

Semi-supervised segmentation within a predictive modelling context

**DG Breed
12242950**

Thesis submitted for the degree Philosophiae Doctor in
Risk Analysis at the Potchefstroom Campus of the North-
West University

Promoter: Prof T Verster
Co-promoter: Prof SE Terblanche

May 2017

Preface

This work was completed whilst being employed on a full-time basis in the banking sector in South Africa and a lot of the experience and best practices gained from predictive model development in the banking sector is utilised in the analysis performed in this study. Furthermore, part of the data used in this study was also obtained from the banking sector in South Africa.

Over the course of this study, various parts of this work were presented in a number of formats to scientific audiences.

A paper, summarising parts of Chapter 3 (to define the developed semi-supervised segmentation algorithm) and initial results on the direct marketing data set in Chapter 5, was presented to the Operations Research Society of South Africa (ORSSA) conference, held at the Cradle of Humankind, South Africa in 2012.

In 2013, a full paper was published in the proceedings of the ORSSA conference, held in Stellenbosch, South Africa (Breed et al., 2013). The paper underwent peer-review and a summary of the paper was also presented at the conference. This paper again introduced the definitions and notation from Chapter 3, summarising further results on the direct marketing data set in Chapter 5 and also compared these results to initial results obtained from Wong's semi-supervised segmentation method discussed in Chapter 7.

A paper was presented at the Fifth International Conference of Mathematics in Finance in 2014, held at Skukuza in the Kruger National Park in South Africa. It sets out results obtained from some of the additional, publicly available data sets used in Chapter 5, but also summarised some results from Chapter 4 and initial results on the simulated data in Chapter 6.

Finally, two papers were submitted to scientific journals in 2016, the first of which summarises background from Chapter 3 and results from Chapter 6 and was submitted to the Orion scientific journal (September 2016). The second paper focuses on the results obtained in Chapter 5 and was submitted to the South African Journal of Science (November 2016). Both these journals are blind-reviewed before approval and publication.

“So let your desire be for wisdom: if you have it, there will be a future, and your hope will not be cut off.”

Proverbs 24:14 (BBE)

NORTH WEST UNIVERSITY (NWU)

Abstract

Faculty of Natural Sciences
Centre for Business Mathematics and Informatics

Doctor of Philosophy

Semi-supervised segmentation within a predictive modelling context

by Douw Gerbrand Breed

Industry standards and best practices on robust model development have been refined over many years. Even though many software tools are available to simplify the process today, developing a practically implementable model for long-term use still involves substantial human input. Subsequently, any methodologies that aid in the improvement of model accuracy or increase the efficiency with which models can be developed is welcomed by all involved.

Segmentation of the data that are used for predictive modelling is a well-established practice in the industry. Segmentation of subjects (i.e. observations or customers) is defined in this study as partitioning of the subjects into distinct groups, or subsets, with the aim of developing predictive models on each of the groups separately. The focus of our study will be on broadening the available techniques that can be used for statistical segmentation. Currently two main streams of statistical segmentation exist in the industry, namely unsupervised and supervised segmentation. Both these streams make intuitive sense, depending on the application and the requirements of the models developed, and many examples exist where the use of either technique improved model performance. However, both these streams focus on a single aspect (i.e. either target separation or explanatory variable distribution) and combining both aspects might deliver better results in some instances.

The primary objective of this research is to develop and define a semi-supervised segmentation algorithm as an alternative to the segmentation algorithms currently in use. This algorithm should allow the user, when segmenting for predictive modelling, to not *only* consider the explanatory variables (as is the case with unsupervised techniques such as clustering) or the target variable (as is the case with supervised techniques such as decision trees), but to be able to optimise both simultaneously during the segmentation exercise.

Once we have defined the semi-supervised segmentation algorithm that is based on standard k-means clustering, we comprehensively analyse it by applying it in several different ways. We illustrate visually how the algorithm differs from standard k-means clustering and how it is able to overcome some of the known weaknesses of k-means clustering.

We apply the algorithm to actual data sets from various industries and compare the results to results of other known segmentation algorithms on the same data sets. A number of popular non-linear modelling techniques are also applied to the data sets to compare the accuracy of those techniques to the accuracy obtained with the various segmentation techniques. Simulated data serve to identify a few key data set characteristics that may cause one segmentation technique to outperform another. In addition, we define data set characteristics that suit the semi-supervised segmentation technique best. Finally, we propose two alternative semi-supervised segmentation techniques and measure how these techniques perform on the industry data sets already analysed. We, furthermore, augment a supervised clustering technique found in literature and compare its results to all other results obtained.

Key words: Segmentation; Predictive modelling; Supervised segmentation; Unsupervised segmentation; Semi-supervised segmentation; Clustering; Supervised clustering; Semi-supervised clustering; k-means; Non-linear modelling; Data simulation.

NOORDWES-UNIVERSITEIT (NWU)

Opsomming

Fakulteit Natuurwetenskappe
Sentrum vir Bedryfswiskunde en Informatika

Doktorsgraad in Risiko Analise

Semi-supervised segmentation within a predictive modelling context

deur Douw Gerbrand Breed

Industriestandaarde en praktyke vir die ontwikkeling van robuuste voorspellingsmodelle is oor die betrek van heelwat jare alreeds verfyn. Selfs al bestaan daar reeds 'n groot aantal sagtewarehulpmiddels wat die proses vergemaklik, verg die ontwikkeling van 'n prakties-implementeerbare voorspellingsmodel wat oor die langtermyn gebruik kan word, steeds aansienlike menslike ingryping. As gevolg hiervan, word enige metode wat die akkuraatheid van modelle verbeter of die effektiwiteit waarmee dit ontwikkel word bevorder, deur alle belanghebbendes verwelkom.

Segmentering van die data wat gebruik word vir die ontwikkeling van voorspellingsmodelle, is 'n goedgevestigde praktyk in die industrie. Segmentasie word gedefinieer in hierdie studie as die skeiding van die onderwerpe wat geanaliseer word in afsonderlike deelversamelings, met die doel om aparte modelle op elk van hierdie deelversamelings te ontwikkel. Ons studie fokus daarop om die beskikbare tegnieke wat vir statistiese segmentasie gebruik kan word, te verbreed. Daar bestaan tans twee strome van statistiese segmentasie in die industrie, naamlik gereguleerde en ongereguleerde segmentasie. Beide strome maak intuïtief sin en 'n hele aantal voorbeelde bestaan waar een metode beter presteer as 'n ander. Beide hierdie strome fokus egter op slegs een enkele aspek (dit is, óf doelwitveranderlike skeiding óf die verdeling van die onafhanklike veranderlikes). Indien die fokus gekombineer kan word, mag dit moontlik tot beter resultate in sekere omstandighede lei.

Die primêre doelwit van hierdie navorsing is om 'n semi-gereguleerde segmentasietegniek te ontwikkel en te definieer wat as 'n alternatief tot die segmentasie algoritmes tans in gebruik, kan dien. Die algoritme moet die gebruiker toelaat om nie slegs die doelwitveranderlike of die onafhanklike veranderlikes te oorweeg tydens segmentasie vir voorspellingsmodelle nie, maar hom in staat stel om beide gelyktydig gedurende die segmentasieproses te optimeer.

Nadat ons die semi-gereguleerde segmentasietegniek, gebaseer op standaard k-gemiddeldes-trosvorming, gedefinieer het, analiseer ons dit deeglik deur dit op verskeie maniere aan te wend.

Ons dui visueel aan hoe die algoritme van die standaard k-gemiddeldesalgoritme verskil en hoe dit in staat is om sommige van die welbekende swakhede van die k-gemiddeldesalgoritme te oorkom.

Ons pas die tegniek op werklike datastelle vanuit verskeie industrieë toe en vergelyk die resultate wat verkry word, met resultate verkry deur ander welbekende segmentasietegnieke, op dieselfde datastelle. 'n Paar gewilde, nie-lineêre modelleringstegnieke word op die datastelle toegepas om die akkuraatheid van die tegnieke met die akkuraatheid van die segmentasietegnieke te vergelyk. Ons gebruik verder gesimuleerde data om 'n paar datastelkarakteristieke te identifiseer wat mag veroorsaak dat een segmentasietegniek beter as 'n ander vaar. Benewens identifiseer ons ook die datastelkarakteristieke wat die mees geskik is vir die semi-gereguleerde segmentasietegniek.

Ten slotte stel ons twee alternatiewe semi-gereguleerde segmentasietegnieke voor en vergelyk hoe hulle op die datastelle uit die industrie vaar. Ons pas ook 'n gereguleerde trosvormingstegniek, verkry uit die literatuur, op dieselfde datastelle toe en vergelyk die akkuraatheid met vorige resultate verkry.

Sleutelwoorde: Segmentasie; Voorspellende modellering; Gereguleerde segmentasie; Ongereguleerde segmentasie; Semi-gereguleerde segmentasie; Trosvorming; Gereguleerde trosvorming; Semi-gereguleerde trosvorming; k-gemiddeld; Nie-lineêre modellering; Data simulاسie.

Acknowledgements

First and foremost, praise and thanks goes to my Heavenly Father who blessed me with the abilities to do this work. I would also like to thank everyone who supported me during my studies:

- My family and loved ones who gave support, counsel and motivation throughout the study.
- My promotor, Prof Tanja Verster, who gave me tremendous encouragement, advice and guidance and was tolerant of slow progress when work or personal pressures interfered with this study.
- My assistant-promotor, Prof. Fanie Terblanche whose mathematical prowess was of great help in this thesis.

Contents

1	Introduction and Motivation	1
1.1	Introduction	1
1.2	Objectives	3
1.3	Problem statement	4
1.4	Motivation	5
1.5	Thesis layout	5
2	Literature review and notation	7
2.1	Introduction	7
2.1.1	Research contributions	7
2.2	Predictive modelling	8
2.3	Segmentation within predictive modelling	9
2.4	Predictive modelling techniques	10
2.4.1	Logistic regression	12
2.4.2	Neural networks	13
2.4.3	Support vector machines	14
2.4.4	K-nearest neighbour models	17
2.4.5	Decision trees	17
2.4.6	Random forests	19
2.5	Existing segmentation techniques	20
2.6	Unsupervised techniques	21
2.6.1	K-means	22
2.6.2	Density clustering: k-nearest neighbour	23
2.6.3	Wong's clustering method	25
2.7	Supervised techniques	26
2.7.1	Decision trees	26
2.8	Semi-supervised clustering	27
2.9	Supervised clustering	29
2.10	Comparison between semi-supervised segmentation, supervised and semi-supervised clustering	32
2.11	Measuring target separation	35
2.11.1	Information value	36

2.11.2	Chi-squared	36
2.12	Measuring cluster validity	38
2.13	Conclusion	39
3	Semi-supervised segmentation as applied to k-means (SSSKM)	40
3.1	Introduction	40
3.1.1	Research contributions	41
3.2	The objective function	41
3.3	Design of the algorithm	42
3.4	Inputs into the K-means semi-supervised segmentation algorithm	43
3.5	Repeating the algorithm to measure local optima convergence	45
3.6	Step 1: Variable identification	45
3.7	Step 2: Segment seed initialisation	47
3.8	Step 3: Initial data set and variable preparation	48
3.9	Step 4: Assignment	48
3.9.1	Calculating Euclidian distances between observation and all segment centres	49
3.9.2	Determining whether the distortion function can be calculated	50
3.9.3	Calculate the distortion value given assignment to each segment	50
3.9.4	Transforming the distances and distortion values	50
3.9.5	Evaluating the objective function and assignment	51
3.9.6	Repeating the update step to avoid volatility	52
3.10	Step 5: Assignment-evaluation and update	54
3.11	Step 6: Summarise and log step statistics	55
3.12	Step 7: Randomise data set	55
3.13	Step 8: Evaluate stopping criteria	55
3.14	Step 9: Overfit evaluation and smoothing	57
3.15	Step 10: Final evaluation and result logging	59
3.16	Conclusion	60
4	Impact of SSSKM on traditional k-means segment assignment and segment centroid movement	62
4.1	Introduction	62
4.1.1	Research contributions	63
4.2	Data used	63
4.2.1	Data set 1: Overlapping normal distributions with equal standard deviation and different means	64
4.2.2	Data set 2: Adjoining uniform distributions with varying ranges	65
4.2.3	Data set 3: Two concentric circles	66
4.3	Measuring cluster accuracy	69
4.3.1	Projecting partitions using supervised learning	70
4.3.2	The adjusted Rand index	70
4.4	Results	71
4.4.1	Results data set 1: Overlapping normal distributions with equal standard deviation and different means	72
4.4.2	Results data set 2: Adjoining uniform distributions with varying ranges	76
4.4.3	Results data set 3: Two concentric circles	79

4.5	Conclusion	82
5	Comparison of SSSKM model accuracy to current segmentation techniques and alternative methodologies	83
5.1	Introduction	83
5.1.1	Research contributions	84
5.2	Data used	84
5.2.1	Direct marketing data set	85
5.2.2	Protein tertiary structures data set	86
5.2.3	Credit application data set	87
5.2.4	Wine quality data set	88
5.2.5	Chess (king-rook vs. king) data set	88
5.2.6	Insurance claim prediction data set	89
5.3	Comparing improvement in Gini coefficient due to segmentation	90
5.4	Segmenting using a decision tree (supervised segmentation)	93
5.5	Segmenting using k-means (unsupervised segmentation)	94
5.6	Segmenting using SSSKM (semi-supervised segmentation)	94
5.7	Comparison of segmented model development results	94
5.7.1	Direct marketing data set results	95
5.7.2	Protein tertiary structures data set results	99
5.7.3	Credit application data set results	101
5.7.4	Wine quality data set results	105
5.7.5	Chess (king-rook vs. king) data set results	107
5.7.6	Insurance claim prediction data set results	111
5.8	Summary of segmentation results	112
5.9	Alternative modelling methodologies	115
5.9.1	Direct marketing data set results	118
5.9.2	Protein tertiary structures data set results	119
5.9.3	Credit application data set results	121
5.9.4	Wine quality data set results	123
5.9.5	Chess (king-rook vs. king) data set results	124
5.9.6	Insurance claim prediction data set results	125
5.9.7	Summary of alternative methodology results	127
5.10	Conclusion	128
6	Segmentation performance on simulated data sets	131
6.1	Introduction	131
6.1.1	Research contributions	131
6.2	Base case scenario	132
6.3	Scenario simulation	136
6.3.1	Generate data set characteristic scenarios	139
6.3.2	Calculate expected data set statistics	141
6.3.3	Sample / select desired scenarios	144
6.3.4	Generate simulated data sets	146
6.3.5	Calculate actual data set statistics	152
6.4	Apply segmentation techniques	154
6.5	Results	155

6.5.1	Results on $O = 5$	155
6.5.2	Results on $O = 15$	159
6.5.3	Results on $O = 10$	162
6.5.4	Unsupervised segmentation success	163
6.5.5	Value of w	166
6.6	Summary of results	167
6.7	Conclusion	168
7	Alternative semi-supervised Segmentation Methodologies	170
7.1	Introduction	170
7.1.1	Research contributions	170
7.2	Adding segment size equality as objective to SSSKMIV	171
7.3	SSSKMIV _{SSE} results	174
7.3.1	Direct marketing data set results for SSSKMIV _{SSE}	174
7.3.2	Protein tertiary structures data set results for SSSKMIV _{SSE}	177
7.3.3	Credit application data set results for SSSKMIV _{SSE}	180
7.3.4	Wine quality data set results for SSSKMIV _{SSE}	182
7.3.5	Chess (king-rook vs. king) data set results for SSSKMIV _{SSE}	185
7.3.6	Insurance claim prediction data set results for SSSKMIV _{SSE}	189
7.4	Summary of SSSKMIV _{SSE} results	191
7.5	Semi-supervised segmentation through density clustering	192
7.6	Objective function	193
7.7	Design of the algorithm	194
7.7.1	Step 1: Preliminary segmentation	195
7.7.2	Step 2: Preliminary segment inspection	196
7.7.3	Step 3: Determine preliminary segment adjacency	196
7.7.4	Step 4: Combine segments until K left	198
7.7.5	Step5: Calculate data set statistics	198
7.8	SSSWong results	199
7.8.1	Direct marketing data set results for SSSWong	199
7.8.2	Protein tertiary structures data set results for SSSWong	200
7.8.3	Credit application data set results for SSSWong	203
7.8.4	Wine quality data set results for SSSWong	203
7.8.5	Chess (king-rook vs. king) data set results for SSSWong	205
7.8.6	Insurance claim prediction data set results for SSSWong	206
7.9	Summary of SSSWong results	208
7.10	Augmenting the LK-Means algorithm for semi-supervised segmentation	210
7.11	LK-Means results	211
7.11.1	Direct marketing data set results for LK-Means	211
7.11.2	Protein tertiary structures data set results for LK-Means	213
7.11.3	Credit application data set results for LK-Means	214
7.11.4	Wine quality data set results for LK-Means	216
7.11.5	Chess (king-rook vs. king) data set results for LK-Means	218
7.11.6	Insurance claim prediction data set results for LK-Means	219
7.12	Summary of LK-Means results	221
7.13	Conclusion	222

List of Figures

2.1	Single layer perceptron neural network schematic	13
2.2	Support vectors illustrated when classes are fully separable	15
2.3	Support Vectors illustrated when classes are not separable	16
3.1	Functional flow of the single k-means semi supervised segmentation algo- rithm	44
3.2	Example of a segmented data set	52
3.3	Distribution of distances and IVs before standardisation	53
3.4	Distribution of distances and IVs after standardisation	53
3.5	How the objective function impacts segment assignment	54
3.6	Example of a segmented data set	57
3.7	Example of a segmented data set	59
4.1	First data set shown with no indication of generated segments	65
4.2	First data set shown with generated segments indicated	65
4.3	Second data set shown with no indication of generated segments	67
4.4	Second data set shown with generated segments indicated	67
4.5	Third data set shown with no indication of generated segments	68
4.6	Third data set shown with generated segments indicated	69
4.7	Unsupervised segmentation result for the first data set	74
4.8	semi-supervised segmentation result for the first data set ($w = 0.2$)	74
4.9	Unsupervised centroid movements for the first data set	75
4.10	Semi-supervised segmentation centroid movements for the first data set ($w = 0.2$)	75
4.11	Unsupervised segmentation result for the second data set	77
4.12	Semi-supervised segmentation result for the second data set ($w = 0.9$) . .	78
4.13	Unsupervised centroid movements for the second data set	78
4.14	Semi-supervised segmentation centroid movements for the second data set ($w = 0.9$)	79
4.15	Unsupervised segmentation result for the third data set	80
4.16	Semi-supervised segmentation result for the third data set ($w = 0.6$) . . .	81
4.17	Unsupervised centroid movements for the third data set	81
4.18	Semi-supervised segmentation centroid movements for the third data set ($w = 0.6$)	82
5.1	Illustration of segmentation comparison process	92
5.2	Comparison of segmentation characteristics by supervised weight	97
5.3	Success of target separation compared to Gini improvement by supervised weight	98

5.4	Comparison of best improvements in Gini	99
5.5	Comparison of average improvements in Gini	100
5.6	Comparison of segmentation characteristics by supervised weight	101
5.7	Success of target separation compared to Gini improvement by supervised weight	102
5.8	Comparison of best improvements in Gini	102
5.9	Comparison of average improvements in Gini	103
5.10	Comparison of segmentation characteristics by supervised weight	104
5.11	Success of target separation compared to Gini improvement by supervised weight	104
5.12	Comparison of best improvements in Gini	105
5.13	Comparison of average improvements in Gini	105
5.14	Comparison of segmentation characteristics by supervised weight	106
5.15	Success of target separation compared to Gini improvement by supervised weight	107
5.16	Comparison of best improvements in Gini	108
5.17	Comparison of average improvements in Gini	108
5.18	Comparison of segmentation characteristics by supervised weight	109
5.19	Success of target separation compared to Gini improvement by supervised weight	109
5.20	Comparison of best improvements in Gini	110
5.21	Comparison of average improvements in Gini	110
5.22	Comparison of segmentation characteristics by supervised weight	111
5.23	Success of target separation compared to Gini improvement by supervised weight	112
5.24	Comparison of best improvements in Gini	112
5.25	Comparison of average improvements in Gini	113
6.1	Process flow diagram for simulating data and comparing results	138
6.2	Distribution of IVs of 20000 scenarios	143
6.3	Distribution of CH values of 20000 scenarios, with $O = 1$ and $O = 15$	144
6.4	Variable roles	147
6.5	Cumulative population distribution of six simulated segments	149
6.6	Illustrating how y is assigned	150
6.7	Graphical representation of how the target variable is assigned for Segment 1	151
6.8	Graphical representation of how the target variable is assigned for Segment 6	151
6.9	Graphical presentation of the cumulative distribution of Equation 6.12	152
6.10	Graphical presentation of the cumulative distribution of Equation 6.13	152
6.11	Accuracy of the simulated IVs	153
6.12	Accuracy of the simulated CH values	154
6.13	IVs obtained by segmentation technique and IV group	156
6.14	CH values obtained by segmentation technique and CH group for $O = 5$	156
6.15	Ginis obtained by segmentation technique and IV group for $O = 5$	157
6.16	Gini improvement as a percentage of true improvement obtained by segmentation technique and IV group for $O = 5$	158

6.17	Gini improvement as a percentage of true improvement obtained by segmentation technique and CH group for $O = 5$	158
6.18	Gini improvement as a percentage of true improvement obtained by segmentation technique and both CH and IV group for $O = 5$	159
6.19	IVs obtained by segmentation technique and IV group	160
6.20	CH values obtained by segmentation technique and CH group for $O = 15$	160
6.21	Gini improvement as a percentage of true improvement obtained by segmentation technique and IV group for $O = 15$	161
6.22	Gini improvement as a percentage of true improvement obtained by segmentation technique and CH group for $O = 15$	161
6.23	Gini improvement as a percentage of true improvement obtained by segmentation technique and both CH and IV group for $O = 15$	162
6.24	IVs obtained by segmentation technique and IV group	163
6.25	CH values obtained by segmentation technique and CH group for $O = 10$	163
6.26	Gini improvement as a percentage of true improvement obtained by segmentation technique and IV group for $O = 10$	164
6.27	Gini improvement as a percentage of true improvement obtained by segmentation technique and CH group for $O = 10$	164
6.28	Gini improvement as a percentage of true improvement obtained by segmentation technique and both CH and IV group for $O = 10$	165
6.29	Gini improvement as a percentage of true improvement obtained by segmentation technique and IV group for low IVs	166
6.30	Gini improvement percentage by IV group and value of w	167
7.1	Two segment example of how the segment size equality function maximises when segment sizes are equal	173
7.2	Impact on population percentages by increasing ν	175
7.3	IVs obtained compared to previous results	176
7.4	CH values obtained compared to previous results	176
7.5	Gini improvement over unsegmented Gini on the validation set shown by number of segments	177
7.6	Gini improvement over unsegmented Gini on the validation set split by value of ν	178
7.7	Impact on population percentages by increasing ν	178
7.8	IVs obtained compared to previous results	179
7.9	CH values obtained compared to previous results	179
7.10	Gini improvement over unsegmented Gini on the validation set shown by number of segments	180
7.11	Gini improvement over unsegmented Gini on the validation set split by value of ν	181
7.12	Impact on population percentages by increasing ν	181
7.13	IVs obtained compared to previous results	182
7.14	CH values obtained compared to previous results	182
7.15	Gini improvement over unsegmented Gini on the validation set shown by number of segments	183
7.16	Gini improvement over unsegmented Gini on the validation set split by value of ν	183
7.17	Impact on population percentages by increasing ν	184

7.18	IVs obtained compared to previous results	184
7.19	CH values obtained compared to previous results	185
7.20	Gini improvement over unsegmented Gini on the validation set shown by number of segments	185
7.21	Gini improvement over unsegmented Gini on the validation set split by value of ν	186
7.22	Impact on population percentages by increasing ν	186
7.23	IVs obtained compared to previous results	187
7.24	CH values obtained compared to previous results	187
7.25	Gini improvement over unsegmented Gini on the validation set shown by number of segments	188
7.26	Gini improvement over unsegmented Gini on the validation set split by value of ν	188
7.27	Impact on population percentages by increasing ν	189
7.28	IVs obtained compared to previous results	190
7.29	CH values obtained compared to previous results	190
7.30	Gini improvement over unsegmented Gini on the validation set shown by number of segments	191
7.31	Gini improvement over unsegmented Gini on the validation set split by value of ν	191
7.32	Example for notational purposes: Wong's method	194
7.33	Process flow diagram for SSSWong	195
7.34	Example of identifying segment neighbours	197
7.35	IV and CH values compared to other segmentation techniques	199
7.36	Comparison of IVs obtained versus Gini improvement	200
7.37	Gini improvement by number of segments	201
7.38	IV and CH values compared to other segmentation techniques	201
7.39	Comparison of IVs obtained versus Gini improvement	202
7.40	Gini improvement by number of segments	202
7.41	IV and CH values compared to other segmentation techniques	203
7.42	Comparison of IVs obtained versus Gini improvement	204
7.43	Gini improvement by number of segments	204
7.44	IV and CH values compared to other segmentation techniques	205
7.45	Comparison of IVs obtained versus Gini improvement	205
7.46	Gini improvement by number of segments	206
7.47	IV and CH values compared to other segmentation techniques	206
7.48	Comparison of IVs obtained versus Gini improvement	207
7.49	Gini improvement by number of segments	207
7.50	IV and CH values compared to other segmentation techniques	208
7.51	Comparison of IVs obtained versus Gini improvement	208
7.52	Gini improvement by number of segments	209
7.53	IV and CH values compared to other segmentation techniques	212
7.54	Comparison of IVs obtained versus Gini improvement	212
7.55	Gini improvement by number of segments	213
7.56	IV and CH values compared to other segmentation techniques	214
7.57	Comparison of IVs obtained versus Gini improvement	214
7.58	Gini improvement by number of segments	215

7.59	IV and CH values compared to other segmentation techniques	215
7.60	Comparison of IVs obtained versus Gini improvement	216
7.61	Gini improvement by number of segments	216
7.62	IV and CH values compared to other segmentation techniques	217
7.63	Comparison of IVs obtained versus Gini improvement	217
7.64	Gini improvement by number of segments	218
7.65	IV and CH values compared to other segmentation techniques	219
7.66	Comparison of IVs obtained versus Gini improvement	219
7.67	Gini improvement by number of segments	220
7.68	IV and CH values compared to other segmentation techniques	220
7.69	Comparison of IVs obtained versus Gini improvement	221
7.70	Gini improvement by number of segments	221
8.1	Functional flow of the single K-means semi supervised segmentation algorithm	226
A.1	Enterprise Miner TM - diagram layout	236
A.2	Segment size	237
A.3	CCC graph	237
A.4	Segment characteristic distributions	237
A.5	FICO Model Builder - Segmentation process	238

