavy metal prediction for different climatic and pollution conditions," *Neural Computing and Applications,* vol. 28, no. 8, pp. 1991-2003, 2017.

Smitha Zacaria obtained her Bachelors in Computer Science and Engineering in 2005 from the Cochin University of Science and Technology, India. She received a Master's degree in Computer Science with distinction from the North-West University (NWU) in 2015. Currently, she is studying for her PhD at the NWU's School of Computer Science and Information Systems.

APPENDIX D. CONFIRMATION OF LANGUAGE EDITING

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 (excluding Appendix A):

An alternative trust calculation method using radial basis function neural networks

by

S Zacaria

Dr Isabel J Swart

Date: 6 September 2022

23 Poinsettia Close
Van der Stel Park
Dormehlsdrift
GEORGE
6529
Tel: (044) 873 0111
Cell: 082 718 4210
e-mail: isaswart@telkomsa.net