List of Tables

2.1	Example of the possible combinations (C) for a data set with four observations	22
2.2	Similarities between LK-Means and the proposed K-means semi-supervised segmentation algorithm	31
2.3	Differences between LK-Means and SSSKM	32
2.4	Comparison between semi-supervised clustering, supervised clustering and semi-supervised segmentation	34
3.1	Comparison of percentage of segments initialised with no observations for 100 segmentation iterations	48
3.2	How the objective function impacts segment assignment	54
4.1	Characteristics of the first data set	64
4.2	Characteristics of the second data set	66
4.3	Characteristics of the third data set	68
4.4	Initial values of $\mathbf{X}_1$ and $\mathbf{X}_2$ for each of the segments	72
4.5	Results for the first data set	73
4.6	Segment-specific accuracy results using PPA	73
4.7	Results for the second data set	76
4.8	Segment-specific accuracy results using PPA	77
4.9	Results for the third data set	79
4.10	Segment-specific accuracy results using PPA	80
5.1	Additional derived characteristics for the chess (king-rook vs. king) data set	91
5.2	Summary of results	113
5.3	Comparison of average SSSKM improvements in Gini between data sets	115
5.4	Comparison of performance of alternative modelling methodologies to segmentation with logistic regression for the direct marketing data set	119
5.5	Further detail on segmentation performance compared to alternative techniques for the direct marketing data set	120
5.6	Comparison of performance of alternative modelling methodologies to segmentation with logistic regression for the protein tertiary structure data set	120
5.7	Further detail on segmentation performance compared to alternative techniques for the protein tertiary structure data set	121
5.8	Comparison of performance of alternative modelling methodologies to segmentation with logistic regression for the credit scoring data set	122

5.9	Further detail on segmentation performance compared to alternative techniques for the credit scoring data set	122
5.10	Comparison of performance of alternative modelling methodologies to segmentation with logistic regression for the wine quality data set	123
5.11	Further detail on segmentation performance compared to alternative techniques for the wine quality data set	124
5.12	Comparison of performance of alternative modelling methodologies to segmentation with logistic regression for the chess (king-rook vs. king) data set	125
5.13	Further detail on segmentation performance compared to alternative techniques for the chess (king-rook vs. king) data set	125
5.14	Comparison of performance of alternative modelling methodologies to segmentation with logistic regression for the claim prediction data set	126
5.15	Further detail on segmentation performance compared to alternative techniques for the claim prediction data set	126
5.16	Summary of results of alternative techniques compared to segmentation with logistic regression	128
5.17	Summary of ranking position of alternative methodologies	129
6.1	Average characteristic values of 100 simulated base case data sets	134
6.2	Fitted Ginis compared to true fit	135
6.3	Success of segmentation algorithms	136
6.4	Example of the information generated for a scenario	142
6.5	Example of strata use and average IV and CH values for $O = 3$	146
6.6	Average requested event rate for example data set	149
6.7	Summary of best performing segmentation methodology for all characteristics explored	167
7.1	Performance ranking of all segmentation techniques on the data sets analysed	222
8.1	Summary of results	228
8.2	Summary of results of alternative techniques compared to segmentation with logistic regression	228
8.3	Summary of best performing segmentation methodology for all characteristics explored	229
8.4	Performance of all segmentation techniques on the data sets analysed	231
B.1	Data dictionary of the direct marketing data set	241
B.2	Data dictionary of the direct marketing data set (continued)	242
B.3	Data dictionary of the direct marketing data set (continued)	243
B.4	Data dictionary of the protein structure data set obtained from UCI	243
B.5	Data dictionary of original data set obtained from Kaggle	244
B.6	Data dictionary of the wine quality data set obtained from UCI	245
B.7	Data dictionary of the chess (king-rook vs. king) data set obtained from UCI	245
B.8	Available information for the claim prediction data set	246
B.9	Statistics retained for comparing segmentation results	249

B.10	Comparison of segmentation characteristics by supervised weight for the direct marketing data set	250
B.11	Success of target separation compared to Gini improvement by supervised weight for the direct marketing data set	251
B.12	Comparison of best improvements in Gini by number of segments for the direct marketing data set	251
B.13	Comparison of average improvements in Gini by number of segments for the direct marketing data set	252
B.14	Comparison of segmentation characteristics by supervised weight for the protein tertiary structure data set	252
B.15	Success of target separation compared to Gini improvement by supervised weight for the protein tertiary structure data set	253
B.16	Comparison of best improvements in Gini by number of segments for the protein tertiary structure data set	253
B.17	Comparison of average improvements in Gini by number of segments for the protein tertiary structure data set	254
B.18	Comparison of segmentation characteristics by supervised weight for the credit scoring data set	254
B.19	Success of target separation compared to Gini improvement by supervised weight for the credit scoring data set	255
B.20	Comparison of best improvements in Gini by number of segments for the credit scoring data set	255
B.21	Comparison of average improvements in Gini by number of segments for the credit scoring data set	256
B.22	Comparison of segmentation characteristics by supervised weight for the wine quality data set	256
B.23	Success of target separation compared to Gini improvement by supervised weight for the wine quality data set	257
B.24	Comparison of best improvements in Gini by number of segments for the wine quality data set	257
B.25	Comparison of average improvements in Gini by number of segments for the wine quality data set	258
B.26	Comparison of segmentation characteristics by supervised weight for the chess (king-rook vs. king) data set	258
B.27	Success of target separation compared to Gini improvement by supervised weight for the chess (king-rook vs. king) data set	259
B.28	Comparison of best improvements in Gini by number of segments for the chess (king-rook vs. king) data set	259
B.29	Comparison of average improvements in Gini by number of segments for the chess (king-rook vs. king) data set	260
B.30	Comparison of segmentation characteristics by supervised weight for the claim prediction data set	260
B.31	Success of target separation compared to Gini improvement by supervised weight for the claim prediction data set	261
B.32	Comparison of best improvements in Gini by number of segments for the claim prediction data set	261
B.33	Comparison of average improvements in Gini by number of segments for the claim prediction data set	262

B.34	Property values for the auto neural model	267
B.35	Property values for the support vector machines model	274
B.36	Property values for the memory based reasoning model	275
B.37	Property values for the decision tree model	282
B.38	Property values for the gradient boosting model	286
B.39	Best Ginis obtained for the direct marketing data set	287
B.40	Best Ginis obtained for the protein tertiary structure data set	287
B.41	Best Ginis obtained for the credit scoring data set	288
B.42	Best Ginis obtained for the wine quality data set	288
B.43	Best Ginis obtained for the chess (king-rook vs. king) data set	288
B.44	Best Ginis obtained for the claim prediction data set	289
C.1	IV values obtained by each segmentation method by IV Group compared to actual IV for $O = 5$	290
C.2	CH values obtained by each segmentation method by IV Group compared to actual CH for $O = 5$	290
C.3	Gini values obtained by each of the segmentation methodologies compared with true Gini by IV group for $O = 5$	291
C.4	Gini values obtained by each of the segmentation methodologies compared with true Gini by CH group for $O = 5$	291
C.5	Improvement in Ginis by IV group as a percentage of true Gini improvement for $O = 5$	291
C.6	Improvement in Ginis by CH group as a percentage of true Gini improvement for $O = 5$	291
C.7	Improvement in Ginis by CH and IV group as a percentage of true Gini improvement for the unsupervised example and $O = 5$	292
C.8	IV values obtained by each segmentation method by IV Group compared to actual IV for $O = 15$	292
C.9	CH values obtained by each segmentation method by IV Group compared to actual CH for $O = 15$	292
C.10	Gini values obtained by each of the segmentation methodologies compared with true Gini by IV group for $O = 15$	292
C.11	Gini values obtained by each of the segmentation methodologies compared with true Gini by CH group for $O = 15$	293
C.12	Improvement in Ginis by IV group as a percentage of true Gini improvement for $O = 15$	293
C.13	Improvement in Ginis by CH group as a percentage of true Gini improvement for $O = 15$	293
C.14	Improvement in Ginis by CH and IV group as a percentage of true Gini improvement for the unsupervised example and $O = 15$	293
C.15	IV values obtained by each segmentation method by IV Group compared to actual IV for $O = 10$	294
C.16	CH values obtained by each segmentation method by IV Group compared to actual CH for $O = 10$	294
C.17	Gini values obtained by each of the segmentation methodologies compared with true Gini by IV group for $O = 10$	294
C.18	Gini values obtained by each of the segmentation methodologies compared with true Gini by CH group for $O = 10$	294

C.19 Improvement in Ginis by IV group as a percentage of true Gini improvement for $O = 10$	294
C.20 Improvement in Ginis by CH group as a percentage of true Gini improvement for $O = 10$	295
C.21 Improvement in Ginis by CH and IV group as a percentage of true Gini improvement for the unsupervised example and $O = 10$	295
C.22 IV values obtained by each segmentation method by IV Group compared to actual IV for the unsupervised example and $O = 15$	295
C.23 CH values obtained by each segmentation method by IV Group compared to actual CH for the unsupervised example and $O = 15$	295
C.24 Gini values obtained by each of the segmentation methodologies compared with true Gini by IV group for the unsupervised example and $O = 15$. .	296
C.25 Gini values obtained by each of the segmentation methodologies compared with true Gini by CH group for the unsupervised example and $O = 15$. .	296
C.26 Improvement in Ginis by IV group as a percentage of true Gini improvement for the unsupervised example and $O = 15$	296
C.27 Improvement in Ginis by CH group as a percentage of true Gini improvement for the unsupervised example and $O = 15$	296
C.28 Improvement in Ginis by CH and IV group as a percentage of true Gini improvement for the unsupervised example and $O = 15$	297
D.1 Impact on population percentages by increasing ν for the direct marketing data set	298
D.2 IV values by distortion weight for the direct marketing data set	298
D.3 CH values by distortion weight for the direct marketing data set	299
D.4 Comparison of Gini improvement by number of segments for the direct marketing data set	299
D.5 Comparison of Gini improvement by number of segments and value of ν for the direct marketing data set	299
D.6 Comparison of actual Ginis by number of segments and value of ν for the direct marketing data set	300
D.7 Impact on population percentages by increasing ν for the protein tertiary structure data set	300
D.8 IV values by distortion weight for the protein tertiary structure data set .	300
D.9 CH values by distortion weight for the protein tertiary structure data set	301
D.10 Comparison of Gini improvement by number of segments for the protein tertiary structure data set	301
D.11 Comparison of Gini improvement by number of segments and value of ν for the protein tertiary structure data set	301
D.12 Comparison of actual Ginis by number of segments and value of ν for the protein tertiary structure data set	302
D.13 Impact on population percentages by increasing ν for the credit application data set	302
D.14 IV values by distortion weight for the credit application data set	302
D.15 CH values by distortion weight for the credit application data set	303
D.16 Comparison of Gini improvement by number of segments for the credit application data set	303

D.17 Comparison of Gini improvement by number of segments and value of ν for the credit application data set	303
D.18 Comparison of actual Ginis by number of segments and value of ν for the credit application data set	304
D.19 Impact on population percentages by increasing ν for the wine quality data set	304
D.20 IV values by distortion weight for the wine quality data set	304
D.21 CH values by distortion weight for the wine quality data set	305
D.22 Comparison of Gini improvement by number of segments for the wine quality data set	305
D.23 Comparison of Gini improvement by number of segments and value of ν for the wine quality data set	305
D.24 Comparison of actual Ginis by number of segments and value of ν for the wine quality data set	306
D.25 Impact on population percentages by increasing ν for the chess data set	306
D.26 IV values by distortion weight for the chess data set	306
D.27 CH values by distortion weight for the chess data set	307
D.28 Comparison of Gini improvement by number of segments for the chess data set	307
D.29 Comparison of Gini improvement by number of segments and value of ν for the chess data set	307
D.30 Comparison of actual Ginis by number of segments and value of ν for the chess data set	308
D.31 Impact on population percentages by increasing ν for the claim prediction data set	308
D.32 IV values by distortion weight for the claim prediction data set	308
D.33 CH values by distortion weight for the claim prediction data set	309
D.34 Comparison of Gini improvement by number of segments for the claim prediction data set	309
D.35 Comparison of Gini improvement by number of segments and value of ν for the claim prediction data set	309
D.36 Comparison of actual Ginis by number of segments and value of ν for the claim prediction data set	310
D.37 CH and IV values by distortion weight for SSSWong for the direct marketing data set	311
D.38 Comparison of Gini and IV improvement for SSSWong for the direct marketing data set	311
D.39 Gini improvement by nr segments for SSSWong for the direct marketing data set	312
D.40 Actual Ginis by nr of segments for SSSWong for the direct marketing data set	312
D.41 CH and IV values by distortion weight for SSSWong for the protein tertiary structure data set	313
D.42 Comparison of Gini and IV improvement for SSSWong for the protein tertiary structure data set	313
D.43 Gini improvement by nr segments for SSSWong for the protein tertiary structure data set	314

D.44 Actual Ginis by nr of segments for SSSWong for the protein tertiary structure data set	314
D.45 CH and IV values by distortion weight for SSSWong for the credit application data set	315
D.46 Comparison of Gini and IV improvement for SSSWong for the credit application data set	315
D.47 Gini improvement by nr segments for SSSWong for the credit application data set	316
D.48 Actual Ginis by nr of segments for SSSWong for the credit application data set	316
D.49 CH and IV values by distortion weight for SSSWong for the wine quality data set	317
D.50 Comparison of Gini and IV improvement for SSSWong for the wine quality data set	317
D.51 Gini improvement by nr segments for SSSWong for the wine quality data set	318
D.52 Actual Ginis by nr of segments for SSSWong for the wine quality data set	318
D.53 CH and IV values by distortion weight for SSSWong for the chess data set	319
D.54 Comparison of Gini and IV improvement for SSSWong for the chess data set	319
D.55 Gini improvement by nr segments for SSSWong for the chess data set . .	320
D.56 Actual Ginis by nr of segments for SSSWong for the chess data set . . .	320
D.57 CH and IV values by distortion weight for SSSWong for the claim prediction data set	321
D.58 Comparison of Gini and IV improvement for SSSWong for the claim prediction data set	321
D.59 Gini improvement by nr segments for SSSWong for the claim prediction data set	322
D.60 Actual Ginis by nr of segments for SSSWong for the claim prediction data set	322
D.61 CH and IV values by distortion weight for LKMeans for the direct marketing data set	323
D.62 Comparison of Gini and IV improvement for LKMeans for the direct marketing data set	323
D.63 Gini improvement by nr segments for LKMeans for the direct marketing data set	324
D.64 CH and IV values by distortion weight for LKMeans for the protein tertiary structure data set	324
D.65 Comparison of Gini and IV improvement for LKMeans for the protein tertiary structure data set	325
D.66 Gini improvement by nr segments for LKMeans for the protein tertiary structure data set	325
D.67 CH and IV values by distortion weight for LKMeans for the credit application data set	326
D.68 Comparison of Gini and IV improvement for LKMeans for the credit application data set	326
D.69 Gini improvement by nr segments for LKMeans for the credit application data set	327

D.70 CH and IV values by distortion weight for LKMeans for the wine quality data set	327
D.71 Comparison of Gini and IV improvement for LKMeans for the wine quality data set	328
D.72 Gini improvement by nr segments for LKMeans for the wine quality data set	328
D.73 CH and IV values by distortion weight for LKMeans for the chess data set	329
D.74 Comparison of Gini and IV improvement for LKMeans for the chess data set	329
D.75 Gini improvement by nr segments for LKMeans for the chess data set . .	330
D.76 CH and IV values by distortion weight for LKMeans for the claim prediction data set	330
D.77 Comparison of Gini and IV improvement for LKMeans for the claim prediction data set	331
D.78 Gini improvement by nr segments for LKMeans for the claim prediction data set	331

Abbreviations

ARI	Adjusted rand index
Avg	Abbreviation used to represent "average"
BIC	Bayesian information criterion
CART	Classification and regression trees
CASP	Critical assessment of protein structure prediction centre
CCC	Cubic clustering criterion
CH value	Calinski-Harabasz criterion, also known as the pseudo-f statistic
CHAID	Chi-squared automatic interaction detection
Chi-squared	The χ^2 statistical hypothesis test
DQP	Decomposed quadratic programming
Dtree	Abbreviation used to represent "decision tree"
FICO	Fair Isaacs corporation
FQP	Full dense quadratic programming
Gini	Used to refer to the Gini Coefficient that measures target separation
IV	Information value
KAGGLE	The world's largest community of data scientists who compete with each other to solve complex data science problems.
KNN	K nearest neighbour
KRK	King-rook-king
LDA	Linear discriminant analysis
LSSVM	Least squares support vector machine
MBR	Memory based reasoning
MDL	Minimum description length
NN	Neural network

Nr	Abbreviation used to represent "number", or "number of"
NrSegs	Abbreviation used to represent "number of segments"
PAM	Partitioning around medoids
PPA	Projected partition accuracy
PSPC	Protein structure prediction center
PWC	Price Waterhouse Coopers
RMSD	Root-mean-square deviation
RPA	Recursive partitioning algorithms
SAS® ¹	A software system for data analysis and report writing, formerly known as Statistical Analysis System. SAS is the world's largest privately held software company
SCEC	Supervised clustering using evolutionary computing
Seg	Abbreviation used to represent "segment"
SPAM	Supervised partitioning around medoids
SRIDHCR	Single representative insertion/deletion steepest decent hill climbing with randomized restart
SSS	Semi-supervised segmentation
SSSKM	Semi-supervised segmentation applied to the k-means algorithm
SSSKMCSQ	Semi-supervised segmentation applied to the k-means algorithm using chi-square as distortion measure
SSSKMIV	Semi-supervised segmentation applied to the k-means algorithm using IV as distortion measure
SSSKMIV _{SSE}	Semi-supervised segmentation applied to the k-means algorithm using IV as distortion measure with the addition of segment size equality
SSSWong	Semi-supervised segmentation applied to Wong's density clustering methodology
SVM	Support vector machine
UCI	University of California, Irvine
Wgt	Abbreviation used to represent "weight"

¹SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. "®" indicates USA registration.

*Dedicated to wife, Melissa Breed, who supported and motivated me
from the start to the very end*

Chapter 1

Introduction and Motivation

1.1 Introduction

As technology around the world develops and the volume and complexity of available information escalates, the tools that businesses use to make decisions need to develop. Businesses are required to gather and store more and more data from customers they interact with, actions they take and decisions they reach (SAICA, 2013). In order for many businesses to remain relevant, they need to develop ways to leverage the additional information available to them. Advances in technology in recent years have led to an exceedingly interconnected financial and corporate realm that enables instant communication globally. This makes it possible for smaller companies to transact on the global stage, which increases competition and demands innovative business models (Parr-Rud, 2012).

It is not only the business sector who benefits from or is impacted by these developments. Due to the advances in technology, the consumer of today has access to global products and services. This means that they can increasingly demand products of improved quality, at lower prices that are delivered faster. For companies to be competitive, therefore, they need to become more efficient in how they act and the utilisation of digital information (in an automated way) is the most prominent method by which this can be achieved (McAfee et al., 2012).

Predictive modelling is one of the ways businesses utilise available data to gain insight into their customers and increase their profits. Using predictive models effectively can serve a variety of goals such as (Parr-Rud, 2012):

- Attracting new customers through direct marketing response modelling.

- Reducing risk through risk, loss or claim modelling.
- Retaining profitable customers through attrition modelling.
- Identifying fraud through fraud modelling.
- Improving product classification through automated classification models.

Predictive modelling, as defined in this study, is concerned with predicting future behaviour or trends based on information available at present (Mouhab, 1995). Predictive modelling is not a new concept, but due to the developments in data and technology, its use has increased exponentially in recent times (Taylor, 2014).

Industry standards and best practices on robust model development have been refined over many years. A large range of software tools may well be available to simplify the process today, but developing a practically implementable model for long term use still involves substantial human intervention (Siddiqi, 2006). Put differently: High quality predictive models require scarce and expensive human resources, a great deal of time and significant investment in infrastructure to ensure successful development and implementation. Subsequently, any methodologies that aid in the improvement of model accuracy or increase the efficiency with which models can be developed, is welcomed by all involved (Anderson, 2007, PWC, 2015, Siddiqi, 2006).

Segmentation of the data employed for predictive modelling is a well-established practice in the industry (Anderson, 2007, Siddiqi, 2006, Thomas, 2009). Segmentation of subjects (i.e. observations or customers) is defined in this study as partitioning of the subjects into distinct groups, or subsets, with the aim of developing predictive models on each of the groups separately.

There are a number of reasons to segment a customer (or subject) base before developing predictive models. These are discussed in more detail in Section 2.3, but the ultimate goal of any segmentation is to achieve more accurate, robust and transparent models (Thomas, 2009).

The segmentation can either be experience-based (heuristic), or statistically driven (Siddiqi, 2006). Experience-based segmentation refers to segmentation that is based on human input or experience, whilst statistically driven segmentation uses statistical techniques to derive the most optimal segmentation. Due to its nature, experience-based segmentation is hard to standardise since it is usually very specific to the company / data source (Anderson, 2007), but standard statistical techniques have been derived for statistical segmentation.

Although the two segmentation approaches are derived through very different means, many industry leaders opt to use statistical segmentation techniques to support or inform experts on their choices for the most optimal experience-based segmentation frameworks. An example would be if an expert is undecided between two options of experience-based segmentation and then relies on statistical information about each of these frameworks to make a final decision. In such a situation the final segmentation framework offers the best of both the experience-based and statistical segmentation approaches (Siddiqi, 2006).

The focus of our study is on broadening the techniques available for statistical segmentation. There are at present two main streams of statistical segmentation in the industry, namely unsupervised and supervised segmentation (Friedman et al., 2001, Hand, 1997).

Unsupervised segmentation (Friedman et al., 2001) maximises the dissimilarity of the character distributions of segments, based on a distance function. The technique focuses on the explanatory variables in the models to be developed and does not take the target variable into account. A popular example of unsupervised segmentation is clustering.

Supervised segmentation maximises the target separation or impurity between segments (Hand, 1997). The technique thus focusses on the target variable and not on identifying subjects with similar independent characteristics. Instead, the goal is to maximise target-rate separation between segments. A very popular example of supervised segmentation is the decision tree.

Both these streams make intuitive sense, depending on the application and the requirements of the models developed and there are a great many examples in which the use of either technique improved model performance (Cross, 2008, Fico, 2014). However, both these streams focus on a single aspect (i.e. either target separation or explanatory variable distribution) and combining both aspects could deliver better results in some instances

1.2 Objectives

This thesis has one primary objective, and five secondary objectives.

The primary objective of this research is to develop and define a semi-supervised segmentation algorithm, as an alternative to those segmentation algorithms currently in use. This algorithm should allow the user, when segmenting for predictive modelling, to not *only* consider the explanatory variables (as is the case with unsupervised techniques such as clustering) or the target variable (as is the case with supervised techniques such

as decision trees), but to be able to optimise both during the segmentation exercise. This is done in Chapter 3 of this thesis.

The first secondary objective is to visually illustrate how the new semi-supervised algorithm, through the addition of a "supervised" element to an unsupervised segmentation algorithm, influences and improves its ability to segment. It also shows how it is able to overcome some of the known weaknesses of unsupervised clustering. This is undertaken in Chapter 4.

The next secondary objective of this study is to apply the semi-supervised segmentation algorithm to actual industry data, develop predictive models on each of the segments separately and measure the improvement in model accuracy. These improvements are then compared to the results achieved when popular supervised and unsupervised segmentation techniques are applied to the same data sets for predictive modelling.

A third secondary objective is to compare the model accuracy achieved on industry data, when segmenting for predictive modelling (using unsupervised, semi-supervised as well as supervised techniques), to the accuracy of popular non-linear modelling techniques that may not require segmentation.

Both the second and third secondary objectives are addressed in Chapter 5.

The fourth secondary objective is to make use of simulated model development data to identify the characteristics that may cause one segmentation technique to outperform another, when segmenting for predictive modelling. This is addressed in Chapter 6 for a non-exhaustive selection of characteristics.

The final secondary objective of this study is to explore a number of variations of the newly developed semi-supervised segmentation algorithm that may offer alternative strengths and observe whether these may further enhance the techniques already investigated. A supervised clustering algorithm is also augmented so that it can be used for segmentation, the results of which are compared to all other results obtained. This is carried out in Chapter 7.

1.3 Problem statement

The problem statement of this thesis is to develop, define and comprehensively analyse a semi-supervised segmentation algorithm as an alternative to current segmentation algorithms which not only considers the explanatory variables or the target variable, but optimises both simultaneously during segmentation for predictive modelling.

1.4 Motivation

As discussed in Section 1.1, the amount of data available for predictive modelling is increasing by the day. The ever-growing number of factors that need to be considered makes it increasingly difficult for model developers to pinpoint data vital to their goal and data that are of little value. For this reason, having the right analytical and statistical tools at their disposal becomes all the more essential. Because more and more businesses are relying on advanced statistical data analysis and since predictive model development is no longer a science that is limited to only a few large corporations or disciplines, it is imperative that businesses extract every ounce of value from their data to retain a competitive advantage. Segmenting data sets intelligently for predictive model development is one way in which this can be achieved (Anderson, 2007, Fico, 2014, Siddiqi, 2006, Thomas, 2009, Transunion, 2006).

Even though it is well understood that intelligent segmentation can improve model accuracy significantly, model developers have limited statistical tools available to aid in segmentation and many, therefore, rely on segmentation frameworks that are based on out-of-date methodologies or opinions from stakeholders who do not necessarily understand the insight offered by statistical analysis (Fico, 2014, Transunion, 2006).

In fulfilling the objectives of this study, the aim is to add an additional segmentation methodology (as well as a number of alternative methodologies) to the techniques in practice at the moment, which can aid developers in analysing their data more thoroughly and ultimately lead to the development of more accurate models.

1.5 Thesis layout

Following this chapter, Chapter 2 provides an overview of the statistical methodologies (including the literature study) and the terminology used within this thesis.

Chapter 3 addresses the primary objective of the study and defines the proposed semi-supervised segmentation algorithm that is based on k-means clustering (SSSKM). The algorithm is divided into ten functional steps and each of these steps is discussed in detail. The chapter also introduces the concept of “overfit” of the SSSKM algorithm, which is a recurring theme throughout the thesis.

Chapter 4 explores how the addition of a supervised element to the objective function of the k-means clustering algorithm (the SSSKM algorithm) alters the behaviour of the algorithm. The chapter centres on illustrating results visually and the impact on cluster centroid movement as well as segment assignment may be visualised. This is achieved

by means of simple, two-dimensional data sets, which can easily be represented in two-dimensional graphs. The chapter also explores how the SSSKM algorithm can overcome some of the known weaknesses of standard k-means clustering.

In Chapter 5 the SSSKM algorithm is applied to six data sets from different disciplines in the industry, on which predictive models can be developed. The characteristics of the resulting segments are analysed and the improvement in model accuracy is assessed. These improvements are compared to the improvements (on the same data sets) achieved by models developed on segments informed by other segmentation techniques that are commonly used (i.e. supervised and unsupervised techniques). Finally, the accuracy of the segmented models is compared to the accuracy achieved by a number of popular non-linear modelling techniques on the same data sets and the results are analysed.

Simulation of data with known characteristics is the focus in Chapter 6. The chapter aims to answer some of the questions, raised in Chapter 5, regarding the characteristics that cause one segmentation technique to outperform another depending on the data used. Data sets are generated with known segment assignments, segment and variable characteristics as well as known target-variable dependencies. The impact of these characteristics on the success (or failure) of the various segmentation techniques are subsequently measured and analysed.

Chapter 7 explores two variations of the newly developed semi-supervised segmentation methodologies that are aimed at addressing some of the weaknesses of the SSSKM algorithm observed. It also augments a supervised clustering algorithm developed by another study to function as a segmentation methodology. All three these methodologies are applied to the same data sets featured in Chapter 5 and their performances are compared to the methodologies already assessed.

Chapter 8 concludes the thesis, summarising key findings and proposes possible areas of further research.

Chapter 2

Literature review and notation

2.1 Introduction

Chapter 1 provides an introduction to this thesis and the objectives of the study, along with the layout of the thesis. This chapter relates background on the models and methodologies used throughout the thesis and begins to establish some of the terminology that will be used.

2.1.1 Research contributions

We specifically contribute the following:

- Position the background of predictive modelling and how segmentation within predictive modelling is applied in the industry.
- Discuss and define a number of predictive modelling techniques that is employed in this thesis.
- Distinguish between supervised and unsupervised segmentation techniques.
- Discuss the main segmentation techniques currently in use for predictive modelling.
- Discuss supervised and semi-supervised clustering as well as the similarities and dissimilarities to semi-supervised segmentation.
- Explore previous work performed that has similarities with this study, as well as indicate how it differs from this study.

- Discuss some popular techniques used to measure target separation. These techniques are incorporated throughout this study.
- Discuss a number of popular techniques to measure cluster validity, also used throughout the study.

2.2 Predictive modelling

Predictive modelling is the general concept of building a model that is capable of making predictions. Predictive modelling can be divided further into two sub areas: continuous targets and discrete targets (or labels) (Anderson, 2007, Thomas, 2009). In predictive modelling of discrete targets, the goal is to assign discrete class labels to particular observations as outcomes of a prediction. This is commonly referred to as supervised classification, and in this context predictive modelling is sometimes also referred to as supervised classification.

Predictive modelling first came into use for commercial purposes, in the late 1950s to early 1960s, in the form of credit scoring to assess the credit risk of a potential borrower. However, with huge advances in the computational power available to model developers in the 1980s, the applications for predictive modelling expanded exponentially and has countless areas of application today (Anderson, 2007, McNab and Wynn, 2003, Neter et al., 1996, Thomas, 2009).

The financial and retail industries apply predictive models for numerous reasons, such as application risk scoring (Thomas, 2009), solicitation response modelling (Ghose Dasgupta et al., 1994), behavioural risk management (Anderson, 2007), retention management (Anderson, 2007), revenue ranking (Anderson, 2007), collection and recovery management, fraud prevention (Siddiqi, 2006), price elasticity (sensitivity)(Allenby and Rossi, 1998, Thomas, 2009) and many others. Industry standards and best practices on robust model development have been refined over many years and though many software tools may be available to simplify the process today, developing a practically implementable model for long-term use still involves substantial human intervention. Siddiqi (2006), elaborates on this in a structured manner with special focus on credit risk scorecards. In summary, high-quality predictive models require a great deal of time and effort from experts (from various different disciplines) to ensure successful development and implementation. Subsequently, most developers appreciate any assistance available to aid in this process.

2.3 Segmentation within predictive modelling

Segmentation of predictive models is a well-established practice in the industry today (Anderson, 2007, Siddiqi, 2006, Thomas, 2009). Segmentation is conducted for a range of reasons and in many ways. This section details some of the *reasons* for segmentation, whilst *methods* used for segmentation are explored in Sections 2.6 and 2.7. Segmentation is defined in this study as the practice of classifying (or partitioning) subjects (i.e. observations or customers) into distinct groups or subsets, with the aim of developing predictive models on each of the groups separately.

Siddiqi (2006) divides the methods used for segmentation into two broad categories, namely experience-based (heuristic) and statistically based. Anderson (2007) builds on this and defines the factors driving the need for predictive model segmentation in credit scoring as follows:

1. Market factors (strategic).
 - Where the party using the model seeks to strategically focus on a certain sector or segment and, therefore, requires accurate decision-making abilities in that sector.
2. Subject (customer) based.
 - Where the characteristic profile of a subject results in some information being unavailable or representing (explaining) something inherently different. For instance, young customers may have less credit history available to use in predictive modelling, which is not unexpected and they may, therefore, need to be assessed in a different manner.
3. Operational data factors.
 - Due to operational differences, certain subjects may have less or different data available for modelling. This is different from the previous point in that these differences are not caused by the subject's profile, but by operational divergence.
4. Process factors.
 - This deals with situations where the circumstances of scored subjects (customers) are expected to vastly differ, depending on decisions made after the modelling outcome. An example is where customers may be offered products by means of varying collections strategies or affordability constraints.

Similar motives for segmentation is provided by Thomas (2009) and other industry leaders (Fico, 2014, Transunion, 2006). Thomas concludes that whatever the driving factor, the ultimate goal of segmentation will always be to improve the way in which the outcome of a target variable is predicted. If a clear improvement cannot be observed, the added complexity of segmentation cannot be justified.

In terms of segmentation, this study focuses on factors that can be measured statistically and no experience-based methods are included. Generally experts in the business or industry find it easier to identify segments relating to points one, three and four mentioned above through expert knowledge or experience, but find the optimal segmentation, based solely on customer characteristics (point two), as far less straightforward and that statistical tools will assist to a greater degree. It can further be argued that a relationship exists between heuristic segmentation and statistical results and that the tools explored in this thesis could provide helpful insight to subject matter experts - thereby assisting to combine both statistical results and expert knowledge. Some may argue that although segmentation may be useful for certain modelling techniques, other techniques have been developed to cater for this sufficiently by analysing and incorporating variable interaction and heteroscedasticity (Francis, 2003). In this regard, one needs to consider the practical shortcomings of more sophisticated techniques along with their improved accuracy (if improvement in accuracy is, indeed, achieved). We compare the results of segmentation for predictive modelling to the accuracy obtained by some popular non-linear predictive modelling techniques in the industry in Chapter 5. The results can certainly not be conclusive on all modelling applications and it is not the focus of this study to explore every modelling technique developed, but it would be prudent to explore a few of the more common techniques on the data sets featured in this study.

2.4 Predictive modelling techniques

As discussed in Section 2.2, predictive modelling has received substantial attention in the last few decades and techniques have been tailored to many specific requirements. Some of the most popular predictive modelling techniques used in the industry include logistic regression, linear regression, neural networks (Anderson, 2007, Thomas, 2009) and decision trees. Other techniques have been developed such as support vector machines (Cristianini and Shawe-Taylor, 2000), random forests (Breiman et al., 1984), nearest neighbour models (Thomas, 2009), robust regression (Andersen, 2008), semi-parametric regression (Ruppert et al., 2003) and generalized linear models (McCullagh et al., 1989).

Every one of these techniques has applications for which they are superior to other techniques, but numerous sources agree that logistic regression is one of the most widely used

in the industry and certainly the financial industry at present (Anderson, 2007, Siddiqi, 2006, Thomas, 2009, Tu, 1996). Many non-linear techniques¹ have been widely tested on industry applications, but they have not been very popular in practical applications. Thomas (2009) avers that this can broadly be ascribed to a lack of robustness and/or transparency. Although it may lead to better predictions, the complexity of the way the predictions are derived make them difficult to understand and interpret and could cause instability in changing environments. As an example, even a simple neural network with only three hidden layers may consist of a multitude of modelled relationships between the target variable and the explanatory variable and these relationships may not be intuitive to explain to stakeholders not familiar with statistical modelling techniques.

The specifics of these and other modelling techniques are detailed in a great number of statistical texts (Anderson, 2007, Caruana and Niculescu-Mizil, 2006, Hand, 1997, Thomas, 2009, Tu, 1996). Since we compare the accuracy of a selection of the more popular non-linear modelling techniques to segmented logistic regression in Chapter 5, we discuss the selected methods in a bit more detail below.

In order to define the functions used by the predictive modelling techniques described below, we will now establish some notation, which will be used (and further enhanced) throughout the thesis (see for instance Section 2.6.1):

- Denote i as the index of observations in the data set under consideration with n observations. i will therefore consist of values 1 through to n .
- Denote p as the index of the m characteristics that are considered in the analysis. This will be the characteristics that are considered both for cluster and regression analysis. p will therefore consist of values 1 through to m .
- Denote $\mathbf{x}_i$ as a single observation in the data set where $\mathbf{x}_i = \{x_{i1}, x_{i2}, \dots, x_{im}\}$.
- Denote $\mathbf{X}$ as the $n \times m$ matrix comprised of all characteristics for all observations considered in the analysis. Further to this, $\mathbf{X}_p$ is denoted as a vector of all observations for characteristic p considered in the analysis where $\mathbf{X}_p = \{X_{1p}, X_{2p}, \dots, X_{np}\}$.
- $\mathbf{y}$ is denoted as the vector of known target values y_i , with $i = \{1, \dots, n\}$.
- $\hat{\mathbf{y}}$ will be denoted as the vector of predicted values (as estimated by the predictive modelling technique used) based on the input matrix $\mathbf{X}$, with $\hat{\mathbf{y}}$ therefore linked to the target vector $\mathbf{y}$

¹Note that the term "non-linear techniques" are used here as a collective for techniques that do not require the assumption of a linear relationship between the target variable and the explanatory variables.

- $a\beta_0$ is a numerical value and is defined as the intercept, or bias of the model. In cases where more than one model is defined for additional classes (say, H classes), β_{h0} is a vector of numerical parameters with $h = \{1, \dots, H\}$. When the values of β_0 are estimated by a model, it is denoted as $\hat{\beta}_0$.
- β is vector of numerical parameters that minimises the objective function for a vector of input variables. If those input variables are the inputs $\mathbf{x}_i = \{x_{i1}, x_{i2}, \dots, x_{im}\}$, then β has m values. Similar to β_0 , β_h can be a $m \times H$ dimensional matrix for the case where more than one class is modelled and the parameter estimates of β is denoted as $\hat{\beta}$.
- α is also used here with similar notation to β for models requiring more than one set of parameters per input combination.

2.4.1 Logistic regression

Although the logistic function was already developed during the early nineteenth century (Cramer, 2002, Verhulst, 1838), its use in multivariate predictive modelling was only proposed more than a hundred years later (Cox, 1958). Logistic regression can model posterior probabilities of any H classes (with $H > 1$), and produces probability values between 0 and 1. The general form of the model can be described as (Friedman et al., 2001)

$$\begin{aligned}
 \log \frac{\Pr(y_i = 1 \mid \mathbf{X} = \mathbf{x}_i)}{\Pr(y_i = H \mid \mathbf{X} = \mathbf{x}_i)} &= \beta_{10} + \beta_1^T \mathbf{x}_i \\
 \log \frac{\Pr(y_i = 2 \mid \mathbf{X} = \mathbf{x}_i)}{\Pr(y_i = H \mid \mathbf{X} = \mathbf{x}_i)} &= \beta_{20} + \beta_2^T \mathbf{x}_i \\
 &\vdots \\
 \log \frac{\Pr(y_i = H - 1 \mid \mathbf{X} = \mathbf{x}_i)}{\Pr(y_i = H \mid \mathbf{X} = \mathbf{x}_i)} &= \beta_{(H-1)0} + \beta_{H-1}^T \mathbf{x}_i,
 \end{aligned} \tag{2.1}$$

with $\mathbf{y}$ in this case a vector of target classes $\{1, \dots, H\}$.

The special case where $\mathbf{y}$ is binary (i.e. only has values 1 or 0) therefore simplifies to

$$\log \left(\frac{\Pr(y_i = 1 \mid \mathbf{X} = \mathbf{x}_i)}{1 - \Pr(y_i = 1 \mid \mathbf{X} = \mathbf{x}_i)} \right) = \beta_0 + \beta^T \mathbf{x}_i. \tag{2.2}$$

In this case only one vector for parameters β needs to be found.

We will use logistic regression in this thesis in combination with a number of segmentation techniques to compare the impact of segmenting data sets for improved predictive model accuracy.

2.4.2 Neural networks

Neural networks have been a key interest within predictive modelling for many years and is based on biological neural networks, or central nervous systems found in human and animal brains. Essentially, as illustrated in Figure 2.1, a neural network consists of a network of neurons that exchange information. McCulloch and Pitts (1943) were the first to publish a computational simulation of neural networks, which paved the way to further developments in the area. Some of the most meaningful contributions to the neural networks used today were made by Frank Rosenblatt (Rosenblatt, 1958), who created the concept of a perceptron, which produces probabilistic outputs through addition and multiplication of various inputs and parameters. Another meaningful contribution was by Werbos (1974), who first proposed the concept of error back-propagation, which is one of the most common techniques used to train neural networks today.

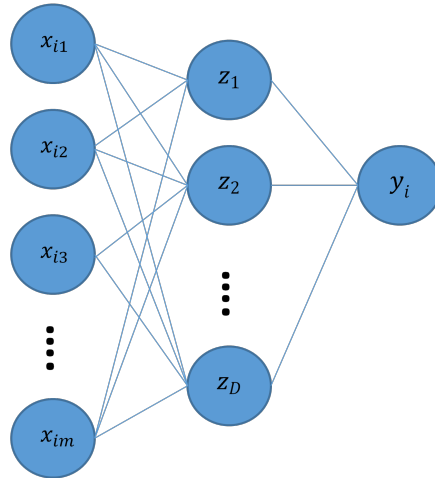


FIGURE 2.1: Single layer perceptron neural network schematic

The neurons illustrated in Figure 2.1 as z_1 to z_D is known as the hidden layer. The neural network illustrated in the figure has only a single hidden layer, but some applications of neural networks can have more. A single output is illustrated in Figure 2.1, which is usually the case when target variables are continuous or binary. However, more outputs are possible if $\mathbf{y}$ is categorical (Friedman et al., 2001).

When only one output is defined, with a single hidden layer (which will be the case when neural networks are used in this study), the output can be described as (Friedman et al., 2001)

$$\begin{aligned} z_d &= \sigma(\alpha_{0d} + \boldsymbol{\alpha}_d^T \mathbf{x}_i), \quad d = 1, \dots, D, \\ \hat{y}_i &= \beta_0 + \boldsymbol{\beta}_d^T \mathbf{z}. \end{aligned} \quad (2.3)$$

In Equation 2.3, $\mathbf{z} = (z_1, z_2, \dots, z_D)$ and $\sigma(v)$ is known as the activation function, which defines the output value and value range, given the input values $\mathbf{x}_i$.

As mentioned, neural networks are often trained through error back-propagation, which aims to minimise some error, usually defined through the sum of square error (Friedman et al., 2001).

2.4.3 Support vector machines

Vladimir Vapnik is generally credited for being the person most instrumental in the development and enhancement of the theory behind modern support vector machines (SVMs) between the early 1960's (Vapnik and Lerner, 1963) end 1990's (Boser et al., 1992, Vapnik, 1996). Since then, applications of support vector machines have grown and this non-linear regression and classification technique is being applied to an increasing number of fields (Beard et al., 2015, Fasshauer et al., 2015, Li, 2015, Sands et al., 2015, Wang and Pardalos, 2015).

Figure 2.2 illustrates the concept of SVMs for the case where two classes are completely separable. The decision boundary is the solid black line and represents a hyperplane defined as (Friedman et al., 2001)

$$\{\mathbf{X}^T \boldsymbol{\beta} + \beta_0 = 0\}. \quad (2.4)$$

Single SVMs can only support two classes for the target variable $\mathbf{y}$ (Vapnik, 1996), although methodologies have been developed to support multi-class targets (Friedman, 1996, Kreßel, 1999). In this study we will only focus on the case where $\mathbf{y}$ is binary. For the purpose of SVMs, $\mathbf{y}$ is defined as $y_i \in \{-1, 1\}$ and $\mathbf{x}_i \in \mathbb{R}^p$. The *margin* shown in Figure 2.2 is of length $2W = \frac{2}{\|\boldsymbol{\beta}\|}$ (with $\boldsymbol{\beta}$ a unit vector measured in Euclidean distance).

If the two classes are separable (as is the case in Figure 2.2), we can find a function $\mathbf{X}^T \boldsymbol{\beta} + \beta_0 = 0$ where $y_i (\mathbf{x}_i^T \boldsymbol{\beta} + \beta_0) > 0 \forall i$. The objective function can be written as (Friedman et al., 2001)

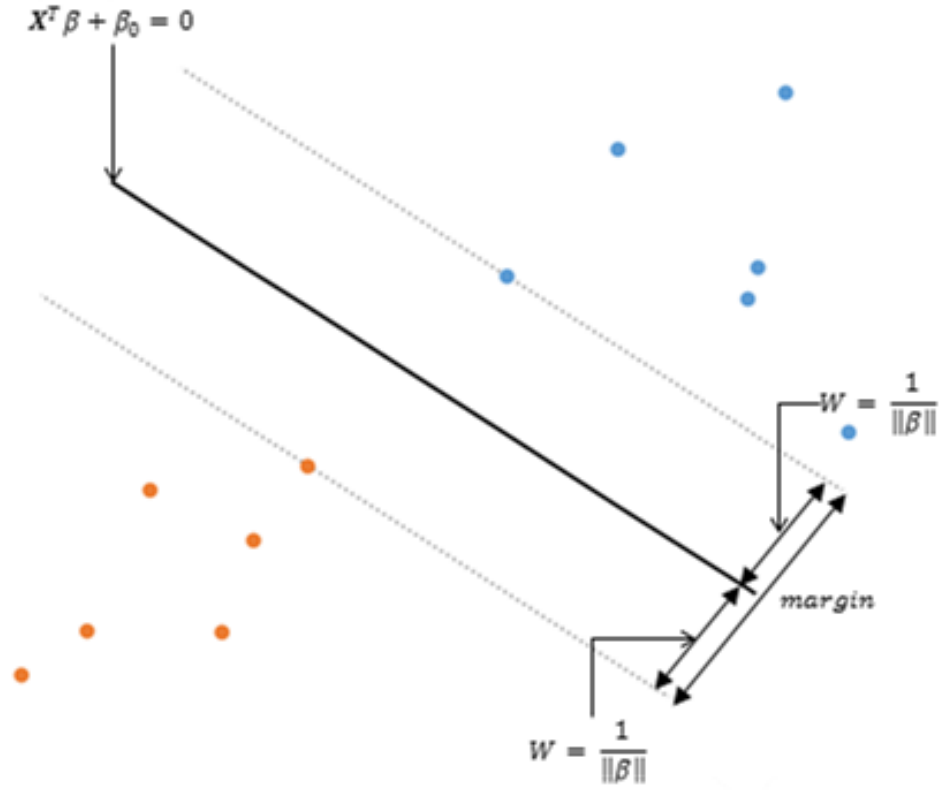


FIGURE 2.2: Support vectors illustrated when classes are fully separable

$$\begin{aligned} & \max_{\beta, \beta_0} \{W\} \\ & \text{subject to: } \|\beta\| = 1, y_i (\mathbf{x}_i^T \beta + \beta_0) \geq W, i = 1, \dots, n. \end{aligned} \quad (2.5)$$

Friedman et al. (2001) shows that Equation 2.5 can be rewritten as:

$$\begin{aligned} & \min_{\beta, \beta_0} \{\|\beta\|\} \\ & \text{subject to: } y_i (\mathbf{x}_i^T \beta + \beta_0) \geq 1, i = 1, \dots, n. \end{aligned} \quad (2.6)$$

Equation 2.6 is a more widely used mathematical format for representing the objective function.

If classes were always completely separable in practice, the study of predictive modelling would certainly not have been as challenging and popular as it is today. Since in practice we deal mostly with cases of inseparable classes, the SVM needs to be expanded to incorporate such instances. This is illustrated in Figure 2.3 (Friedman et al., 2001).

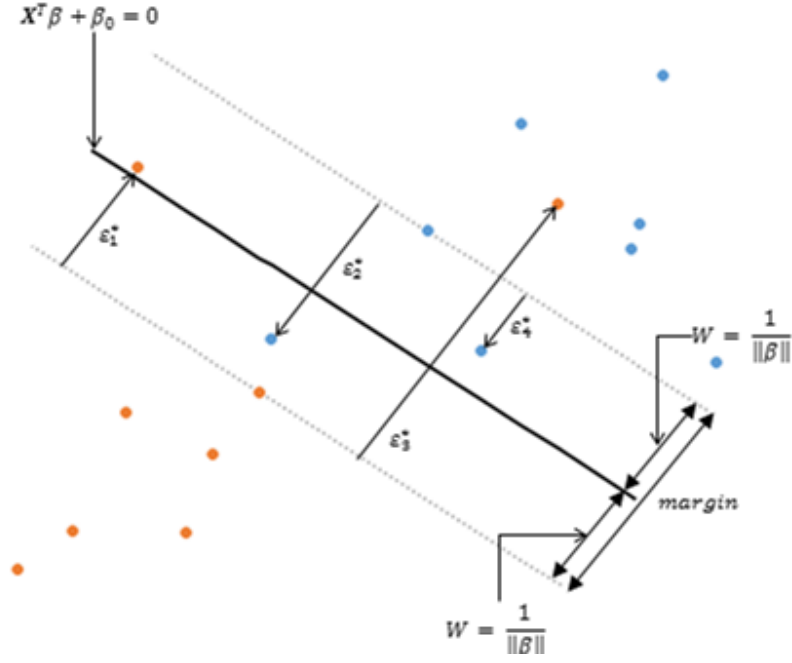


FIGURE 2.3: Support Vectors illustrated when classes are not separable

In Figure 2.3, the error terms ε_i^* is expressed here in combination with W , such that $\varepsilon_i^* = W\varepsilon_i$. The only reason why the error term, ε_i^* , is defined in such a way, is to facilitate a simpler objective final function (see Equation 2.8). We can define error terms for each observation $\mathbf{x}_i$, i.e. $\boldsymbol{\varepsilon} = (\varepsilon_1, \varepsilon_2, \dots, \varepsilon_n)$ (where $\varepsilon_i = \frac{\varepsilon_i^*}{W}$). Where $\mathbf{x}_i$ is situated on the correct side of the hyperplane, $\varepsilon_i = 0$. The constraint in Equation 2.5 can now be altered to incorporate the error functions (Friedman et al., 2001),

$$\begin{aligned} y_i (\mathbf{x}_i^T \boldsymbol{\beta} + \beta_0) &\geq W(1 - \varepsilon_i) \quad \forall i, \\ \varepsilon_i &\geq 0, \sum_{i=1}^n \varepsilon_i \leq L. \end{aligned} \quad (2.7)$$

In Equation 2.7, L represents some constant value.

Through similar assumptions that allows Equation 2.5 to be re-expressed as Equation 2.6, we can re-express Equation 2.7 to be in the format generally used as the objective function for the SVM with non-separable classes (Cristianini and Shawe-Taylor, 2000, Friedman et al., 2001, Wang, 2005),

$$\begin{aligned} \min_{\boldsymbol{\beta}, \beta_0} \{ &\frac{1}{2} \|\boldsymbol{\beta}\|^2 + C \sum_{i=1}^n \varepsilon_i \} \\ \text{subject to: } &\varepsilon_i \geq 0, y_i (\mathbf{x}_i^T \boldsymbol{\beta} + \beta_0) \geq 1 - \varepsilon_i, \forall i. \end{aligned} \quad (2.8)$$

In this case, C is a cost parameter that is a representation of the inverse of both the number of observations that are misclassified, as well as the distance of those observations to the decision boundary. The separable case in Equation 2.6 is a special case where $C = \infty$.

2.4.4 K-nearest neighbour models

We describe k-nearest neighbour (KNN) clustering in Section 2.6.2, and therefore align to the notation that will be used there. Note that Section 2.6.2 deals with KNN being used for unsupervised clustering as opposed to KNN being used for supervised classification or predictive modelling, as discussed here. Assume $d(\mathbf{x}_i, \mathbf{x}_{K(i)})$ is the Euclidean distance from point $\mathbf{x}_i$ to the K th-nearest observation, where K is a user-specified value indicating the number of observations to be used for classification. In the training set, each $\mathbf{x}_i$ has a known class, y_i . KNN classification assigns the estimated class $\hat{y}_i$ to $\mathbf{x}_i$ based on a vote from each of the observations within the N dimensional sphere with $\mathbf{x}_i$ at its centre and radius $d(\mathbf{x}_i, \mathbf{x}_{K(i)})$. It has been shown in studies that results for this algorithm can be improved by weighing votes according to some measure of relevance (e.g. distance to $\mathbf{x}_i$), rather than giving each of the K observations an equal vote. (Dudani, 1976, Gou et al., 2011, MacLeod et al., 1987, Zavrel, 1997).

Duda et al. (2000) show that, in instances where many observations are available, the error rate of a simplistic 1NN (i.e. a KNN classification with $K = 1$ - therefore only the single closest neighbour is used) classifier is less than twice that of the Bayes error rate. The Bayes error rate is the lowest possible error rate for any classifier of a random outcome (see also Friedman et al. (2001)).

As will be seen in Section 3.14 and Section 5.9, we use k-nearest neighbour classifiers in the study both for smoothing (to avoid overfit) as well as for classification.

2.4.5 Decision trees

Decision trees are most often used in this thesis for supervised segmentation, and are therefore discussed in more detail in Section 2.7.1. We give a short description of how the algorithm works here as a predictive modelling technique, and describe its applications further in Section 2.7.1.

Decision trees use recursive partitioning algorithms (RPA) to classify observations into homogenous groups, the groups are formed through repeated attempts at finding the best possible split on the previous branch. They have been developed, applied and

thoroughly researched in various fields, like statistics (Breiman et al., 1984), artificial intelligence (Safavian and Landgrebe, 1991) as well as machine learning (Quinlan, 1993).

Algorithm 2.1 Decision tree algorithm

Input: Training data set

Input: Stopping criteria

For each terminal node of the tree:

while Stopping criteria not met **do**

 calculate which variable will produce the best binary split according to splitting criteria

 Split node and evaluate stopping criteria for new terminal nodes

end while

For each end node E : $Pr(\mathbf{y} = 1) = \frac{\sum_{e \in E} y_e}{Nodesize_E}$

where $Nodesize_E$ is the number of observations in end node E

Algorithm 2.1 provides a simple example of a binary tree (i.e. each terminal node is only split into two nodes) with $\mathbf{y}$ also binary. Note that $Pr(\mathbf{y} = 1)$ refers to the proportion of the actual target variable, y , that has a value of 1 in the end node. The stopping criteria referred to in the algorithm can be a number of rules. Common stopping criteria are (Maimon, 2007):

- Complete purity - i.e. all instances in the terminal node belong to a single value of $\mathbf{y}$.
- Maximum tree depth - i.e. the number of levels formed from the origination node has reached a certain value.
- Minimum node size - i.e. the number of observations in the terminal node is less than some minimum prescribed number (n_{min}).
- Splitting criteria below minimum - i.e. the criteria used to determine best possible splits is below minimum for the particular split.

Various splitting criteria for decision trees have been proposed, with the most common being (Breiman, 1996b):

- Gini - measuring the success of the split by measuring how classes are separated through the Gini coefficient. If $Pr(h)$ is the proportion of $\mathbf{y}$ belonging to class h in the end node, then the Gini for the split is calculated as $1 - \sum_{h=1}^H Pr(h)^2$.
- Entropy - minimising information loss: $-\sum_{h=1}^H Pr(h) \log(Pr(h))$.

- Twoing - used only for binary decision trees and focused on finding two "super-classes" making up more than half of a node, and optimizing the decrease in impurity. Let F_1 be the proportion of the parent node observations that is split into the first end node, and F_2 the proportion split into the second. Similarly $Pr_1(h)$ is the proportion of h in the first node and $Pr_2(h)$ the proportion in the second node. The twoing criteria optimises the function $F_1 F_2 \left(\sum_{h=1}^H |Pr_1(h) - Pr_2(h)| \right)^2$.

2.4.6 Random forests

The concept of random forests as well as the key to their success is closely linked to (and can be seen as having been built upon) the concepts motivating the development of bagging and boosting (Breiman, 1996a, 2001, Freund and Schapire, 1997, Friedman et al., 2001).

Bagging, which is a shortened term for **bootstrap aggregating**, was first proposed by Breiman (1996a). Bagging randomly samples (with replacement) a number of training data sets from the original data set and develop models on each of these sets. The final value assigned to observation $\mathbf{x}_i$ (call it $f(\mathbf{x}_i)$) is the average of the output of each of the models. The reason for using bagging (and for that matter, boosting) over decision trees is to reduce the variance of the error that is usually observed with decision trees.

Boosting, on the other hand, was developed to address a question posed by Kearns and Valiant (1994) on whether weak variables / predictors can form a strong predictor / model when used in conjunction. The standard boosting process (as introduced by Freund and Schapire (1997)) is iterative, and starts by developing a single predictive model. Each observation $\mathbf{x}_i$ in the data set of the previous model is then iteratively reweighed proportionate to the error made by the previous model and a new model is fitted on the reweighed data set until all iterations have been completed (the number of iterations can be a user input). The final value assigned to $\mathbf{x}_i$ (i.e. $f(\mathbf{x}_i)$) is a weighted combination of each of the iterative models fitted.

One common objective (apart from predictive accuracy) of both bagging and boosting, is to reduce the variance of the final model (Maclin and Opitz, 1997, Quinlan, 1996). In a similar sense, traditional random forests can be seen as an altered form of bagging (Breiman, 2001), and provides a simple algorithmic and relatively transparent methodology to achieve reduced variance by averaging unpruned (and therefore unbiased, but noisy) decision trees (Friedman et al., 2001).

Friedman et al. (2001) summarises the random forest methodology as shown in Algorithm 2.2. As can be seen, the algorithm creates a number of ensemble trees (B) and

randomly selects variables that will be considered for splitting each end node. The best split is selected at each iteration, but the fact that the variables that are considered are first chosen at random, ensures that the ensemble trees will likely be very different. The nodes are split until the next split will result in less than a specified minimum number of observations remaining in an end node.

The probability of an observation belonging to a specific label is then calculated as the average of the proportion of that label in each of the end nodes of each ensemble tree to which the observation belongs. For binary target variables, this can be seen as the probability weighted outcome of all the ensemble trees.

Algorithm 2.2 Random forest algorithm

Input: Training Data set

Input: n_{min} {as minimum node size}

Input: B {as number of ensemble trees required}

Input: N {as size of random sample (bootstrap) drawn for each tree}

Input: m {as number of variables to consider for a single split}

for $b = 1$ **to** B **do**

 Draw a random sample $\mathbf{Z}^*$ of size N from the training data set

 For each terminal node of the tree:

while $Nodesize > n_{min}$ **do**

 select m variables at random from the p available variables

 calculate which variable will produce the best binary split

 split node and calculate $Nodesize$ for new terminal nodes

end while

end for

 Output tree ensemble T_b

 for each new point $\mathbf{x}_i$:

$$f(\mathbf{x}_i) = \frac{1}{B} \sum_{b=1}^B T_b(\mathbf{x}_i)$$

2.5 Existing segmentation techniques

Currently two main streams of statistical segmentation exist in the industry, unsupervised and supervised segmentation (Friedman et al., 2001, Hand, 1997).

Unsupervised segmentation (Friedman et al., 2001) maximises the dissimilarity of the character distributions of segments based on a distance function. The technique focuses on the characteristics which will form the explanatory variables in the models to be developed and does not take the target variable into account. A popular example of unsupervised segmentation is clustering.

Supervised segmentation maximises the target separation or impurity between segments (Hand, 1997). The technique therefore focusses on the target variable and not on identifying subjects with similar independent characteristics. Instead, the goal is to maximise target rate separation between segments.

Both these streams make intuitive sense depending on the application and the requirements of the models developed and many examples exist where the use of either technique improved model performance (Cross, 2008, Fico, 2014). However, both these streams focus on a single aspect (i.e. either target separation or explanatory variable distribution) and combining both aspects might deliver better results in some instances. In order to proceed with details on the proposed semi-supervised approach, an overview of supervised and unsupervised segmentation as used in the industry is provided next.

2.6 Unsupervised techniques

Unsupervised learning techniques have been around for many years and essentially focus on finding similarities between observations in data sets where no target variable is available. Many techniques exist, but most do this by combining observations within data sets in a way that ensure that the combined observations are as similar to one another as possible, whilst being as dissimilar to the observations they have not been combined with as can be achieved.

Popular unsupervised techniques include:

- Clustering (e.g., k-means, density based or hierarchical clustering).
- Hidden markov models (Baum and Petrie, 1966).
- Feature extraction techniques such as principal component analysis.

Of these techniques, the one most commonly used for segmentation is clustering (Friedman et al., 2001). A good example of where this is used in mainstream software tools is the SAS[®] Enterprise Miner[™] software's data exploration toolkit ².

Cluster analysis traces its roots back to the early 1960's (Sokal and Sneath, 1963), and was the subject of many studies from the early 1970's (Anderberg, 1973, Bijnen et al., 1973, Jardine and Sibson, 1971). As with all popular techniques, the applications of cluster analysis grew significantly from its origin, and today it has an unquantifiable number of applications.

²SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

The next few paragraphs provide some background on popular clustering techniques currently used for predictive model segmentation.

2.6.1 K-means

K-means clustering is one of the simplest and most common clustering techniques used in data analysis today (Friedman et al., 2001). It follows a very simple iterative process that continuously cycles through the entire data set until convergence is achieved. We build on the notation defined in Section 2.4 as we describe the functional details of the k-means clustering algorithm:

On completion of the clustering algorithm each observation $\mathbf{x}_i$, with $i = 1, 2, \dots, n$, will be assigned to one of the segments $S_1, S_2, \dots, S_K$ where each S_j denotes an index set containing the observation indices of all the variables assigned to it for $S_j = 1, \dots, K$. That is, if observation $\mathbf{x}_i$ is assigned to segment S_j , then $i \in S_j$. Further denote $\mathbf{u}_j = \{u_{j1}, u_{j2}, \dots, u_{jm}\}$ as the mean (centroid) of segment S_j , for example $\mathbf{u}_{j1}$ will be the mean of characteristic $\mathbf{X}_1$. The distance from each observation $\mathbf{x}_i$ to the segment mean $\mathbf{u}_j$ is given by a distance function $d(\mathbf{x}_i, \mathbf{u}_j)$. If an Euclidian distance measure is used, then

$$d(\mathbf{x}_i, \mathbf{u}_j) = \|\mathbf{x}_i - \mathbf{u}_j\|^2 \quad (2.9)$$

where the terminology $\|\mathbf{g}\|^2$ is used to represent the length of vector $\mathbf{g}$ in Euclidean distance.

The objective of the ordinary k-means clustering algorithm is to make segment assignments in order to minimise the inter-segment distances. For notational purposes we introduce $c \in \mathbb{C}$ as an index of an assignment of all the observations to different segments with $\mathbb{C}$ the set of all combinations of possible assignments. Table 2.1 illustrates the concept for a data set with four observations and two segments. There is a total of seven different combinations of assigning the four observations to the two segments and the cell entries in the table give the specific observations assigned to each segment.

c	S_{c1}	S_{c2}
1	1	2,3,4
2	2	1,3,4
$\vdots$	$\vdots$	$\vdots$
7	1,4	2,3

TABLE 2.1: Example of the possible combinations ($\mathbb{C}$) for a data set with four observations

The notation S_{cj} is now introduced to reference all the observations for a given assignment $c \in \mathbb{C}$ and for a given segment index j . For example, from the table $S_{22} = \{1, 3, 4\}$. In addition, $\mathbf{u}_{cj}$ is the centroid of segment S_{cj} . The objective function of the ordinary k-means clustering algorithm can now be stated in generic form as

$$\min_{c \in \mathbb{C}} \sum_{j=1}^K \sum_{i \in S_{cj}} d(\mathbf{x}_i, \mathbf{u}_{cj}). \quad (2.10)$$

2.6.2 Density clustering: k-nearest neighbour

Density based clustering techniques have been researched by Hartigan (1975), Wong and Lane (1981) and many others, and refers to clustering techniques that use probability density estimates to define dissimilarity as well as cluster adjacency. In contrast to the k-means algorithm, these clustering techniques do not start off with a pre-defined number of clusters, but is agglomerative in that it starts off with each observation in its own cluster. Clusters are then systematically combined to minimise the dissimilarity measure used. Computationally, these techniques are significantly more complex than k-means but possess the ability to form clusters of any form and size (Hand, 1997).

In density clustering, clusters are defined as populations of observations which are positioned close together (high density) and in theory are surrounded by areas of lower density.

Likely the best known density clustering methodology is the k-nearest neighbour method (Hartigan, 1975). This method combines observations by estimating the impact it will have on density estimates. In Section 2.4.4 we discussed the KNN classification methodology that is used for predictive modelling. KNN density clustering discussed in this section differs from KNN classification in that it is intended as an unsupervised clustering methodology, with the output being a data set divided into clusters as opposed to estimates of target values as is the case in Section 2.4.4.

Assume $d(\mathbf{x}_i, \mathbf{x}_{K(i)})$ is the Euclidean distance from point $\mathbf{x}_i$ to the K th-nearest observation, where K is a user-specified value indicating the number of observations to be used in density calculation. A density estimate at $\mathbf{x}_i$ is subsequently defined as the number of observations within an m -dimensional sphere (see again Section 2.4 for the definition of m) with its centre at $\mathbf{x}_i$ and with radius $d(\mathbf{x}_i, \mathbf{x}_{K(i)})$. This density estimate ($f(\mathbf{x}_i)$) will be defined as

$$f(\mathbf{x}_i) = \frac{\Gamma\left(\frac{m}{2} + 1\right)}{\pi^{\frac{m}{2}} \times d(\mathbf{x}_i, \mathbf{x}_{K(i)})} \times K \quad (2.11)$$

where $\Gamma()$ is the standard Gamma function.

Observations $\mathbf{x}_i$ and $\mathbf{x}_{i'}$ (with $i' = 1, \dots, n$) is considered adjacent only if $d(\mathbf{x}_i, \mathbf{x}_{i'}) < d(\mathbf{x}_i, \mathbf{x}_{K(i)})$ and $d(\mathbf{x}_{i'}, \mathbf{x}_i) < d(\mathbf{x}_{i'}, \mathbf{x}_{K(i')})$. If the two are adjacent, a new dissimilarity measure, $\delta(\mathbf{x}_i, \mathbf{x}_{i'})$, is computed as

$$\delta(\mathbf{x}_i, \mathbf{x}_{i'}) = \frac{1}{2} \left(\frac{1}{f(\mathbf{x}_i)} + \frac{1}{f(\mathbf{x}_{i'})} \right). \quad (2.12)$$

If they are not adjacent, the dissimilarity is assumed to be infinite. Observations are only compared with other observations that are not in their own segment. Segments are iteratively combined in this methodology (using single linkage, as described below) to minimise the distortion value until the desired number of segments are obtained or only a single segment remains.

Density clustering is only one of many agglomerative (or hierarchical) clustering methodologies that exist. The details of these are available in many texts (see (Lance and Williams, 1967, Massart and Kaufman, 1983, Sokal, 1958, Ward Jr, 1963)). Agglomerative clustering methodologies are mainly distinguished by the measure used to define distance (or dissimilarity) between clusters (also called the linkage function). Some examples of linkage functions are (Lance and Williams, 1967, Massart and Kaufman, 1983, Sokal, 1958, Ward Jr, 1963):

- Single linkage - defines the distance between clusters as the minimum distance between any of the observations within the two clusters.
- Complete linkage - defines the distance between clusters as the maximum distance between any of the observations within the two clusters.
- Average linkage - defines the distance between clusters as the average distance between all of the observations within the two clusters.
- Centroid linkage - defines the distance between clusters as the distance between the centroids of the two clusters.
- Linkage defined by an objective function, like Ward's method, which uses a squared error.

In Appendix A, Section A.1 we discuss an example of where clustering is used for segmentation of predictive models in the industry through the SAS[®] Enterprise Miner[™] software. The software provide three clustering options which are the Average, Ward and Centroid methodologies discussed above.

2.6.3 Wong's clustering method

Since many predictive models can now be developed on relatively large data sets (> 1000 observations and 20 or more characteristics), segmentation algorithms needs to be viable for such large data sets. This virtually eliminates the more common kernel based density methods like k-nearest neighbourhood (Wong and Lane, 1981) due to its complexity (see Section 2.6.2). Wong (1982) proposed a methodology that combines the speed of k-means with the advantages of density based clustering techniques.

Wong's clustering method consists of two stages (Wong, 1982):

1. A preliminary clustering analysis is performed using a k-means algorithm with k much larger than the number of final clusters required.
2. The k clusters formed in stage one is analysed and combined based on density clustering dissimilarities until the required number of clusters is formed or only a single cluster remain.

Preliminary clusters s_{cr} and s_{ct} are considered adjacent if the midpoint between the centroids $\mathbf{u}_{cr}$ and $\mathbf{u}_{ct}$ is closer to either $\mathbf{u}_{cr}$ or $\mathbf{u}_{ct}$ than to any other preliminary cluster mean based on Euclidean distance. Each preliminary cluster therefore has only one potential cluster that it can be combined with. The cluster pair which is combined at each step is the pair for which the following density based dissimilarity measure is the minimum (see (Wong, 1982) for further detail and derivation):

$$\delta(s_{cr}, s_{ct}) = \begin{cases} \frac{\left(\sum_{i \in s_{cr}} d(\mathbf{u}_{cr}, \mathbf{x}_i) + \sum_{i \in s_{ct}} d(\mathbf{u}_{ct}, \mathbf{x}_i) + \frac{1}{4} (|s_{cr}| + |s_{ct}|) d(\mathbf{u}_{cr}, \mathbf{u}_{ct}) \right)^{\frac{m}{2}}}{(|s_{cr}| + |s_{ct}|)^{1 + \frac{m}{2}}} & \text{if } s_r || s_t \\ \infty & \text{otherwise,} \end{cases} \quad (2.13)$$

where the terminology $|s|$ is used to represent the number of observations in segment s and $s_r || s_t$ is used to indicate that s_r is adjacent to s_t .

In Section 7.5 we will adapt Wong's clustering method for the purposes of semi-supervised segmentation. We will adjust some of the assumptions of the algorithm, like cluster adjacency and cluster initialisation, but we will leverage the balanced advantages of Wong's approach (speed vs. complexity) to analyse its effectiveness as a semi-supervised algorithm for predictive modelling.

As an industry example of where clustering is used for segmenting predictive models we discuss the SAS[®] Enterprise Miner[™] software in Appendix A (see Section A.1). Enterprise Miner[™] 12.1 offers a "Cluster" node within its "Explore" tab which enables the user to perform cluster analysis on a data set.

Density clustering is also available in the software under its MBR node, but this node is designed to serve as a non-linear predictive model, and not an unsupervised clustering methodology, so it requires the input of a target variable.

2.7 Supervised techniques

Supervised learning is the term that is referred to in *machine learning* for inferring a function from *labeled* data points through some iterative process. This is also known as predictive modelling in financial and other fields. The key difference between unsupervised and supervised learning is the presence of a target variable, or labels. The focus of supervised techniques are to minimise some error term based on the target variable by using independent characteristics in a functional format. This differs from unsupervised techniques in that the objective function is not based on the characteristics that are used. Supervised techniques are also often used to segment populations for predictive model development. For this reason we will explore supervised techniques further in this section.

2.7.1 Decision trees

In Section 2.4.5 we discussed the way in which decision trees work. Decision trees are discussed here with the background of it being used to segment data sets for predictive modelling.

Decision trees (originating from classification and regression trees (CART) (Breiman et al., 1984)), are one of the most common supervised learning techniques used for model segmentation (Anderson, 2007). They belong to a modelling (or supervised learning) framework called non-parametric techniques. These techniques have the useful attribute of requiring almost no assumptions about the underlying data (Anderson, 2007). Decision trees use recursive partitioning algorithms (RPA) to classify observations into homogenous groups, the groups are formed through repeated attempts at finding the best possible split on the previous branch. They have been developed, applied and thoroughly researched in various fields, like statistics (Breiman et al., 1984), artificial intelligence (Safavian and Landgrebe, 1991) as well as machine learning (Quinlan, 1993).

There are a number of advantages of decision trees over other supervised classification techniques (Anderson, 2007). Decision trees' non-parametric nature enables it to identify patterns in data, as well as interactions very successfully. Results are simple to interpret and implement, especially for relatively small trees. Implementing the logic is computationally simple once the tree has been developed and it is able to identify extreme groups (very high or very low target probabilities) well.

However, decision trees also has some disadvantages. When used for predictive modelling on data sets where interactions are not prevalent, or where segmentation addresses interaction sufficiently, regression based models tend to provide better accuracy (Anderson, 2007). Since the original population is divided with the addition of each extra dimension, population sizes diminish quickly. This can result in small, meaningless segments when used for segmentation, or overfit when used as predictive models (Thomas, 2009). To avoid the risk of overfitting (or small segments), decision trees often need to be pruned back to restrict the number of end nodes, which reduces granularity, and may mean that important characteristic-interactions are not identified (Pearl, 1984). Lastly, because decision trees optimise each split locally, it has been proven that they do not necessarily guarantee the return of a globally optimal solution (Hyafil and Rivest, 1976).

As mentioned, the first decision trees were based on the CART methodology. Another popular methodology for splitting is the CHAID (chi-squared automatic interaction detection, see Kass (1980)) methodology. CART decision trees can only do binary splits, whilst CHAID decision trees can split into more than two nodes.

In summary, because of its ability to capture interactions in independent characteristics, decision trees are often used when segmenting for predictive model development. However, the "greedy" way in which it selects splits may not always result in the best possible segmentation and optimising similarity from an independent characteristics perspective is not considered.

In its white paper named "Using segmented models for better decisions" (Fico, 2014), Fair Isaac Corporation (FICO) describe the technique they developed to segment for predictive model development, which is based on decision trees. This is a practical example of how segmentation is used in the industry. Section A.2 in Appendix A provides more detail of how FICO applies decision trees in practice.

2.8 Semi-supervised clustering

In comparison to traditional clustering, semi-supervised clustering is a relatively new concept (Grira et al., 2005). In most applications of semi-supervised clustering, the

aim is to assist or guide the unsupervised clustering algorithm in finding the correct clusters by providing pairwise constraints or class labels on observations where cluster relationships are known. In these applications, the class or cluster assignment cannot be fully supervised as the quantity of information known is too little to generalise well (Bair, 2013, Bilenko, 2004, Grira et al., 2005).

Although the available data could be specific labels or classes indicating which segment the observation belongs to (Basu et al., 2002), the most common constraints are in the form of "must-link" (i.e. identifying observations that *has to be* in a similar cluster) and "cannot-link" (i.e. identifying observations that *cannot be* in a similar cluster) constraints (Bair, 2013, Cohn et al., 2003, Klein et al., 2002, Xing et al., 2003). These constraints have an impact on how clusters are formed, since they indicate which observations should be combined in the same cluster, and which should not be in similar clusters.

On a high level, semi-supervised clustering is done using one of two methods (Bair, 2013, Bilenko, 2004, Grira et al., 2005):

- Similarity adapting methods.
 - These methods use existing clustering algorithms to perform clustering, but alters the objective function values (for instance, Euclidean distance in k-means clustering) of specific data points in response to the known constraints. Once the adjustments have been made, the clustering algorithm is performed without any change to the methodology on the altered data. The objective functions (or dissimilarity measures) researched for these techniques include Kullback–Leibler (KL) divergence (Cohn et al., 2003), Euclidean distance (Klein et al., 2002) and Mahalanobis distance (Xing et al., 2003).
- Search-based.
 - Search-based methods alters the clustering algorithm used by either penalising clusters that do not comply through an additional term (or distortion function) which is added to the objective function (Demiriz et al., 1999), or initialising clusters through analysis of the labeled data only (Basu et al., 2002), or simply by forcing compliance to the constraints (Wagstaff and Cardie, 2000).

Although most of the work done on semi-supervised clustering can be divided into the two frameworks mentioned above, Bilenko (2004) integrates the two approaches, and shows that results can be improved by doing this. All studies show that the addition

of labeled data has advantages to either the speed with which results are obtained, the accuracy of the final result, or both.

The semi-supervised segmentation technique developed in this study will be based on the addition of a distortion function which penalises the objective function found within the search-based algorithms in semi-supervised clustering.

2.9 Supervised clustering

We next explore a more recent evolution of clustering that has even more similarities to the semi-supervised segmentation algorithm proposed in this study. Supervised clustering was formally introduced by Eick et al. (2004). They define the goal of supervised clustering as the quest to find "class uniform" clusters with high probability densities, otherwise known as *label purity*. They contrast supervised clustering with semi-supervised clustering in that the assumption is made that all observations are "labeled", or have a target variable assigned. This is opposed to semi-supervised clustering which typically has only a small number of labeled instances. We will explore some of the work performed in the line of supervised clustering here.

Eick et al. (2004) modifies four clustering algorithms to consider target variables. The first they call SRIDHCR for "single representative insertion/deletion steepest decent hill climbing with randomized restart", which is a greedy algorithm and largely based on the K-medoids clustering methodology (see (Zeidat and Eick, 2004) for more detail on K-medoid clustering). They also consider an algorithm they call SPAM (supervised partitioning around medoids), based on the PAM clustering algorithm. The other two algorithms they compare are the "top down splitting algorithm", which is meant to create clusters quickly and a clustering algorithm based on evolutionary algorithms (SCEC for "Supervised Clustering using Evolutionary Computing"). Their results show that the SCEC algorithm performs well consistently, although being computationally intensive.

Additional studies have been published that fits the adopted definition of supervised clustering, and was performed before Eick et al. (2004). Sinkkonen et al. (2002) proposes a method they call discriminate clustering which minimises distortion within clusters. In line with the supervised clustering goal put forward by Eick et al., the goal of the algorithm is to produce clusters with high label purity. Tishby et al. (2000) focuses on minimising information loss relating to labels between clusters which they call the "information bottleneck method". They have since applied the method to a number of applications (Harremoës and Tishby, 2007).

Subsequent work was done by Ye et al. (2008) specifically on the k-means algorithm. They propose an algorithm (which they call DisKmeans) that simultaneously solves both linear discriminant analysis (LDA) and k-means. The methodology requires intensive optimisation at each iteration which seems computationally intensive, but their study has not elaborated on the performance implications of the method.

The work that seems to have the most similarities with the K-means semi-supervised segmentation algorithm, proposed in this study, is proposed by Peralta et al. (2013). They propose an algorithm they call LK-Means. The algorithm modifies the objective function of the standard k-means algorithm to consider an additional, supervised outcome with the aim of encouraging label purity within different clusters. The final objective function of the LK-Means algorithm is therefore comprised of a supervised term as well as an unsupervised term. In line with the distortion weight in this study (introduced in Chapter 3), the algorithm uses a weight which balances the supervised and unsupervised clustering scores.

The terminology, "distortion weight" will be used in this thesis (see also Chapter 3) to refer to a weight which determines how much emphasis is placed on either the supervised or unsupervised element of an objective function.

As will be discussed in Chapter 3, the semi-supervised segmentation (SSS) algorithm proposed in this study can be applied to k-means in a number of ways. In order to enable meaningful and concise comparisons with the methodologies explored here, we will refer to the specific application of our proposed semi-supervised segmentation algorithm to k-means as "SSSKM" in this chapter. We will elaborate on the exact detail and variations of SSSKM in Chapter 3.

The main difference of the LK-Means algorithm proposed by Peralta et al. (2013) to SSSKM is that the supervised score is still a function of Euclidean distance. The supervised score comprises of a prior factor (which represents the confidence of a specific label (target value) - i.e. the label purity within the cluster) with the distance between a new observation and the mean of observations in the same cluster with the same label. The term "label" is used in their paper to refer to a target variable that is not continuous, but consists of two or more classes.

To facilitate representing the objective function of the LK-Means algorithm mathematically, we expand the notation introduced in Section 2.6.1 to introduce S_{cjl} , to reference all the observations for a given assignment $c \in \mathbb{C}$, for a given segment index j and a given label l . Similarly $\mathbf{u}_{cjl}$ represents the mean, or centroid of S_{cjl} . For this algorithm, the assumption is made that the labels (or target variable values) take on L discrete

values and are not continuous. The objective function of the LK-Means algorithm that is to be minimised then becomes

$$\min_{c \in \mathbb{C}} \left[w \sum_{j=1}^K \sum_{l=1}^L \sum_{i \in S_{cjl}} d(\mathbf{x}_i, \mathbf{u}_{cjl}) \nu_{cjl} + (1 - w) \sum_{j=1}^K \sum_{i \in S_{cj}} d(\mathbf{x}_i, \mathbf{u}_{cj}) \right], \quad (2.14)$$

where ν_{cjl} is the ratio of the number of observations assigned to cluster j with label l divided by the number of total observations assigned to cluster j . This ratio represents the "confidence" of label l in cluster j . w is the distortion weight that is also introduced in Chapter 3 and adjusts the influence of the supervised element in the equation with values between 0 and 1.

In order to ensure no mathematical errors occur due to some labels not being assigned to a specific cluster, the authors make use of Laplace smoothing (Peralta et al., 2013) with a small constant which they interpret as the global uncertainty about the label of an observation.

Tables 2.2 and 2.3 summarise some of the key similarities and dissimilarities between the LK-Means algorithm proposed by Peralta et al. (2013) and SSSKM (proposed in Chapter 3 of this thesis). Since the details of SSSKM is only provided in Chapter 3, the table also contains references to the sections where relevant characteristics of SSSKM is discussed in this thesis.

Similarities	Reference sections
Both algorithms make use of the iterative update and assignment steps of k-means	3.3, 3.9, 3.10
Both algorithms supplement the standard k-means objective function with an additional supervised element	3.2, 3.9.3, 3.9.5
Both algorithms uses a distortion weight that enables the adjustment of the influence of the supervised element on the final outcome	3.2, 3.9.5

TABLE 2.2: Similarities between LK-Means and the proposed K-means semi-supervised segmentation algorithm

Due to the fact that the LK-Means algorithm shares so many similarities with the SSSKM algorithms proposed in this study, we augment the LK-Means algorithm in Chapter 7 so that it can be used to segment data sets for predictive model development. Please refer to Section 7.10 for further detail on this.

Differences	Reference sections
LK-Means relies on Euclidean distance in the supervised element, whilst the supervised element of SSSKM is a function that measures target rate separation, and is not based on Euclidean distance	2.11, 3.2, 3.9.3
In SSSKM, the supervised and unsupervised elements are independently normalised, which is not required with LK-Means	3.9.4
The supervised element in SSSKM can be any function that measures target rate separation whilst the supervised element in LK-Means is fixed	2.11, 3.2, 3.9.3
The flexibility of the supervised element in SSSKM allows the algorithm to deal with continuous targets whilst LK-Means assumes discrete targets	2.11.2
Whilst LK-Means uses Laplace smoothing to cater for unassigned labels, SSSKM deals with unassigned labels by making assignments based on Euclidean distance only	3.9.2

TABLE 2.3: Differences between LK-Means and SSSKM

2.10 Comparison between semi-supervised segmentation, supervised and semi-supervised clustering

In the previous sections we have provided some background on methodologies and work related to the semi-supervised segmentation (SSS) algorithm proposed in this study. Three approaches have been mentioned (semi-supervised clustering, supervised clustering and semi-supervised segmentation), and due to the overlap in these techniques, more clarity on how these are distinguished may be required. This section will briefly summarise the similarities and dissimilarities between the methodologies. This section is summarised in Table 2.4.

The semi-supervised segmentation algorithm is similar to semi-supervised clustering algorithms that alter the objective function to either reward or penalise constraints (search - based methods). The main difference is in the fact that the semi-supervised segmentation algorithm has target values (or labels) for all observations, and the semi-supervised clustering algorithms mostly do not. The way in which the constraints are applied also differ between these approaches. Semi-supervised clustering algorithms usually define their constraints as "must-link" and "cannot-link" constraints, whilst a distortion function is used for semi-supervised segmentation - which can be any function that measures the target separation. The distortion weight used in semi-supervised segmentation is also not present in most semi-supervised clustering methods.

As mentioned, *supervised clustering* is even more closely related to semi-supervised segmentation when directly compared to semi-supervised clustering. The first similarity which is not present with semi-supervised clustering is the assumption that all observations has "labels" or a value for the target variable. Some work done (Peralta et al., 2013) also implements some form of a distortion function with a distortion weight, which alters the objective function of standard k-means.

However, even though many similarities may exist, the ultimate goal of both semi and fully supervised clustering is very different from the ultimate goal of semi-supervised segmentation. Because the output of a clustering analysis is seen as the final result, the number of "known" labels (or target variable values) will usually be similar or close to the number of clusters that are formed (Eick et al., 2004) for both supervised and semi-supervised clustering. Unlike these clustering techniques, the goal of a segmentation analysis is simply to provide the best segmentation of populations in order to facilitate further predictive modelling, and therefore the number of segments has no set limit. Furthermore, all clustering algorithms assume that the target variable will only assume a specific number of values (labels), whilst segmentation for predictive modelling can be performed on a continuous target variable. As long as the distortion function used supports continuous target variables, the algorithm will work and the goal can be achieved.

We have named the methodology proposed here "semi-supervised segmentation", even though it shares many similarities with supervised clustering. The reason is that both supervised and unsupervised segmentation already exists (for instance decision trees are supervised segmentation methodologies and unsupervised clustering is an example of unsupervised segmentation). The term semi-supervised segmentation therefore describes aptly the goal of this methodology - which is to provide a segmentation methodology that considers both the target variable and the explanatory variables when segmenting for predictive modelling.

Another point to note is that the term "semi-supervised segmentation" is already used in the field of computer vision and pattern recognition. These applications mainly aims to assist with identifying or grouping spatial images or objects based on their perceivable content. The principles used are similar to semi-supervised clustering described in this thesis, and has been used to segment photo images (Houhou et al., 2009, Socher and Fei-Fei, 2010, Song and Fan, 2002, Wang et al., 2012), videos (Badrinarayanan et al., 2013), ultrasound images (Ciurte et al., 2014), spine images (Haq et al., 2015) and even peptide mass segmentation (used in fingerprint identification) (Bruand et al., 2011). The use of semi-supervised segmentation in computer vision and pattern recognition should

not be confused with its use in the context of predictive modelling, as introduced in this thesis.

Criterion	Semi-supervised clustering	Supervised clustering	Semi-supervised segmentation (SSS)
Target variable value known	Some	All	All
Incorporation of the target variable	Must-link and cannot-link constraints	Simultaneous optimisation or adjustment of the objective function	Adjustment of the objective function
Ability to adjust influence of target variable	No	Some	Yes
Target variable type	Categorical	Categorical	Categorical or continuous
Typical number of clusters / segments	Ideally close to number of distinct target values	Ideally close to number of distinct target values	As many as required for improved predictive modelling
Goal	Improving cluster accuracy	Improving cluster accuracy	Segmenting for predictive modelling

TABLE 2.4: Comparison between semi-supervised clustering, supervised clustering and semi-supervised segmentation

In summary we therefore define the following for the purposes of this thesis:

- Unsupervised clustering.
 - The partitioning of observations into separate groups (called clusters), by means of optimising a pre-defined objective function, which does not incorporate any known outcomes (i.e. a label or target variable), with the end goal of ensuring observations in the same cluster are as similar to one another as possible, whilst those in different clusters are as different from one another as possible.
- Semi-supervised clustering.
 - The partitioning of observations into separate groups (called clusters), by means of optimising a pre-defined objective function, which incorporates known outcomes (i.e. a label or target variable) for some observations, usually

through must-link, or cannot link-constraints, with the end goal of ensuring observations in the same cluster are as similar to one another as possible, whilst those in different clusters are as different from one another as possible.

- Supervised clustering.
 - The partitioning of observations into separate groups (called clusters) by means of optimising a pre-defined objective function, which incorporates known outcomes (i.e. a label or target variable) for all observations, usually through a distortion function, with the end goal of ensuring label (or target variable) purity is as high as possible.
- Semi-supervised segmentation.
 - The partitioning of observations into separate groups (called segments) by means of optimising a pre-defined objective function, which incorporates known outcomes (i.e. a target variable) for all observations, through a distortion function, with the end goal of developing predictive models on the segments that optimise overall predictive accuracy of the target variable.

2.11 Measuring target separation

As mentioned in Section 1.2, the primary objective of this thesis is to develop and define a semi-supervised segmentation algorithm that will consider both the explanatory variables and the target variable when segmenting for predictive modelling. In order to measure the success of the algorithm in considering these, we need the most appropriate measurement statistics. In this section we focus on finding a measure for how well target separation was obtained.

Target separation (or impurity) measurements will be defined in this study as any technique that quantifies how populations differ with regards to the variable that will be used as the dependent, or target variable in a model development. Popular techniques include Gini impurity (Breiman, 1996b), twoing (Breiman et al., 1984), information gain (Cover and Thomas, 2006), information value (IV) (Kullback, 1958), chi squared (Hand, 1997), R squared (Siddiqi, 2006) and many other. We will focus on two metrics that are popularly used in predictive model development namely IV and chi-squared. These are discussed in more detail in the next sections.

2.11.1 Information value

As discussed in Sections 2.8 and 2.10, in this thesis we will require a distortion function that will serve as the supervised element of the semi-supervised segmentation algorithm (see Equation 3.2). An example of such a distortion function is the information value (IV) of a specific population split (see Kullback (1958)). The IV is a measure of the separation, or impurity, of the target variable between segments if the target variable ($\mathbf{y}$) is binary. Consider a specific segment assignment $c \in \mathbb{C}$ and let $P_{cj}^T = \sum_{i \in S_{cj}} y_i / \sum_{l=1}^K \sum_{i \in S_{cl}} y_i$ be the proportion of events ($y_i = 1$) of segment S_{cj} relative to the total population. Let $P_{cj}^F = \sum_{i \in S_{cj}} (1 - y_i) / \sum_{l=1}^K \sum_{i \in S_{cl}} (1 - y_i)$ be the proportion of non-events ($y_i = 0$) of segment S_{cj} relative to the total population. IV is then defined as

$$\varphi(c) = \begin{cases} -\sum_{j=1}^K \left[\left(P_{cj}^T - P_{cj}^F \right) \times \log \left(\frac{P_{cj}^T}{P_{cj}^F} \right) \right] & \text{if } 0 < P_{cj}^T < 1 \\ 0 & \text{otherwise.} \end{cases} \quad (2.15)$$

The advantage of the IV is that it does not often assume very small or very large values. This makes it intuitive and easy to use and interpret.

IV has two disadvantages:

- It is not always defined. Each segment needs at least one event and one non-event before IV can be defined.
- It is only defined for binary target variables, which means continuous target variables cannot be considered when IV is used.

2.11.2 Chi-squared

The chi-square statistic is a very popular methodology used to define target separation. A good example of this is its use in decision trees, called chi-squared automatic interaction detection (CHAID) (Hand, 1997). The chi-square statistic compares the observed total target value in a segment to an expected value. In this application of the chi-square test, the goal is to obtain a measure for target separation (or impurity), which means the null hypothesis is that the target rate (or average target value if the target is continuous) in each segment is equal. The chi-squared statistic is calculated as

$$\chi^2 = \sum_{j=1}^K \frac{\sum_{i \in S_{cj}} y_i - \frac{1}{n} \sum_{h=1}^n y_h |S_{cj}|}{\frac{1}{n} \sum_{h=1}^n y_h |S_{cj}|}, \quad (2.16)$$

where:

- n is the number of observations in the input data set as defined in Section 2.6.1.
- K is the number of segments over which χ^2 is calculated as defined in Section 2.6.1.
- y is the target variable *and can be either binary or continuous*.
- The terminology $|S|$ is used to represent the number of observations in segment S .

The final measure of target separation can be obtained by using the chi-square probability density function to find the probability that the χ^2 calculated in Equation 2.16 could be a random deviation from the population average given the number of segments (degrees of freedom). If chi-square is used in semi-supervised segmentation, then the distortion measure $\varphi(c)$ for each observation $\mathbf{x}_i$ and each segment S_{cj} (with i assigned to S_{cj} in each case) can be defined as

$$\varphi(c) = \frac{\chi^{2(\frac{K-1}{2})-1} e^{-\frac{\chi^2}{2}}}{2^{\frac{K-1}{2}} \Gamma(\frac{K-1}{2})}, \quad (2.17)$$

where $\Gamma(\frac{K-1}{2})$ denotes the standard gamma function of $\frac{K-1}{2}$.

Although using the null hypothesis probability as distortion function may seem most intuitive, it is also possible to use the standard chi-squared function from Equation 2.16 directly as distortion function. Since the result of Equation 2.17 can be very small when there are large target separation between segments, Equation 2.16 is used as distortion function for practical reasons in the newly proposed semi-supervised segmentation (SSS) algorithm using chi-squared (see Chapter 5). In this scenario $\varphi(c) = \chi^2$.

Using the chi-squared as distortion function has two main advantages:

- It is always defined within a segmentation scheme, unlike IVs that are not defined until each segment has been assigned at least one event and one non-event.
- It is not only defined for binary variables, but also for continuous variables - this allows segmentation for predictive modelling of non-binary (continuous) targets.

It also has some disadvantages:

- In cases where the target rates between segments are big, the probability calculated by Equation 2.17 may tend to zero, making comparison between segmentation schemes difficult.
- It is sensitive to small data sets, and may not provide statistically robust results in cases where data sets are small.

2.12 Measuring cluster validity

Because of the wide-spread use of unsupervised clustering in various applications, methodologies (or indices) for measuring the success of a clustering exercise have been developed since the early 1970's (Caliński and Harabasz, 1974, Davies and Bouldin, 1979, Dunn, 1973). These are therefore well known, and well established.

Halkidi et al. (2002*a,b*) explores methodologies for validating clusters, and divides it into three general approaches:

- External validity indices compare results from a clustering exercise to results obtained externally, either through prior knowledge, or subjective interpretation by the user (recent uses for these methodologies include uses in supervised and semi-supervised clustering, see Section 4.3). Examples of such statistics include the Rand index (Hubert and Arabie, 1985), Jaccard similarity coefficient (Jaccard, 1901, 1912), and the Fowlkes and Mallows index (Fowlkes and Mallows, 1983).
- Internal validity indices uses quantities and features that are inherent to the data set. Examples of validity indices using this approach are the cophenetic correlation coefficient (Rohlf and Fisher, 1968), the cubic clustering criterion (Sarle, 1983) (see also Section A.1), and the pseudo-f statistic (or CH value) (Caliński and Harabasz, 1974).
- Relative validity indices compares different cluster solutions from a single clustering algorithm, finding the cluster inputs that results in the best clusters. As an example, these are used to find the most optimal number of clusters when clustering a data set using k-means. Examples are the minimum description length (MDL) (Rissanen, 1983) and Schwartz's bayesian information criterion (BIC) (Schwarz, 1978).

In this study we will use the pseudo-f statistic developed by Caliński and Harabasz (1974), also known as the CH measure, when we need to measure the success of unsupervised segmentation. The pseudo-f statistic is not linked to any specific clustering

criterion, and is well suited for our requirements to measure the success of the "unsupervised" element of the semi-supervised algorithm we propose. The CH measure is defined as the ratio of the separation (or "between cluster sum-of-squares") to the cohesion (or "within cluster sum-of-squares"),

$$\frac{\sum_{j=1}^K |S_{cj}| (\mathbf{u} - \mathbf{u}_{cj})}{\sum_{j=1}^K \sum_{i \in S_{cj}} (\mathbf{x}_i - \mathbf{u}_{cj})}, \quad (2.18)$$

where $\mathbf{u}$ is the mean, or centroid of the entire data set. The advantage of the pseudo-f statistic is that it is transparent, and can be easily applied to any clustering algorithm. A disadvantage is that it can be sensitive to outliers in the data.

2.13 Conclusion

In this section we undertook a literature study on some areas pertinent to this study.

We achieved the following:

- Positioned the background of predictive modelling and segmentation within predictive modelling in the industry.
- Discussed and defined a number of predictive modelling techniques employed in this thesis.
- Distinguished between supervised and unsupervised segmentation techniques.
- Discussed the main segmentation techniques currently in use for predictive modelling.
- Discussed supervised and semi-supervised clustering as well as the similarities and dissimilarities to semi-supervised segmentation.
- Explored some previous work performed that has similarities with this study, as well as indicated how it differs from this study.
- Discussed some popular techniques used to measure target separation.
- Discussed some popular techniques to measure cluster validity.

The next chapter introduces the newly proposed semi-supervised segmentation algorithm as applied to k-means. First the algorithm's global objective function is defined and then the algorithm is broken into ten steps, each with its own unique role.

Chapter 3

Semi-supervised segmentation as applied to k-means (SSSKM)

3.1 Introduction

Chapter 2 details some of the known methodologies used for segmentation along with the concepts of semi-supervised and supervised clustering. The concept of semi-supervised segmentation, as a methodology to divide (or partition) observations into separate segments with the aim of developing predictive models on these segments independently (specifically applied on the k-means algorithm), is now introduced. The primary objective of this research is to provide an alternative methodology to those currently in use that allows the user, when segmenting for predictive modelling, to not only consider the explanatory variables (as is the case with unsupervised techniques such as clustering) or the target variable (as is the case with supervised techniques such as decision trees), but to be able to optimise both simultaneously during the segmentation exercise.

In Section 2.10 the similarities and dissimilarities of semi-supervised clustering, supervised clustering and semi-supervised segmentation are discussed. In a sense this application of semi-supervised segmentation can be viewed as a multi-target optimisation problem that utilises clustering techniques as its core vehicle of execution. The fact that it uses clustering techniques as its core vehicle of execution also sets it apart from other known optimisation techniques, like evolutionary algorithms (Bäck and Schwefel, 1993, Coello et al., 2002, Fonseca and Fleming, 1995). The ultimate goal of semi-supervised segmentation, as opposed to supervised or semi-supervised clustering, is not final object classification, but rather an informed separation of observations into groups on which supervised classification (or predictive modelling) can be performed, i.e. segmentation for predictive modelling.

3.1.1 Research contributions

This chapter will do the following:

- Define the global objective function of the semi-supervised segmentation algorithm applied to k-means (SSSKM).
- Describe the operation of the algorithm in ten steps.
- Define the local objective function for SSSKM that is performed for each observation.
- Introduce the concept of overfit for SSSKM and how overfit is dealt with in this algorithm.

We, therefore, address the primary objective of this study in this chapter, which is the development of a semi-supervised segmentation algorithm that can serve as an alternative to current segmentation algorithms available.

3.2 The objective function

Equation 2.10 defines the objective function for the standard k-means algorithm as

$$d(\mathbf{x}_i, \mathbf{u}_j) = \|\mathbf{x}_i - \mathbf{u}_j\|^2. \quad (3.1)$$

We will now define an objective function for the semi-supervised segmentation technique using k-means. Equation 2.10 is generic enough to represent the objective function of any clustering algorithm that aims to minimise inter-segment distances using a predefined distance measure. Furthermore it can easily be extended to include “supervision”

Recall from Section 2.11 (see Equation 2.15), that $\varphi(c)$ denotes a distortion function that measures the separation of the target variable $\mathbf{y}$ between the segments $S_{c1}, S_{c2}, \dots, S_{cK}$ for a given assignment $c \in \mathbb{C}$. Examples of such distortion functions were provided in Sections 2.11.1 and 2.11.2. Let w be a weight ($0 \leq w \leq 1$) of how much the objective function of the clustering algorithm is penalised by the distortion function (the distortion weight). The proposed optimisation problem for the SSSKM algorithm, taking inter-segment distances into account is (Breed et al., 2013)

$$\min_{c \in \mathbb{C}} \left[w \varphi(c) + (1 - w) \sum_{j=1}^K \sum_{i \in S_{cj}} d(S_{cj}, \mathbf{x}_i) \right]. \quad (3.2)$$

As can be seen, the algorithm minimises the sum of the Euclidean distances of all observations to its segment centre (the unsupervised part), as well as the distortion function (the supervised part). This means that both the explanatory variables as well as the target variable are considered during segmentation. The weight, w , can be adjusted. We will see later in the thesis that the optimal value of w differs depending on the characteristics of the data set analysed, and we therefore propose an iterative approach to finding the optimal value of w (this is also the case for the optimal number of segments, K).

The next few sections will elaborate in a pragmatic way on how this objective function have been implemented for this study.

3.3 Design of the algorithm

This section will provide an overview of how the standard k-means algorithm is adjusted for semi-supervised segmentation. The algorithm is developed in base SAS[®] software and was designed in such a way as to aid model developers in identifying potential segmentation for predictive modelling. Specific focus was therefore put on automation and utility for this use. One can therefore see the final result as a K-means semi-supervised segmentation solution or toolkit.

The algorithm consists broadly of ten separate functional parts as illustrated in Figure 3.1. The steps fulfil the following functions:

1. *Variable identification.* This step identifies the specific explanatory variables in the input data set $\mathbf{x}_i = \{x_{i1}, x_{i2}, \dots, x_{im}\}$ to be used in the SSSKM algorithm.
2. *Segment seed initialisation.* This step initialises the segment centroids $\mathbf{u}_j = \{u_{j1}, u_{j2}, \dots, u_{jm}\}$ by selecting their initial coordinates at random.
3. *Initial data set and variable preparation.* This step prepares the data set for the iterative process to follow by initialising the required parameters and arranging the input as necessary.
4. *Assignment.* This is the first step of the iterative process where every observation $\mathbf{x}_i = \{x_{i1}, x_{i2}, \dots, x_{im}\}$ in the input data set is assigned to a specific segment $S_{c1}, S_{c2}, \dots, S_{cK}$ based on the outcome of Equation 3.2.
5. *Assignment evaluation and update.* This step evaluates the validity of the assignments $c \in \mathbb{C}$ for segments $S_{c1}, S_{c2}, \dots, S_{cK}$, ensuring every segment has

observations assigned to it and recalculates the position of the segment centroids $\mathbf{u}_j = \{u_{j1}, u_{j2}, \dots, u_{jm}\}$.

6. *Summarize and log step statistics.* This step logs required detail of the *Assignment evaluation and update* step for analysis purposes.
7. *Randomize data set.* This step sorts observations $\mathbf{x}_i = \{x_{i1}, x_{i2}, \dots, x_{im}\}$ randomly to reduce the probability of local optima by ensuring the order in which observations are considered is different with every iteration.
8. *Evaluate stopping criteria.* This step compares the position of the new centroids $\mathbf{u}_j = \{u_{j1}, u_{j2}, \dots, u_{jm}\}$ to the previous position to determine how much the centroids have moved. If the combined distance moved is no longer reducing, the iterative process is stopped, otherwise it is repeated.
9. *Overfit evaluation and smoothing.* This step uses a k-nearest-neighbour smoothing algorithm to assign the observations of a validation set to the segments as determined by the iterative process performed on the build data set. This ensures that the results evaluated are not overfitted.
10. *Final evaluation and result logging.* This step evaluates the way in which both the separation of the target variable through $\varphi(c)$ and the explanatory variables through $d(\mathbf{x}_i, \mathbf{u}_{cj})$ were achieved on the validation data set. Results are logged for further analysis.

More detail will be provided on these steps throughout the rest of this chapter. The code for the algorithm (written in SAS[®] software) is available from the author of this thesis on request.

3.4 Inputs into the K-means semi-supervised segmentation algorithm

The main inputs required for a single run of the SSSKM algorithm to function are the following:

- An input data set with m numeric explanatory variables ($\mathbf{x}_i$) and a continuous or binary target variable, $\mathbf{y}$.
- The number of segments required, K .
- The value of the distortion weight, w .

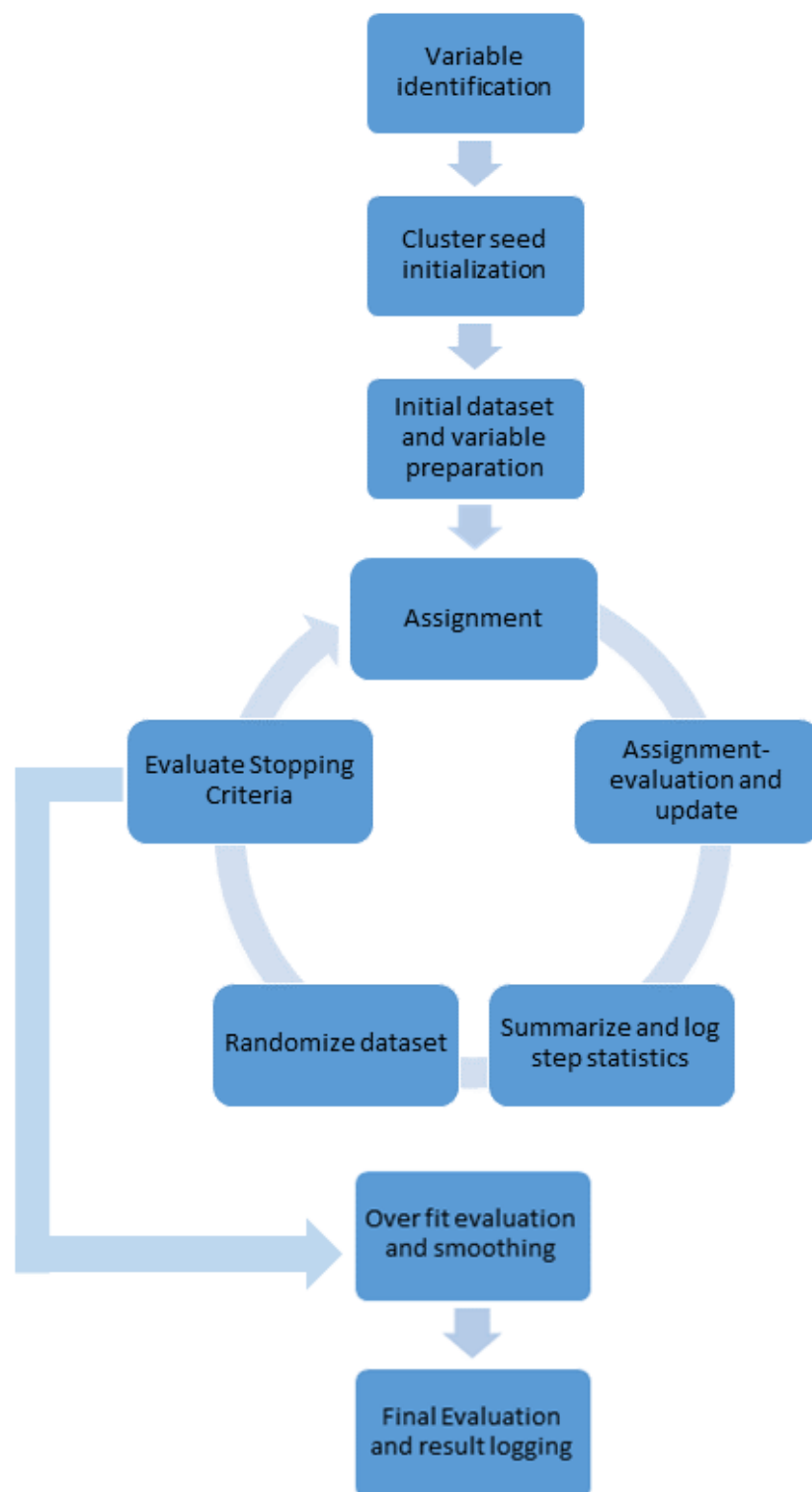


FIGURE 3.1: Functional flow of the single k-means semi supervised segmentation algorithm

- Some operational commands indicating assumptions (like the choice of distortion function to use) and controlling process flow.
- Location and names of the tables containing all summary statistics to be generated.

As indicated in the list above, there are several inputs which control process flow as well as indicate key assumptions to be made within the algorithm. We discuss the key assumptions made within the algorithm in detail throughout the rest of this chapter, but since the operational inputs are not vital to the core functionality of the algorithm, we will not elaborate on the detail of these inputs.

3.5 Repeating the algorithm to measure local optima convergence

In any general unsupervised k-means analysis, the iterative process is terminated when no changes in segment assignment are made from one step to the next (Friedman et al., 2001). Even though this could be interpreted as complete convergence, many studies have shown that the k-means algorithm is still susceptible to local optima and could arrive at different segments depending on the starting coordinates (Pelleg and Moore, 2000, Steinley, 2001, 2004). Due to the additional information being utilised in the SSSKM approach (with the order in which specific points are considered playing a role in segment assignment) complete convergence (with no observations being re-assigned) is very unlikely. For this reason, a different termination processes need to be followed which will be discussed in more detail in Section 3.13.

As a consequence of both the local optima issue as well as the lack of complete convergence of the SSSKM algorithm, a single performance of the algorithm illustrated in Figure 3.1 cannot be considered adequate to encapsulate the full range of possible segments for a given data set and a given value of the distortion weight w . In line with other studies (Steinley, 2001), we will analyse the results of repeated runs of the algorithm to increase the odds of finding a *globally* optimal solution.

More detail and further analysis on this is discussed in later sections.

3.6 Step 1: Variable identification

The first step of the algorithm simply reads the names of the m explanatory variables denoted in this thesis as $X_1 \cdots X_m$ into an array which allows the algorithm to identify

and use them. This is therefore a very simple step, however the assumptions made by the algorithm regarding the characteristics of the values in the variables require further clarification to ensure optimal use.

Since the algorithm is based on the k-means clustering algorithm, it assumes that input variables are numerical and continuous. Furthermore, due to its isotropic nature (i.e. its tendency to form clusters that are circular in m dimensions), it is common practice to standardise all input variables for k-means analysis (Milligan and Cooper, 1988, Schaffer and Green, 1996, Steinley, 2004). This also applies to the SSSKM algorithm.

In practical applications of SSSKM however, it is likely that the explanatory variables contain values of a categorical nature. As discussed in Chapter 2, the numerical and continuous assumption is not unique to clustering algorithms, but is also present in regression analysis. Numerous techniques for incorporating categorical variables into regression have been developed, and can be considered for the semi-supervised segmentation algorithm as well. Each of these does, however, have an impact on the final result.

We will discuss some practical implications of using the following known techniques to convert categorical variables to numeric:

- Using standardised target rates (Anderson, 2007).
- Using weights of evidence (WoE) (Siddiqi, 2006).
- Using 'dummy' variables (Anttila-Hughes and Hsiang, 2013).

The first two methodologies have the advantage of minimising the number of required input variables to the segmentation algorithm, as each categorical variable will only have a single numerical representative. These methodologies involve binning all input variables, and assigning either average target rates to each bin, or assigning WoE values¹ to each bin. In the case of standardised target rates, the target rates of each variable are standardised by subtracting the mean and dividing by the standard deviation before being used, whilst the input variables from the WoE technique will not have standardised variances. This essentially means that the WoE technique will cause the SSSKM algorithm to place larger emphasis on strong predictors of the target during the semi-supervised segmentation analysis since stronger predictors will have larger variances.

The biggest disadvantage of using either standardised target rates or WoE is that the assumption is made that all variables have the same relationship with the target variable

¹WoE values constitute the values that are added together to calculate the value of IV, as defined in Equation 2.15. The WoE of bin j is therefore calculated as $(P_{cj}^T - P_{cj}^F) \times \log\left(\frac{P_{cj}^T}{P_{cj}^F}\right)$.

in all segments (Anttila-Hughes and Hsiang, 2013). The values assigned to each bin will be ordered according to the variable's overall relationship with the target, and the SSSKM algorithm will not be able to identify different relationships in underlying segments.

For this reason it is recommended that the technique of using 'dummy' variables for categorical inputs are used. This has the disadvantage that each category (except for one) of the variable will be an input on its own, but does not force a specific relationship with the target variable on all segments (Anderson, 2015). It is also recommended that all inputs be standardised so that equal consideration is applied to all variables.

3.7 Step 2: Segment seed initialisation

Cluster seed initialisation has been the focus of many studies (Kang and Cho, 2009, Khan and Ahmad, 2004, Pena et al., 1999, Redmond and Heneghan, 2007) for the k-means algorithm. Although the studies vary in their recommendations, it is clear that the initialisation of cluster centres has an impact on the final results. Certain techniques are able to improve the speed with which convergence is achieved, but may bias the final result (Pena et al., 1999).

Since the SSSKM algorithm adds an additional dimension to the standard k-means algorithm, it is expected that most of the techniques proposed in studies will be less effective on the algorithm since these techniques are generally focused on density approximations and does not take the dependent variable into account.

For that reason, and since the focus of this study is not on seed initialisation, the initialisation step for the SSSKM algorithm initialises segment seeds $\mathbf{u}_j = \{u_{j1}, u_{j2}, \dots, u_{jm}\}$ randomly. Two methodologies were considered for random initialisation:

- Initialisation based on variable range (range initiation) (Pena et al., 1999).
- Initialisation based on random observation selection (Khan and Ahmad, 2004, Pena et al., 1999).

The first methodology involves selecting a random coordinate for each variable between its maximum and minimum value. That is, for each initial segment centroid $\mathbf{u}_j$, a set of coordinates is chosen between $\max\{x_{i1}, x_{i2}, \dots, x_{im}\}$ and $\min\{x_{i1}, x_{i2}, \dots, x_{im}\}$. The advantage of this is that segment centres are then truly random and less dependent on the data sample, but the disadvantage is that outliers may affect the relevance of the position chosen. In such cases, segment centres may be initialised in positions where

none of the observations are closest to it, causing the segment to be empty after the first iteration.

To avoid this, it was decided to initialise segment centres by choosing K observations at random out of the data set $\mathbf{x}_i$ as the segment centroids (initial centre point of segments). This decreases the probability of empty segments and increases the possibility that segments will be initialised where observations are most concentrated.

Table 3.1 shows how the random observation selection initialisation method reduces the probability of segments being initialised without any assigned observations. This experiment was performed on the data sets used in Chapter 5 (the datasets are described in detail in Section 5.2). The results were measured on 100 segmentation iterations.

	Percentage initialised segments - range initiation	Percentage initialised segments - random observation selection
Direct marketing data set	32.83%	0.0%
Data set on protein tertiary structures	26.5%	0.0%
Credit application data set	44.33%	0.0%
Wine quality data set	22.0%	0.0%
Chess data set	0.33%	0.0%
Insurance claim prediction data set	34.5%	0.17%

TABLE 3.1: Comparison of percentage of segments initialised with no observations for 100 segmentation iterations

3.8 Step 3: Initial data set and variable preparation

Step 3 initialises the data set with the required variables needed for the semi-supervised segmentation analysis. This is the last initialisation step before the iterative *Assignment evaluation and update* process commences.

3.9 Step 4: Assignment

The *Assignment* step assigns observations to segments based on the SSSKM local objective function discussed in Section 3.2. By default, the measure on which the distortion function $\varphi(c)$ is based, is the information value (IV) formula discussed in Section 2.11.1.

However, any distortion function which measures target separation can be used. Each observation $\mathbf{x}_i$ is put through several sub-steps which is discussed here:

- Calculating Euclidian distances between the observation and all segment centres.
- Determining whether the distortion function can be calculated.
- Calculate the distortion value given assignment to each segment.
- Standardising the distances and distortion values.
- Evaluating the objective function and assignment.

3.9.1 Calculating Euclidian distances between observation and all segment centres

Similar to the standard k-means clustering algorithm, the objective function of the SSSKM algorithm includes the Euclidean distance $d(\mathbf{x}_i, \mathbf{u}_{cj})$ between the observation and each of the segment centres $\mathbf{u}_j$. This is calculated in accordance to Equation 2.9, and a result is retained for each segment. In pseudo code, the Euclidean distance for each specific combination of segment choice c at observation i is calculated as shown in Algorithm 3.1.

Algorithm 3.1 Calculate Euclidean Distance

Input: K {as number of segments}

Input: $\mathbf{x}_i$ {a single observation in the data set where $\mathbf{x}_i = \{x_{i1}, x_{i2}, \dots, x_{im}\}$ }

Input: $\mathbf{u}_j$ {the mean (centroid) of segment S_{cj} }

for $j = 1$ **to** K **do**

$d(\mathbf{x}_i, \mathbf{u}_{cj}) = 0$

for $p = 1$ **to** m **do**

$d(\mathbf{x}_i, \mathbf{u}_{cj}) = d(\mathbf{x}_i, \mathbf{u}_{cj}) + (x_{ip} - u_{jp})^2$

end for

$d(\mathbf{x}_i, \mathbf{u}_{cj}) = \sqrt{d(\mathbf{x}_i, \mathbf{u}_{cj})}$

end for

Output: An Euclidian distance $d(\mathbf{x}_i, \mathbf{u}_{cj})$ for each segment S_{cj} where $j = \{1, \dots, K\}$.

We denote a new vector $\mathbf{d} = \{d_1, d_2, \dots, d_K\}$

It can be seen from Algorithm 3.1 that the output is a vector $\mathbf{d} = \{d_1, d_2, \dots, d_K\}$ that contains the Euclidian distance $d(\mathbf{x}_i, \mathbf{u}_{cj})$ for each segment S_{cj} where $j = \{1, \dots, K\}$.

3.9.2 Determining whether the distortion function can be calculated

Some distortion functions, like the information value, is not defined for certain values. The information value is undefined if any of the segments does not contain at least one event and one non-event. For the first number of observations, this is often the case, so a step is required to test whether the distortion function is defined. If the distortion function is not defined, the segment allocation will be made based on Euclidean distance $d(\mathbf{x}_i, \mathbf{u}_{cj})$ only, similar to standard k-means clustering. This step ensures that the necessary tests are performed for the distortion function used and that the decision whether to use the distortion function is executed.

3.9.3 Calculate the distortion value given assignment to each segment

If the distortion function is defined for the current set of assignments to $S_{c1}, S_{c2}, \dots, S_{cK}$, the next step is to calculate the impact that assigning the next observation i to each of the possible segments will have on the overall distortion value $\varphi(c)$. In effect we are generating a new assignment c^* for each segment. This is shown in Algorithm 3.2.

Algorithm 3.2 Calculate distortion factor

Input: K {as number of segments}

Input: y_i {the value of the target variable for observation i }

Input: c {the current assignment of observations to segments $S_{c1}, S_{c2}, \dots, S_{cK}$ }

$i \in \{\}$ {observation i is unassigned before this step}

for $j = 1$ **to** K **do**

 Let $S_{c^*j} = S_{cj} \cup \{i\}$

 calculate $\varphi(c^*)$

end for

Output: A distortion function $\varphi(c^*)$ given assignment of i to each S_{cj} where $j = \{1, \dots, K\}$. We denote a new vector $\boldsymbol{\varphi} = \{\varphi_1, \varphi_2, \dots, \varphi_K\}$

As can be seen from Algorithm 3.2, the output is a vector $\boldsymbol{\varphi}$ of distortion values given assignment of observation i to each of the segments.

3.9.4 Transforming the distances and distortion values

From the outputs of the algorithms discussed in Sections 3.9.1 and 3.9.3, both the unsupervised (Euclidean distances) and supervised (IVs) elements are now available to calculate assignment. What remains is to ensure all elements are standardised so the

variances of both the supervised and unsupervised elements are equal. This prohibits one element from dominating the value of the objective function, as discussed below.

Because the average value and value range of the distances to each of the segment centroids d_j could differ quite substantially from the values of the distortion value φ_j , all values of φ and $\mathbf{d}$ are standardised for observation i in respect to every segment S_{cj} . The value of the distortion function is also negated since, for most distortion functions of this nature, higher values indicate better separation and from Equation 3.2 it is clear that the objective function is to be minimised. Mathematically put, for each observation i , the distance d_j and distortion factor φ_j is respectively replaced with standardised distance d'_j and negated standardised distortion factor φ'_j for every segment S_{cj} , and is calculated as

$$d'_j = \left[\frac{d_j - \bar{d}}{\sigma_d} \right], \quad \varphi'_j = \left[-\frac{\varphi_j - \bar{\varphi}}{\sigma_\varphi} \right], \quad (3.3)$$

where:

- $\bar{d}$ is the average Euclidean distance from observation i to each of the segment centroids $\mathbf{u}_{cj}$.
- $\bar{\varphi}$ is the average value of the distortion function given assignment to each of the segments S_{cj} .
- σ_d and σ_φ are the standard deviations of $\mathbf{d}$ and φ respectively.

To illustrate the importance of standardising data, consider the example of a data set on which the SSSKM algorithm is applied (using IV as distortion function) in Figure 3.2.

Figures 3.3 and 3.4 show the difference in distribution between the distances and the IVs before and after standardisation. From these, the requirement for standardisation is clear: The distributions of distances and IVs are not comparable in Figure 3.3, whilst distributions are very similar in Figure 3.4.

3.9.5 Evaluating the objective function and assignment

The next step is to assign observation i in such a way as to minimise the value of the objective function. Mathematically put, assign observation i to $S_{cj_{min}}$, where

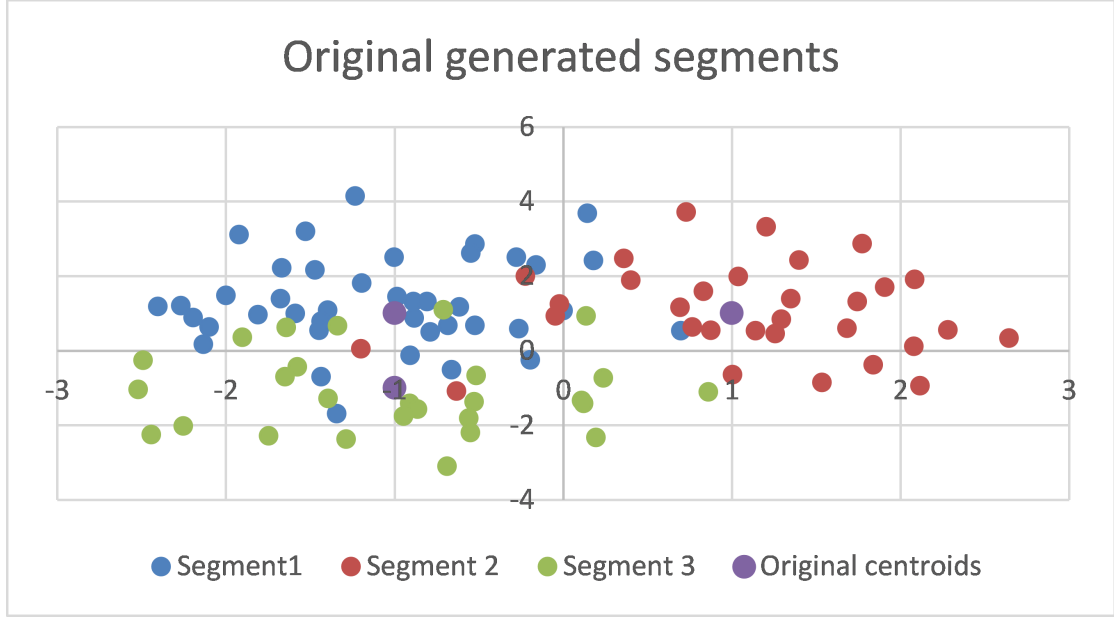


FIGURE 3.2: Example of a segmented data set

$$j_{min} = \underset{j=1,\dots,K}{\operatorname{argmin}} [w\varphi'_j + (1-w)d'_j] . \quad (3.4)$$

Equation 3.4 is our local objective function for the semi-supervised segmentation algorithm as applied to k-means (SSSKM).

3.9.6 Repeating the update step to avoid volatility

Because the distortion functions used to adjust the k-means algorithm for SSSKM are aggregate functions over a number of observations, it has the weakness that it can be volatile when calculated over only a few data points. This means that the first number of assignments done by the algorithm may be nonsensical. This is not a problem for large data sets containing many observations since the nonsensical assignments are too few to influence centroid updates, but when data sets are small it may influence the result. It could also prevent the algorithm from converging.

This weakness is unique to the SSSKM algorithm due to the addition of the distortion function. To overcome this weakness, the option exists to repeat the assignment step a number of times on the same data set after randomly re-ordering the observations. Each result is then appended to the previous and the update step is performed on a much larger data set. By doing this, the nonsensical assignments will be randomly distributed across all observations and will therefore have a negligent impact on the update step.

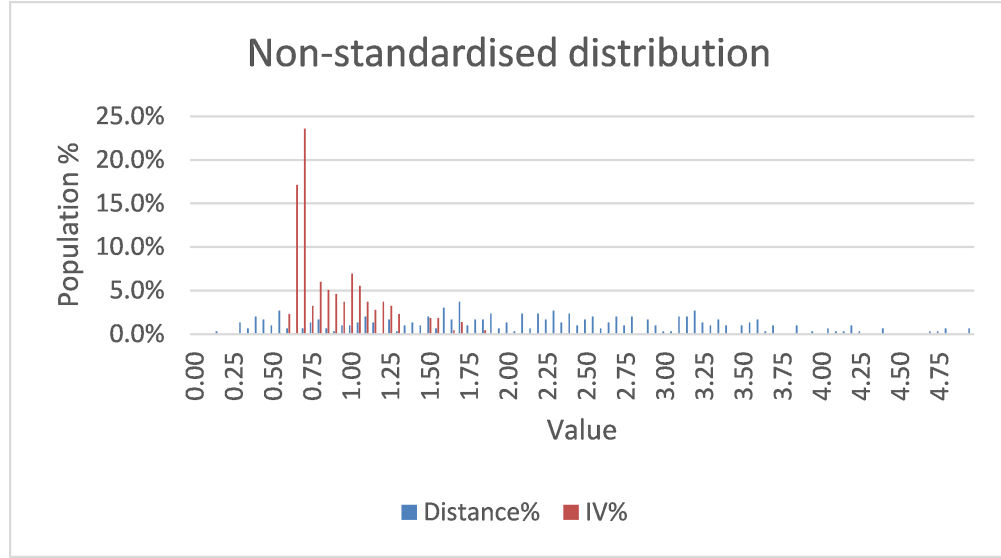


FIGURE 3.3: Distribution of distances and IVs before standardisation

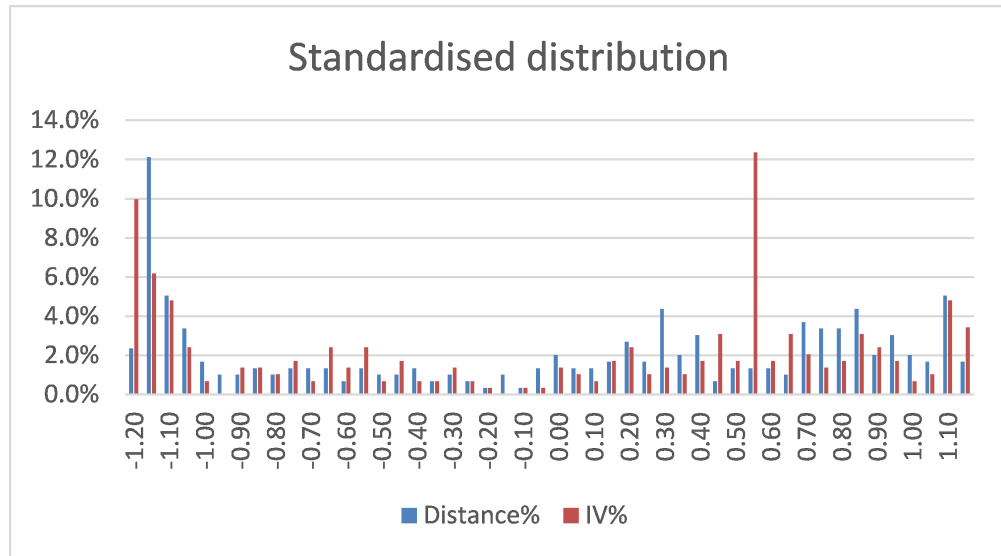


FIGURE 3.4: Distribution of distances and IVs after standardisation

Example

To illustrate the impact of the SSSKM algorithm's objective function on observation assignment, consider Figure 3.5 and Table 3.2. The example assumes a distortion weight w equal to 0.5. It further assumes that all observations in the data set have already been assigned except for the indicated observation. Table 3.2 shows the standardised values of d'_j and φ'_j if the unassigned observation is assigned to segment 1, 2 and 3 (indicated by row 1, 2 and 3).

From the example it can be seen that, even though the Euclidian distance to Segment 1 is smaller, the observation will be assigned to Segment 2 since the objective function is smallest for this segment. This simplistic example shows how the addition of the

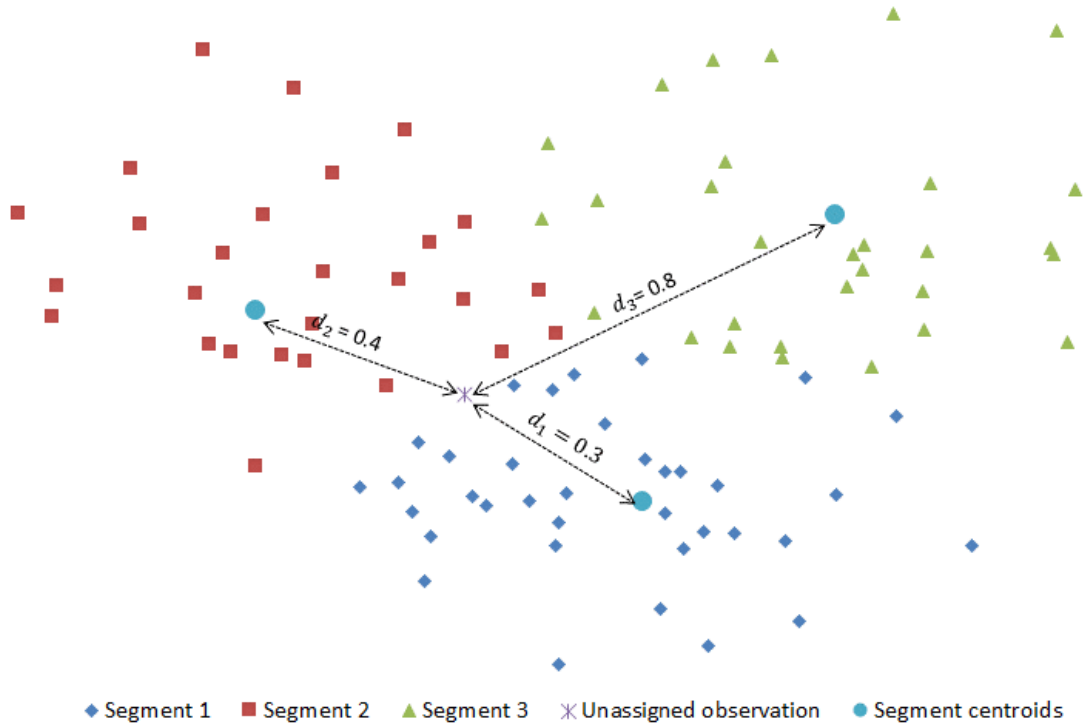


FIGURE 3.5: How the objective function impacts segment assignment

j	d'_j	φ'_j	Objective function
1	0.3	0.4	0.7
2	0.4	0.2	0.6
3	0.8	0.2	1.0

TABLE 3.2: How the objective function impacts segment assignment

distortion function influence segment assignment to optimise both the dependent and explanatory variables.

3.10 Step 5: Assignment-evaluation and update

Similar to the standard k-means algorithm, the update step of the SSSKM algorithm updates the segment centroids μ_1 to μ_k based on the new assignments made in Step 4. The option exists to use either the mean or alternatively the median to avoid the impact of outliers (in this study we have always used the mean). This step also evaluates the assignments made to ensure all segments have had observations assigned to it. Since the algorithm assumes that the user requires the indicated number of segments, the segmentation process will be randomly re-initiated if a segment has no assigned observations.

3.11 Step 6: Summarise and log step statistics

In order to assess various aspects and results of the SSSKM analysis, a number of key statistics are logged throughout the process. These are:

- The coordinates of the segment centres after each iteration.
- The distance moved by each centroid after each iteration.
- The target rate (or target average) of each segment after each iteration.
- The percentage of the observations in each segment after each iteration.
- The CH value of the segmentation after each iteration.
- The value of the distortion function $\varphi(c)$ after each iteration.
- The relative distance of each segment to the other segments.
- The number of segment assignments that was changed due to the influence of the distortion factor. That is, how many observations were assigned to a different segment due to the addition of $\varphi(c)$ to the objective function.
- The time and speed with which each iteration was performed as well as an estimated conclusion time as calculated after each iteration.

3.12 Step 7: Randomise data set

As explained in Section 3.9.6, the order in which observations are assessed could make a difference to the segment they are assigned to. In order to avoid the order in which observations are assessed biasing the final output, the observations are randomly resorted after each assignment step. This biasing effect was pointed out by Wagstaff et al. (2001) and a number of subsequent studies in supervised (Eick et al., 2004) and semi-supervised clustering (Basu et al., 2004, Bilenko, 2004) implemented similar measures to avoid it.

3.13 Step 8: Evaluate stopping criteria

In standard k-means analyses, the iterative *Assignment and update* process is stopped when no assignments change from one step to the next. This works well for k-means and can be considered complete convergence to an optima. This complete convergence is,

however, very unlikely in the case of SSSKM as applied in this study. This is once again due to the distortion function being dependent on the order in which observations are assessed. Similar behaviour was observed in other studies regarding supervised clustering (Peralta et al., 2013). For this reason the stopping criteria needed to be revisited for the SSSKM algorithm. Two options were considered and are available as options in the algorithm, one being more thorough than the other, but also taking much longer. The shorter option follows these steps:

- The standard k-means convergence criteria is assessed, and if no observations were reassigned after the previous step, the process is stopped.
- If the standard convergence criteria is not met, the average distance that the segment centroids moved from one step to the next is measured for a number of runs.
- As long as the distances that the segment centroids travel are still reducing, the process is repeated. Whenever the average distance increases from one step to the next, the process is stopped and convergence assumed.

This methodology has the weak point that the possibility exists that the process can be halted prematurely. However, as mentioned, it is much quicker than alternative methodologies and therefore allows the user to assess segments a number of times to measure consistency of results.

Another option also exists:

- The standard k-means convergence criteria is assessed, and if no observations were reassigned after the previous step, the process is stopped.
- If the standard convergence criteria is not met, the average distance that the segment centroids moved from one step to the next is measured for a number of runs.
- As long as the average distance that the segment centroids travel is still reducing, the process is repeated. Whenever the average distance increases from one step to the next the process initiates a stopping assessment stage.
- The assessment stage essentially saves the results from the last iteration where the distance increased. It then runs a number of additional iterations to assess whether a new downward trend occurs for the distances that centroids move. If a downward trend continues, the process is stopped after the first time that a smaller average movement is measured. However, if no downward trend can be found, the result that was retained is used.

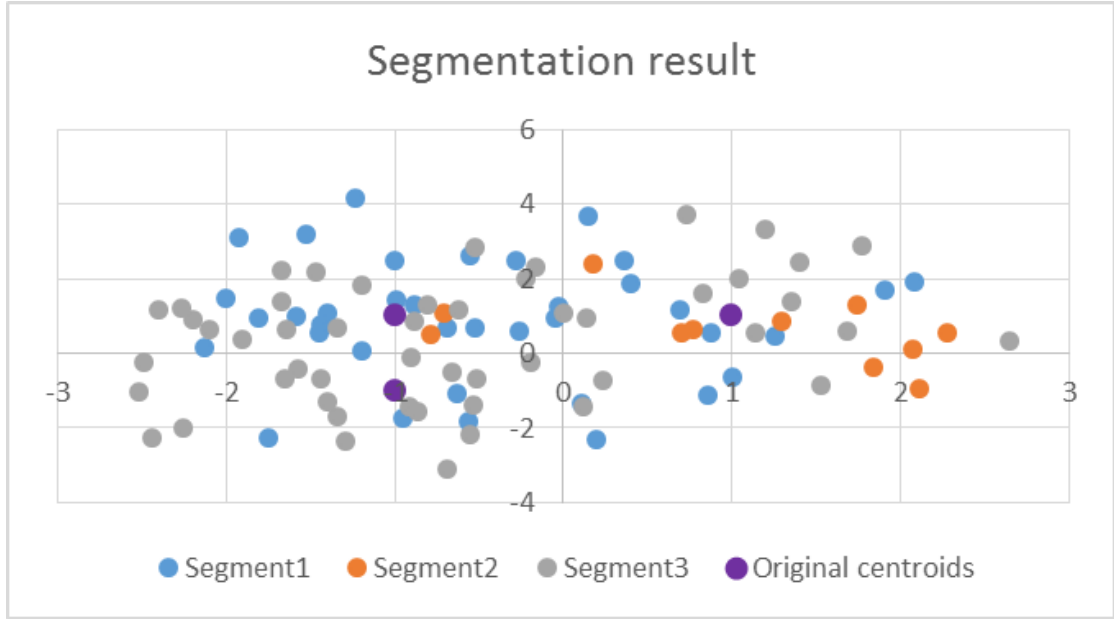


FIGURE 3.6: Example of a segmented data set

In this study we used the first option for all analysis due to the much shorter run-time.

3.14 Step 9: Overfit evaluation and smoothing

As with most supervised classification algorithms, the SSSKM algorithm can overfit when values of w becomes too large. We define overfit in this study as the phenomenon that occurs when observations are no longer logically assigned to segments based on explanatory variable proximity during a segmentation analysis, but almost entirely due to their target value. When this happens, segments cannot be applied on new data sets since there is no way to describe them through their explanatory variables. Figure 3.6 shows an example of such a set where the value for w is too high.

The effect of overfit in SSSKM is that the distortion value of the set on which the analysis is performed (i.e. training or development set) will continue to increase to very large values, however, distortion values will be low when applied to a validation set.

To avoid overfit and ensure that it can be clearly noticed when it occurs, the SSSKM algorithm makes use of k nearest neighbourhood (KNN) smoothing. The KNN smoothing methodology has been well established. See for instance Friedman et al. (2001), and we also discuss it in Section 2.4.4. In Section 2.4.4 the algorithm is discussed within the context of supervised classification, but by applying the same "nearest neighbour" methodology, a data set can be smoothed such that areas in the data set will have more uniformity in segment assignment. The methodology can also be used to make segment

assignments to new data sets, so the data set is split into a development data set and a validation data set. The SSSKM algorithm is applied to the development data set, and the smoothing algorithm is used to assign segments to the validation set.

This works in that each observation in the validation set is compared to a set number of closest (according to Euclidean distance) observations in the development data set. The observation is then assigned to the segment that is best represented by the set number of closest observations. The final statistics of the validation set can then be used to determine the success of the semi-supervised segmentation exercise.

A similar exercise can be performed on the development set as well, which will identify observations in the same segments that are far from each other and assign these outlier observations to a segment that is more suited and well generalised.

An input to this KNN algorithm is the number of observations to use for the comparison. Guidelines for choosing the best value for this input has been researched by many, and for many applications (Devroye, 1996, Györfi and Krzyzak, 2002, Hall et al., 2008). The optimal input value is, however, sensitive to the application for which it is used. In this application, if this value is too large, it is possible to lose some valuable information of the semi-supervised segmentation exercise, and if it is not large enough, overfitting will not be fully avoided. The value should be a function of both the number of observations in the data set as well as the number of segments required.

Since the aim of this study is not on finding optimal values for KNN smoothing, we allowed the choice of the input to be guided by the user. The final value is a function of the total number of observations, the number of segments and the user's input as

$$KNNVal = \frac{n \times KNN_{input}}{K}, \quad (3.5)$$

where:

- n is the number of observations in the data set.
- K is the number of segments.
- KNN_{input} is an input from the user and a value between 0 and 1 which indicates how granular the algorithm needs to smooth the data.

Figure 3.7 shows conceptually what this means for the final value of the distortion function of a data set. It can be seen that both the development set and the validation set reach peaks for certain values of w . Note that both the validation and development

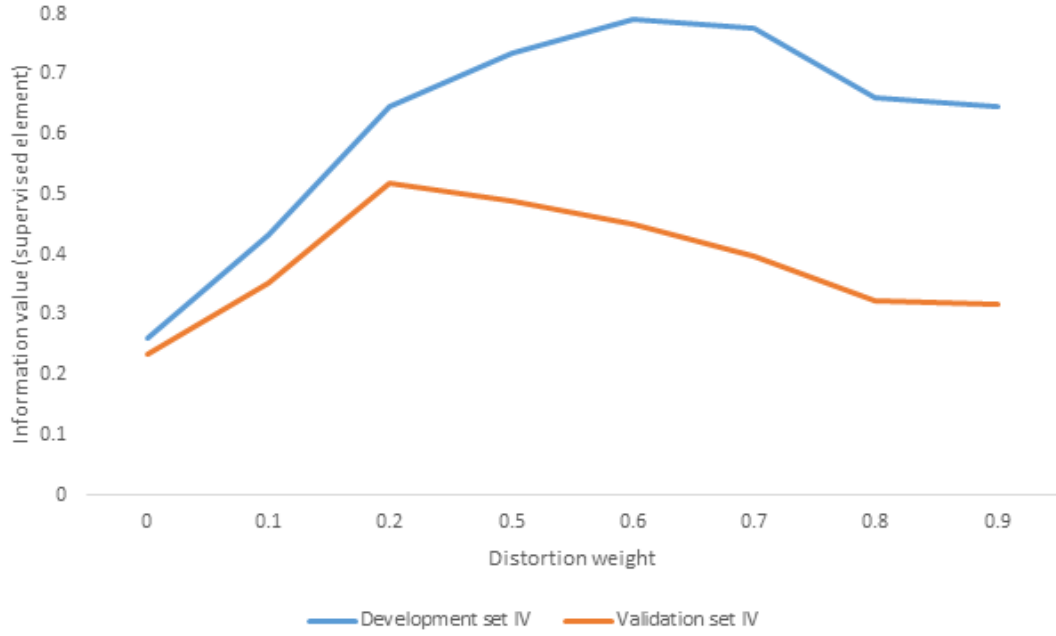


FIGURE 3.7: Example of a segmented data set

sets have been smoothed using the k nearest neighbour smoothing algorithm. The optimal value of w in this case is 0.2 to avoid all overfit. In practice, however, some overfit could be tolerated since the actual goal of the segmentation is an improvement in model accuracy and not an optimal overall distortion value. The best value for w will therefore be driven by this.

Another option for avoiding overfit on the development set that was also explored is the use of the so-called "voting" methodology to remove outlier assignments. This is done through a similar exercise as was explained in Section 3.9.6 (specifically the principle of repeating the update step more than once). The update step is performed a number of times for the data set and each observation will therefore have a number of assignments done. Each of these assignments are then seen as a "vote" for a certain segment and the segment with the most votes will be the final one to which the observation is assigned. This methodology has the advantage of not requiring input like KNN_{input} from Equation 3.5, but has the disadvantage that it cannot be used to make assignments to a validation set. For this reason, the voting option is not further explored in this study.

3.15 Step 10: Final evaluation and result logging

After the nine steps described above have been completed, a development and a validation data set are available. These have also been constructed to avoid overfitting. The

last step in the algorithm compiles some further statistics on these data sets so that the result can be compared to other runs. The statistics that are calculated are:

- The new segment centroids post the smoothing exercise for both the validation and development set.
- The target rate for each segment.
- The final population percentage in each segment.
- The CH value to describe the quality of the segmentation from an explanatory variable perspective.
- The IV value to describe the quality of the segmentation from a target variable perspective.
- The overall distortion value.
- The final distances between segment centroids.
- The impact of the smoothing exercise which is expressed as a percentage of the validation set's observations that remained the same.

The data set can then be used for development of statistical models and the impact measured on the validation data set.

3.16 Conclusion

In this chapter we introduced and defined the semi-supervised segmentation algorithm, as applied to k-means. The ten steps that form the framework of the algorithm's operation were discussed in detail. In summary, the following was achieved:

- We defined the global objective function of the SSSKM algorithm.
- We described the operation of the algorithm in ten steps.
- We defined the local objective function for the SSSKM algorithm that is performed for each observation.
- We introduced the concept of overfit for SSSKM and how overfit is dealt with in this algorithm.

In order to ensure concise referencing of techniques in the following chapters, we now provide names for more specific applications of the SSSKM algorithm. When the information value, or IV, is used as the distortion factor, we call the algorithm SSSKMIV (semi-supervised segmentation as applied to k-means using information value). When Chi-squared is used as the distortion function, we call the algorithm SSSKMCSQ (semi-supervised segmentation as applied to k-means using chi-squared).

The next chapter visually illustrates the effect that the addition of a "supervised" element to the objective function has, both on the movement and final position of segment centroids (means) as well as the shape and position of final segments. It also shows how the altered objective function allows the SSSKM algorithm to overcome some of the known weaknesses of traditional k-means clustering. In Chapter 5, we compare the SSSKM algorithm to popular current segmentation techniques on six data sets from various disciplines in the industry as well as popular non-linear modelling methodologies and in Chapter 6 we conduct similar comparisons on simulated data sets.

Chapter 4

Impact of SSSKM on traditional k-means segment assignment and segment centroid movement

4.1 Introduction

Chapter 3 introduced the semi-supervised segmentation algorithm, as applied to k-means (SSSKM). The adjustments to the standard k-means objective function are discussed in detail as well as the way in which the algorithm was implemented. In this chapter we do the following:

- Generate simple, two dimensional example data sets to enable us to explore and illustrate the properties of the SSSKM algorithm.
- Visually illustrate the effect of the addition of a "supervised" element to the objective function, both on the movement and final position of segment centroids (means) as well as the shape and position of final segments.
- Use accuracy measures, obtained from supervised clustering, to show how the addition of the target variable to the objective function enhances the algorithm's ability to identify underlying segments in the data and to overcome some of the known weaknesses of traditional k-means clustering.

In this chapter our focus is not yet on the final goal of the SSSKM algorithm, which is segmentation for predictive modelling, but solely on the impact that the addition of a "supervised" element has on its ability to find underlying segments within a data

set. For this reason, we use simple data sets with characteristics designed specifically to illustrate these effects. IV serves as our distortion function, therefore using the SSSKMIV algorithm.

4.1.1 Research contributions

In this chapter we will contribute the following:

- We visually illustrate the effect that the addition of a "supervised" element to the objective function has, both on the movement and final position of segment centroids (means) as well as the shape and position of final segments.
- We employ accuracy measures, obtained from supervised clustering, to show how the addition of the target variable to the objective function enhances the algorithm's ability to identify underlying segments in the data and overcome some of the known weaknesses of traditional k-means clustering.

The next section elaborates on the data selected.

4.2 Data used

For the purposes of this section, we will generate data with specific, known characteristics. We will use only two explanatory (or independent) variables (i.e. $m = 2$), which will enable us to illustrate the impact of the supervised element visually. The target variable will be binary in all cases, and target variable values for segments will be uniform, i.e. a segment will either have a target rate of 100%, or 0%. These are extreme examples to enable us to clearly demonstrate the impact of the "supervised" element. The two explanatory variables will be named $\mathbf{X}_1$ and $\mathbf{X}_2$, whilst the dependent, or target variable will be named $\mathbf{y}$. The three data sets we will use will be:

- Overlapping normal distributions with equal standard deviation and different means.
- Adjoining uniform distributions with varying sizes.
- "Two concentric circles" - two uniformly distributed rings with similar means, but different radiuses.

The first two data sets that will be generated will have six segments. The choice of six segments for the first two data sets is purely synthetic since it presents well visually

and allows for a number of different variations in segment characteristics (especially for the second data set, where segment dimensions are varied). Similar results can certainly be obtained using fewer, or more segments, but this may either limit the number of variations that can be considered or will present a greater challenge to illustrate succinctly in graphs and tables. The third data set illustrates the globular propensity of k-means clusters and does not require six segments. In this case, six segments will be excessive, as two segments will serve to prove the point sufficiently. Similar examples using two segments are also seen in other studies (Jain, 2010).

We will define each data set in more detail in the next sections.

4.2.1 Data set 1: Overlapping normal distributions with equal standard deviation and different means

For this data set, each characteristic ($\mathbf{X}_1$ and $\mathbf{X}_2$) will be drawn from a normal distribution. For each segment, $\mathbf{X}_1$ and $\mathbf{X}_2$ will be drawn from a distribution with the same standard deviation (STD), but their means will vary in such a way that the segments overlap a lot, making it almost impossible to identify one segment from another spatially. By using a normal distribution, we ensure that the overlapping regions of segments have lower densities than the regions closer to the centroids (see Figure 4.2), which at least gives the algorithms a chance to find the correct centroids. The clearest distinguishing factor between these segments will be their target rate. The characteristics of the segments are summarised in Table 4.1

Segment	Target rate	Mean $\mathbf{X}_1$	Mean $\mathbf{X}_2$	STD $\mathbf{X}_1$	STD $\mathbf{X}_2$	Nr. Observations
1	100%	-0.5	-1	0.6	0.6	2500
2	0%	-0.5	0	0.6	0.6	2500
3	100%	-0.5	1	0.6	0.6	2500
4	0%	0.5	-1	0.6	0.6	2500
5	100%	0.5	0	0.6	0.6	2500
6	0%	0.5	1	0.6	0.6	2500

TABLE 4.1: Characteristics of the first data set

Figure 4.1 shows a scatter plot of the data set as it will be perceived by the algorithm, with no indication of segment assignment. It is clear from the graph that there are virtually no clear segments (or clusters) from a spatial perspective. Figure 4.2 shows each of the segments that was generated in a different colour. It is clear to see that there are large overlaps between the segments. For this reason one would not expect any algorithm to be able to find rules to separate the segments with high accuracy.

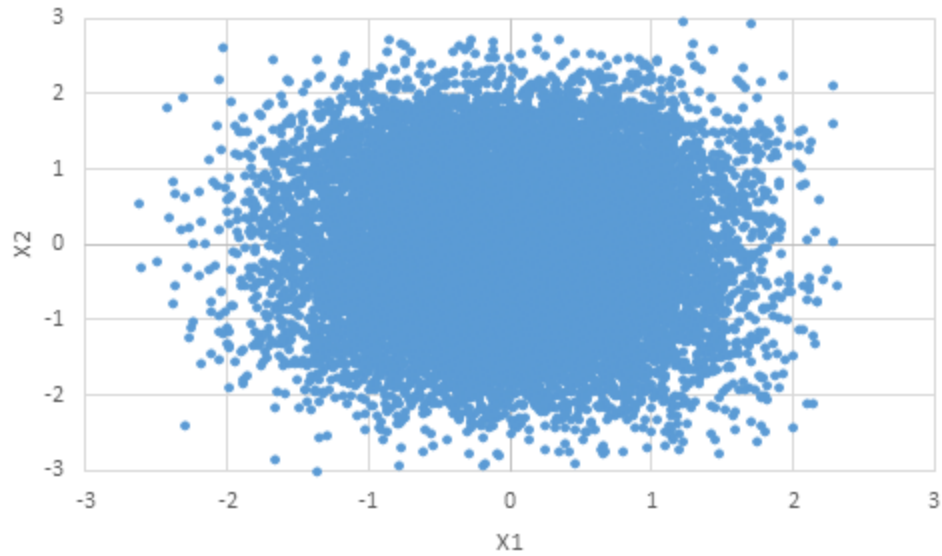


FIGURE 4.1: First data set shown with no indication of generated segments

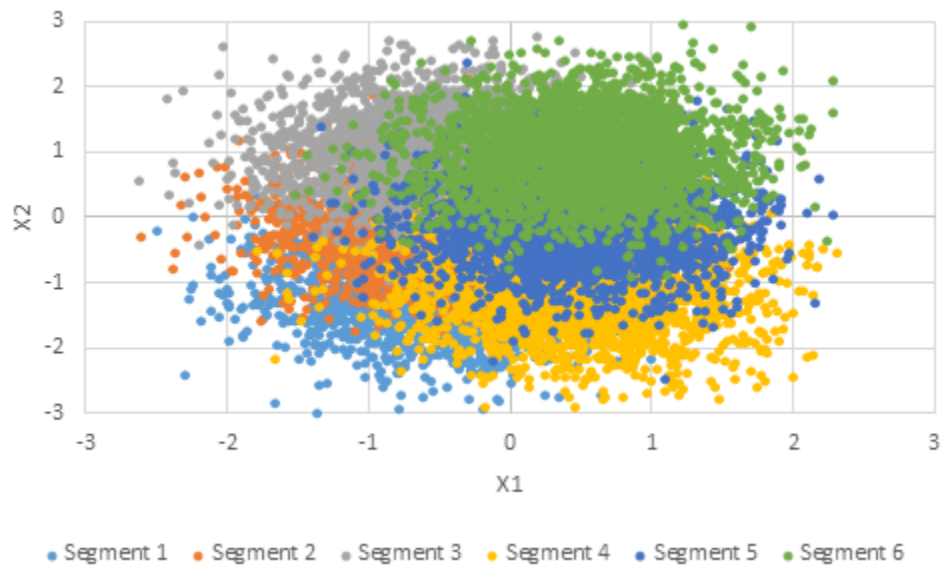


FIGURE 4.2: First data set shown with generated segments indicated

4.2.2 Data set 2: Adjoining uniform distributions with varying ranges

A well known characteristic of the k-means algorithm is that it will have a tendency to form clusters of similar sizes (Yan, 2005). When used for segmentation, this may in some cases be regarded as a positive characteristic (see Section 7.2), but when the actual underlying clusters differ in size, this may cause the k-means algorithm to identify clusters incorrectly, and can therefore be regarded as a weakness.

The second data set to be explored will be generated to analyse this tendency for the

semi-supervised algorithm. Similar to the first data set, the second data set will also consist of six segments. In contrast to the first set, however, the characteristics in these segments will not be drawn from normal distributions, but instead from uniform distributions. Because the characteristics will be drawn from uniform distributions with different ranges, it is possible to ensure that none of the segments overlap - which means that one can expect a segmentation algorithm to have greater success in identifying segments clearly. The segment characteristics are, however, drawn from distributions that are adjacent and unequally sized which means that they will not be easily identified by a standard k-means algorithm. Table 4.2 summarises the characteristics of the second data set.

Segment	Target rate	Lower range boundary for $\mathbf{X}_1$	Higher range boundary for $\mathbf{X}_1$	Lower range boundary for $\mathbf{X}_2$	Higher range boundary for $\mathbf{X}_2$	Nr. Observations
1	100%	-0.5	0	-1.5	-0.5	2500
2	0%	-0.5	0	-0.5	0	1250
3	100%	-0.5	0	0	1.5	3750
4	0%	0	0.5	-1.5	-0.5	2500
5	100%	0	0.5	-0.5	0	1250
6	0%	0	0.5	0	1.5	3750

TABLE 4.2: Characteristics of the second data set

Figure 4.3 shows a scatter plot of the data set as it will be perceived by the algorithm, with no indication of segment assignment. Once again segments are virtually impossible to distinguish from a spatial perspective, with cluster densities being similar. Figure 4.4 shows each of the segments that was generated in a different colour. Here it can be seen that, in contrast with the first data set, the generated segments do not overlap, making it easier for segmentation algorithms to assign observations to specific segments.

4.2.3 Data set 3: Two concentric circles

Another one of the well-known weaknesses of k-means clustering is its requirement for clusters to be approximately spherical, or globular in m dimensions (Riesen and Bunke, 2008, Wu, 2008). If clusters are not globular, k-means tends to find it difficult to identify clusters, even if they are spatially easy to identify (Wu, 2008). For this reason we will explore the ability of the semi-supervised segmentation algorithm to overcome this weakness by measuring its success in identifying segments of a non-globular nature under the right circumstances. The third data set will consist of two concentric circles (i.e. only two segments), both centred at 0, but with different radiuses and ranges. In this case, six segments are not used, as two segments are enough to illustrate the point

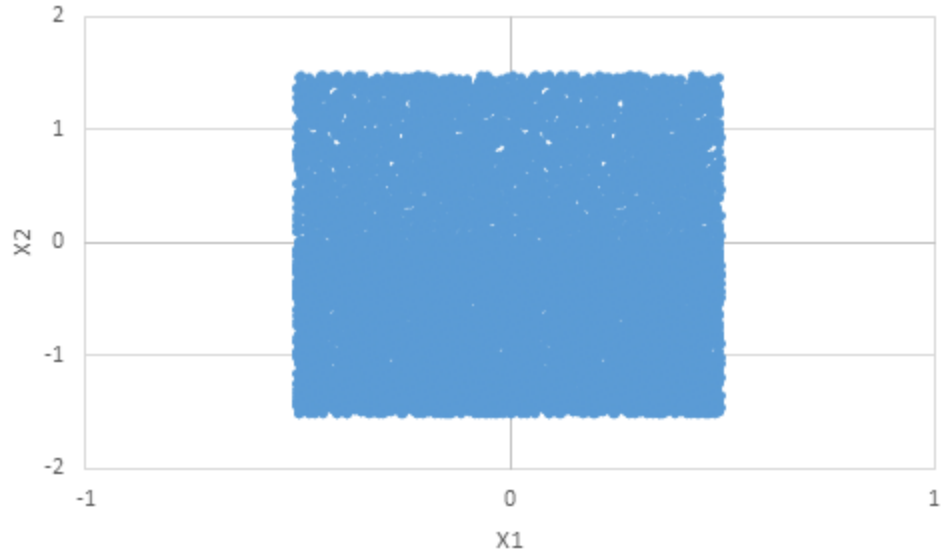


FIGURE 4.3: Second data set shown with no indication of generated segments

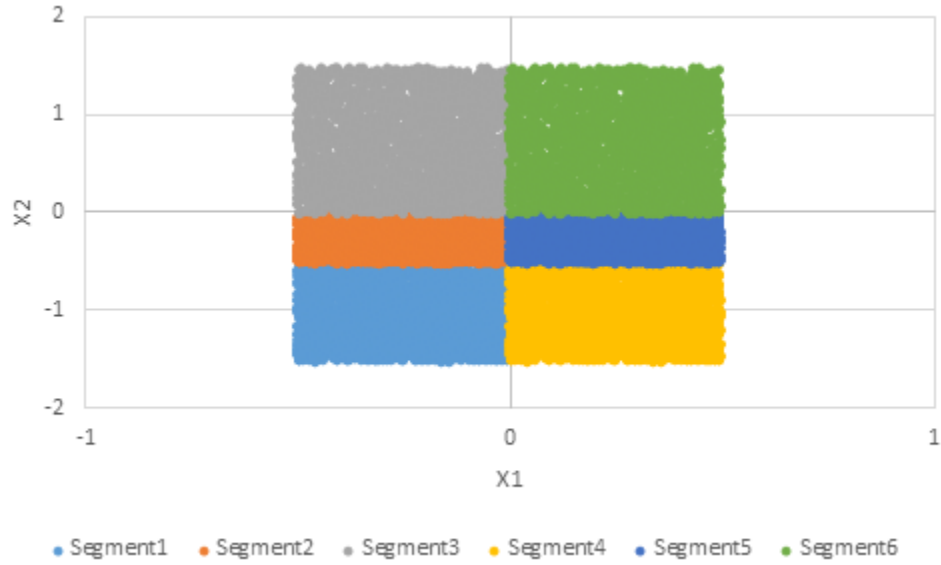


FIGURE 4.4: Second data set shown with generated segments indicated

clearly. A similar data set was used in other studies to illustrate the same point (Jain, 2010).

Mathematically put, Segment 1 will be drawn from a uniform distribution with range for all values of $\mathbf{X}_p$ (in this case $\mathbf{X}_1$ and $\mathbf{X}_2$) between -1 and 1. However, it will be subject to the constraints

$$0.6 < \|\mathbf{x}_i - \mathbf{u}_1\|^2 < 0.8, \quad (4.1)$$

with $\mathbf{u}_1 = (0, 0)$ in this case and $\|\mathbf{g}\|^2$ is again used to represent the length of vector $\mathbf{g}$ in Euclidean distance.

In a similar way, Segment 2 will be drawn from a uniform distribution with range between -0.5 and 0.5, subject to the constraints

$$0.2 < \|\mathbf{x}_i - \mathbf{u}_2\|^2 < 0.3, \quad (4.2)$$

with $\mathbf{u}_2 = (0, 0)$ as well.

Table 4.3 summarises the characteristics of the third data set.

Segment	Target rate	Mean $\mathbf{X}_1$	Mean $\mathbf{X}_2$	Min distance from mean	Max distance from mean	Nr. Observations
1	100%	0	0	0.6	0.8	2500
2	0%	0	0	0.2	0.3	2500

TABLE 4.3: Characteristics of the third data set

Figure 4.5 shows a scatter plot of the data set as it will be perceived by the algorithm, with no indication of segment assignment. The segments can be clearly identified spatially, but are not globular. Figure 4.6 shows the two segments in colour.

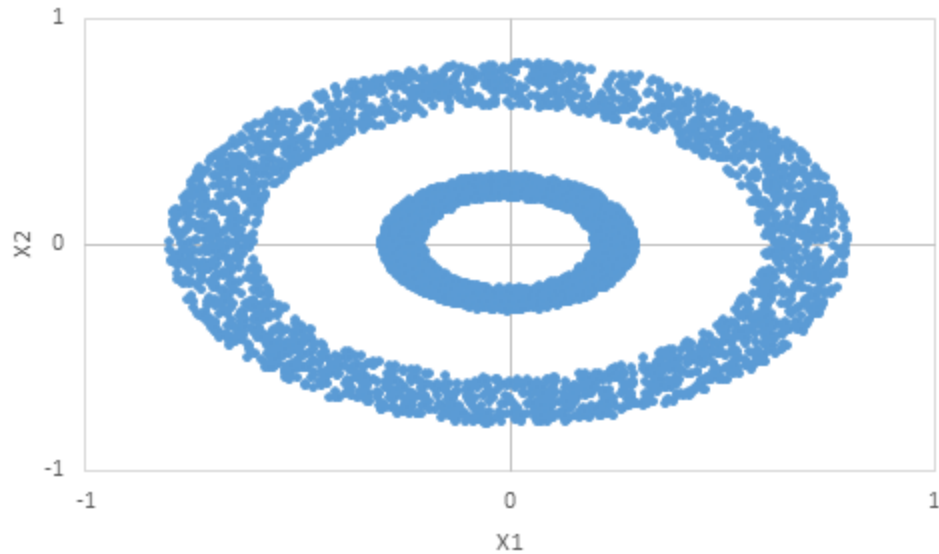


FIGURE 4.5: Third data set shown with no indication of generated segments

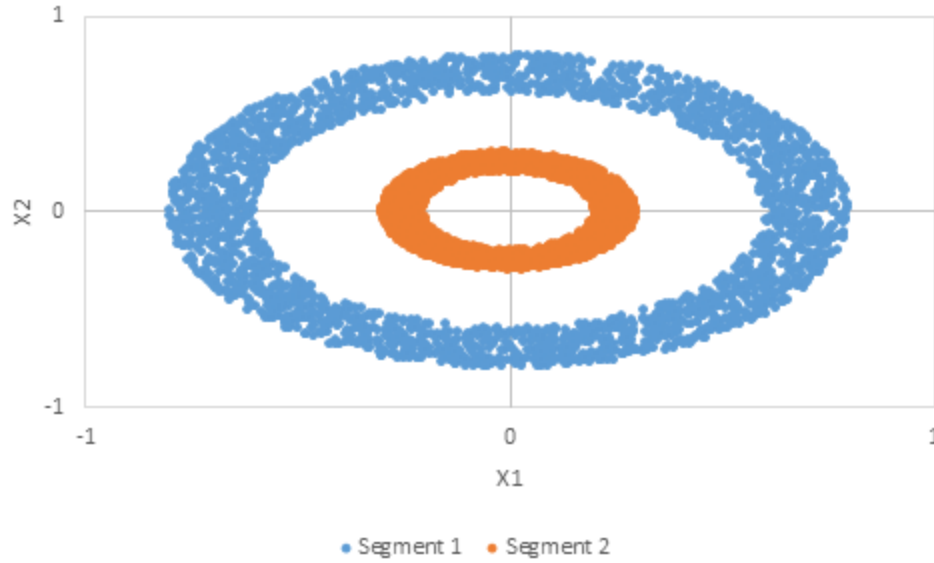


FIGURE 4.6: Third data set shown with generated segments indicated

4.3 Measuring cluster accuracy

Along with the recent interest in supervised clustering, the need arose for algorithms that can measure the success with which a labeled data set's segments have been discovered. Traditional accuracy measures for supervised techniques, like r-squared cannot be used directly since the cluster numbers that are identified will not necessarily coincide with the label numbers, even though the result may be completely correct. Meilă (2005, 2007) compare a number of popular measures, but show that there is not a single measure that can be used without shortcomings. Drawing from some recent studies (Peralta et al., 2013, Youness and Saporta, 2010), we will use two measures to indicate the success with which segments were identified:

- Projecting partitions using supervised learning (or projected partition accuracy - PPA) (Youness and Saporta, 2010).
- The adjusted Rand index (ARI) (Hubert and Arabie, 1985).

In order to distinguish between the known segments and the segments proposed by the algorithm, we will denote the known segments as $S_1, S_2, \dots, S_K$ with index j , and the segment assignments from the segmentation or clustering algorithm as $S_{c1}, S_{c2}, \dots, S_{cK}$ with index l .

4.3.1 Projecting partitions using supervised learning

The first measure use a supervised learning algorithm to project the output of the segmentation exercise $S_{c1}, S_{c2}, \dots, S_{cK}$ onto the known original segments $S_1, S_2, \dots, S_K$. Since the input variables to the supervised learning algorithm is simply the output of the clustering, or segmentation algorithm (i.e. values $S_{c1}, S_{c2}, \dots, S_{cK}$) and since we assume here that we will ensure the algorithm produce only K segments, the methodology that will be used for projecting the segments will be a simplistic optimisation algorithm. The algorithm iteratively projects an output segment S_{cl} onto an original segment S_j that has the largest intersect with S_{cl} (i.e. has the most mutual observations). This projection is then completed and the algorithm moves on to the next segment, until all segments have been projected. Once the projection is done for all segments, the success of the clustering (or segmentation) exercise can be measured by the percentage of correctly projected segments i.e. projected partition accuracy (PPA). The PPA is therefore a value between 0% and 100%, and indicates the percentage of projected observations that has the correct segment assignment.

Algorithm 4.1 provides a summary of the iterative process.

Algorithm 4.1 Project segments

```

Obtain intersect of all output segments with original segments
 $\{S_1 \cap S_{c1}, S_1 \cap S_{c2} \dots S_K \cap S_{cK}\}$ 
Sort intersects from largest overlap to smallest
repeat
  Project largest intersect output segment  $S_{cl}$  onto relevant original segment  $S_j$  so
  that  $j, l = \operatorname{argmax}(S_j \cap S_{cl})$ 
  Remove all remaining projections onto  $S_j$ 
  Move to next projection
until all projections are complete
  
```

4.3.2 The adjusted Rand index

The adjusted Rand index (ARI) has been proposed by Hubert and Arabie (1985) as a normalised measure for cluster similarity as derived from the Rand index (Rand, 1971). This measure has been extensively used to measure supervised clustering success in recent studies (Meilă, 2005, 2007, Peralta et al., 2013, Youness and Saporta, 2010). The ARI expresses the calculated Rand index as a percentage of the theoretical "perfect" Rand index value, and therefore assumes values between 0% and 100%, with a value of 100% meaning that a perfect likeness was obtained. We assume again that the algorithm will be forced to produce only the number of known segments. We denote n_j as the

number of observations in original segment S_j , and n_{cl} as the number of observations in the algorithm's proposed segment, S_{cl} . We use $S_j \cap S_{cl}$ to represent the number of observations that are assigned to both S_j and S_{cl} (i.e. the *intersect* of S_j and S_{cl}) and we use $(S_j \cap S_{cl})^C$ to represent the number of observations that are not assigned to both S_j and S_{cl} (i.e. the *compliment* of the intersect of S_j and S_{cl}). If n is the number of observations in the full data set (as defined in Section 2.4), then the Rand index (Rand, 1971) is calculated by

$$\frac{\sum_{j,l} \binom{S_j \cap S_{cl}}{2} + \sum_{j,l} \binom{(S_j \cap S_{cl})^C}{2}}{\binom{n}{2}}. \quad (4.3)$$

The adjusted Rand index (ARI) corrects the standard Rand index for chance (see (Hubert and Arabie, 1985) and (Vinh et al., 2009)), and is calculated as (Yeung and Ruzzo, 2000)

$$\frac{\sum_{j,l} \binom{S_j \cap S_{cl}}{2} - \left[\sum_{j=1}^K \binom{n_j}{2} \sum_{l=1}^K \binom{n_{cl}}{2} \right] / \binom{n}{2}}{\frac{1}{2} \left[\sum_{j=1}^K \binom{n_j}{2} + \sum_{l=1}^K \binom{n_{cl}}{2} \right] - \left[\sum_{j=1}^K \binom{n_j}{2} \sum_{l=1}^K \binom{n_{cl}}{2} \right] / \binom{n}{2}}. \quad (4.4)$$

Note that the equations were not copied verbatim from the references used (i.e. (Yeung and Ruzzo, 2000) and (Rand, 1971)), but that the notation was adjusted to be consistent with the notation used throughout this thesis. The principles were, however, obtained from the quoted references, and hence the reason for the citations.

4.4 Results

The results of the segmentation analysis on each of the three data sets will be analysed from two points of view. Firstly the two accuracy measures provided above (PPA and ARI) will be used to provide an overview of the accuracy with which assignments have been made to segments. We will then also examine how the addition of the supervised element to the objective function influence the movement of the centroids of the data sets being analysed. This will be explained further along with the results.

In order to obtain a thorough view of the success of the semi-supervised algorithm, each data set will be analysed a number of times with a number of different values for w . As mentioned before, the k-means algorithm is prone to convert to local optimum points. For this reason, each weight is assessed 20 times to provide a large enough result sample to ensure these local optimum points do not provide a skewed view on the results. The distortion weight, w , will be increased from 0 (standard k-means clustering) to 0.9

by increments of 0.1 to show how the change in the weight impacts the final segment assignment. Each data set is therefore analysed at least 200 times. Since we will also explore how centroid movements are affected, we will force the algorithm to initiate the centroids at a fixed point each time - this will ensure that we compare the centroid movements from the same starting points for all values of w . Table 4.4 provides the initial values of $\mathbf{X}_1$ and $\mathbf{X}_2$ for each of the segments. The initial values were chosen to be well distributed throughout the data set, and also to ensure the centroid movements can be clearly distinguished in Figures 4.9 and 4.10. Also note that accuracy measures will be calculated on the validation set as described in Section 3.14.

Segment	Initial value of $\mathbf{X}_1$	Initial value of $\mathbf{X}_2$
1	-0.25	0
2	-0.05	0
3	-0.15	0
4	-0.35	0
5	-0.55	0
6	-0.75	0

TABLE 4.4: Initial values of $\mathbf{X}_1$ and $\mathbf{X}_2$ for each of the segments

Note that data set 3 below only has two segments, which means the values of segments three to six do not apply.

4.4.1 Results data set 1: Overlapping normal distributions with equal standard deviation and different means

Table 4.5 shows a summary of the accuracy with which segments were identified for the first data set. It is clear from the table that the addition of a supervised element improves performance for values of $w = 0.1$ to $w = 0.8$. What can also be seen is the impact of overfit (discussed in Section 3.14) from $w = 0.6$, where accuracy starts to decrease again. Note here that the target variable ($\mathbf{y}$) is used in the "supervised" part of the semi-supervised segmentation. This is another example of how our semi-supervised segmentation algorithm differs from supervised clustering - supervised clustering typically uses the originally generated segments as its target (Peralta et al., 2013, Ye et al., 2008).

Table 4.6 shows the segment-specific accuracy using the PPA measure. What is clear to see is that Segments 2 and 5 mostly have the lowest accuracy. This is because they have segments overlapping on both sides. We know from Table 4.5 that the overall accuracy is best for values of w between 0.1 and 0.5, but we can see that Segments 1 and 4 actually decreases in accuracy when $w > 0$. Investigation revealed that Segments

Distortion weight (w)	Average PPA	Average ARI	Nr. Iterations
0 (standard k-means)	52.6%	24.9%	20
0.1	58.42%	27.7%	20
0.2	58.44%	27.7%	20
0.3	58.25%	27.6%	20
0.4	58.31%	27.6%	20
0.5	58.34%	27.7%	20
0.6	57.22%	27.0%	20
0.7	55.21%	26.2%	20
0.8	52.62%	24.9%	20
0.9	49.27%	23.3%	20

TABLE 4.5: Results for the first data set

1 and 4 are chosen to be large enough by unsupervised k-means to include most of its own observations, but it also included many observations from Segments 2 and 5 (and hence their low accuracy). In order to improve overall segment accuracy (and specifically Segments 2 and 5), the range of Segments 1 and 4 are reduced by the semi-supervised segmentation algorithm, which means including slightly less of their own observations, but improving accuracy for Segments 2 and 5.

Distortion weight (w)	Segment 1	Segment 2	Segment 3	Segment 4	Segment 5	Segment 6
0 (standard k-means)	65.5%	40.7%	54.1%	60.2%	43.3%	51.5%
0.1	62.4%	53.9%	60.9%	55.4%	56.6%	61.1%
0.2	61.5%	54.6%	61.3%	57.7%	56.9%	59.4%
0.3	62.5%	54.8%	60.4%	55.6%	56.9%	59.2%
0.4	60.9%	55.6%	60.7%	56.1%	57.3%	59.2%
0.5	59.8%	55.6%	62.1%	56.8%	56.9%	58.7%
0.6	60.5%	50.9%	62.3%	57.6%	53.9%	58.7%
0.7	56.8%	46.3%	63.7%	60.2%	43.8%	60.5%
0.8	50.8%	39.3%	63.8%	62.5%	42.4%	56.9%
0.9	51.9%	33.5%	56.7%	56.8%	32.1%	62.9%

TABLE 4.6: Segment-specific accuracy results using PPA

Figures 4.7 and 4.8 show examples of results of unsupervised and semi-supervised segmentation analysis on the first data set respectively. The weight used in this example is $w = 0.2$. The reason for the improved results for the semi-supervised segmentation is clear to see from these figures (refer back to Figure 4.2). It can be seen that the unsupervised results opts for a segment in the middle of the data set. However, because of the added information, the semi-supervised segmentation algorithm is able to propose a far more representative segmentation framework.

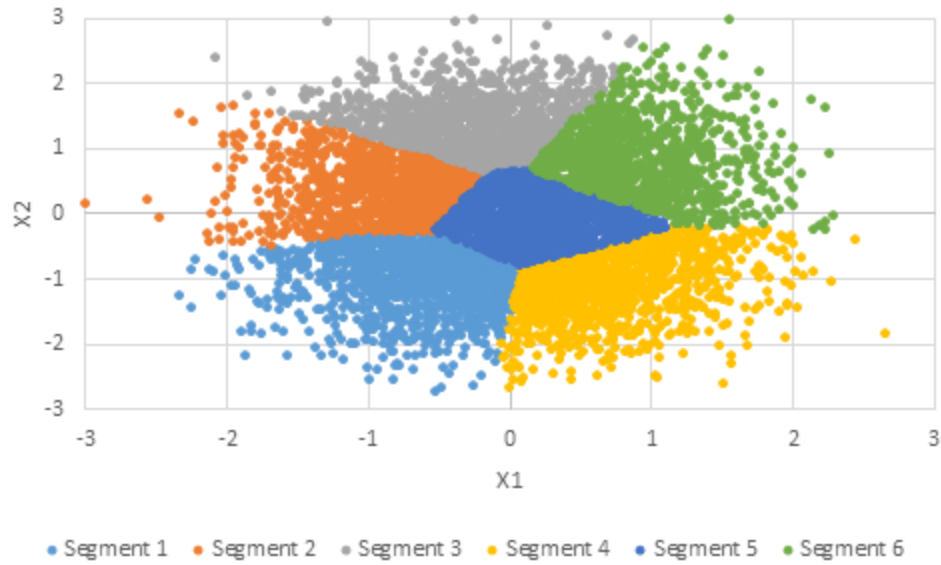
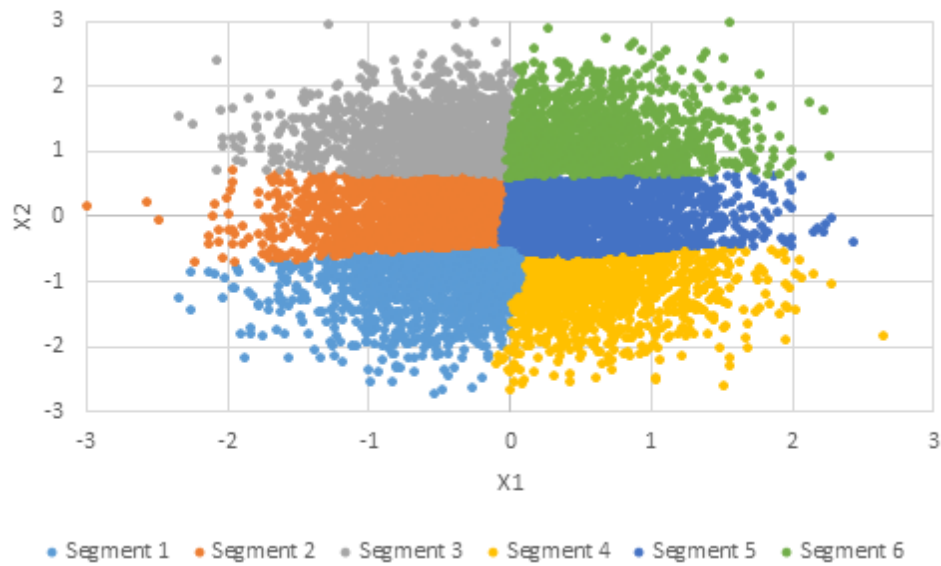


FIGURE 4.7: Unsupervised segmentation result for the first data set

FIGURE 4.8: semi-supervised segmentation result for the first data set ($w = 0.2$)

We next examine how the movement of the segment centroids are affected by the addition of the supervised element to the objective function. Since the k-means clustering algorithm works by moving the centroids of all clusters in response to cluster assignment, the positions of the centroids can be seen as a very informative characteristic in any methodology based on k-means. We will therefore show how the movement of these centroids are affected by the altered objective function.

We record the position of each centroid after a selection of iterations, and plot this progression on a scatter plot. In order to visually show the progression, we gradually

decrease the size of the marker of each new centroid position. The final position is indicated by a black dot for each centroid. We also indicate the actual segment centroids with red dots. From Figures 4.9 and 4.10 it can be seen that the endpoints of the semi-supervised segmentation analysis results in centroids that are much closer to the actual centroids, compared to unsupervised segmentation.

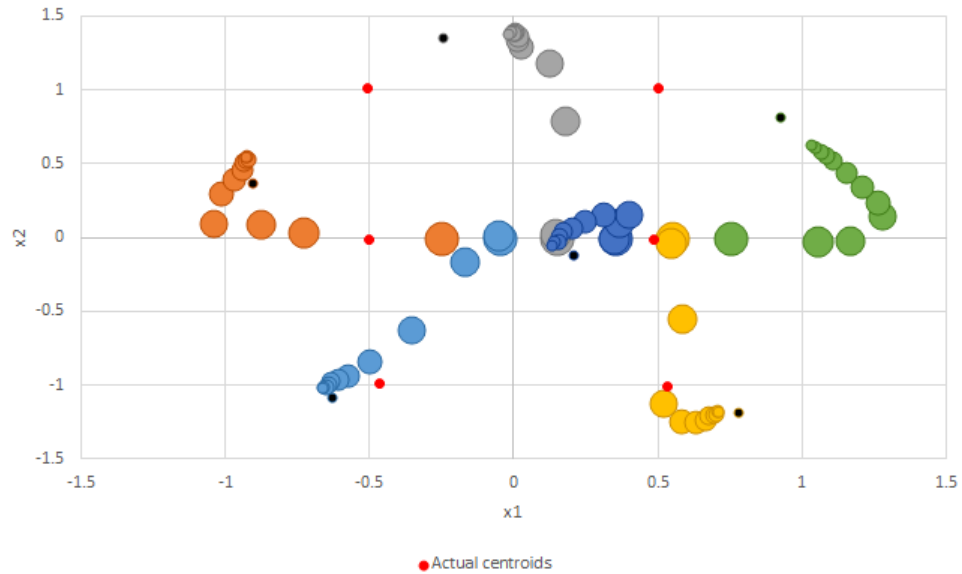


FIGURE 4.9: Unsupervised centroid movements for the first data set

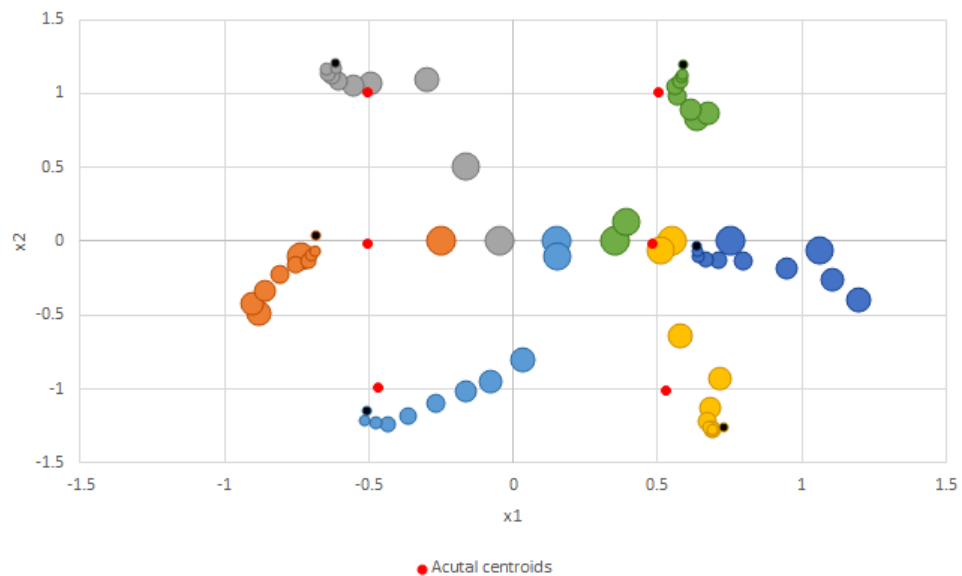


FIGURE 4.10: Semi-supervised segmentation centroid movements for the first data set
($w = 0.2$)

4.4.2 Results data set 2: Adjoining uniform distributions with varying ranges

Table 4.7 shows a summary of the accuracy with which segments were identified for the second data set. Once again, a clear improvement over standard k-means clustering can be seen for values of $w \geq 0.3$. Of note in these results are that, as expected, the accuracy measures used are generally higher, indicating more successful segment assignment. This is because, in contrast to the first data set, there is no overlap between segments. Overfit is less pronounced in this instance, and the best results are obtained for high values of w . This shows that the optimal value for w will differ depending on the characteristics of the data set analysed.

Distortion weight (w)	Average PPA	Average ARI	Nr. iterations
0 (Standard k-means clustering)	58.99%	43.99%	20
0.1	58.09%	47.45%	20
0.2	56.18%	47.26%	20
0.3	59.35%	48.27%	20
0.4	59.53%	50.67%	20
0.5	61.21%	51.96%	20
0.6	65.42%	55.42%	20
0.7	70.4%	61.92%	20
0.8	76.46%	67.96%	20
0.9	77.38%	69.89%	20

TABLE 4.7: Results for the second data set

Table 4.8 shows the segment-specific accuracy using PPA. Segments 1, 3, 4 and 6 benefits most from semi-supervised segmentation, whilst there are small decreases in accuracy for Segments 2 and 5.

Figures 4.11 and 4.12 show examples of results from unsupervised and semi-supervised segmentation analysis on the second data set respectively. The weight used for this example is $w = 0.9$. Because the unsupervised k-means algorithm does not use target rate information, it identifies segments that run across the entire width of the data set. The algorithm also tends to prefer segments of similar sizes, which means Segments 2 and 5 are larger, as well as misaligned compared to the original segments. The semi-supervised segmentation algorithm is able to identify the smaller ranges of Segments 2 and 5 better and retains the original shapes and alignments of the original segments.

From Figures 4.13 and 4.14 it can be seen that the centroid endpoints of the semi-supervised segmentation analysis is once again close to the original centroids. The unsupervised k-means algorithm's centroids migrate to different positions for Segments

Distortion weight (w)	Segment 1	Segment 2	Segment 3	Segment 4	Segment 5	Segment 6
0 (Standard k-means clustering)	64.42%	74.91%	50.92%	64.28%	70.94%	50.53%
0.1	61.98%	68.12%	54.38%	51.18%	79.48%	53.06%
0.2	46.55%	64.31%	54.38%	57.04%	72.93%	55.51%
0.3	59.21%	62.54%	54.23%	63.21%	81.6%	53.3%
0.4	61.5%	38.43%	54.38%	71.34%	79.67%	55.48%
0.5	65.87%	20.56%	54.8%	80.56%	89.01%	55.38%
0.6	77.43%	27.63%	55.51%	84.01%	91.79%	58.15%
0.7	83.99%	28.82%	62.45%	92.57%	91.33%	60.84%
0.8	96.31%	38.53%	68.33%	96.6%	95.56%	63.62%
0.9	92.91%	55.08%	73.13%	91.06%	69.1%	72.17%

TABLE 4.8: Segment-specific accuracy results using PPA

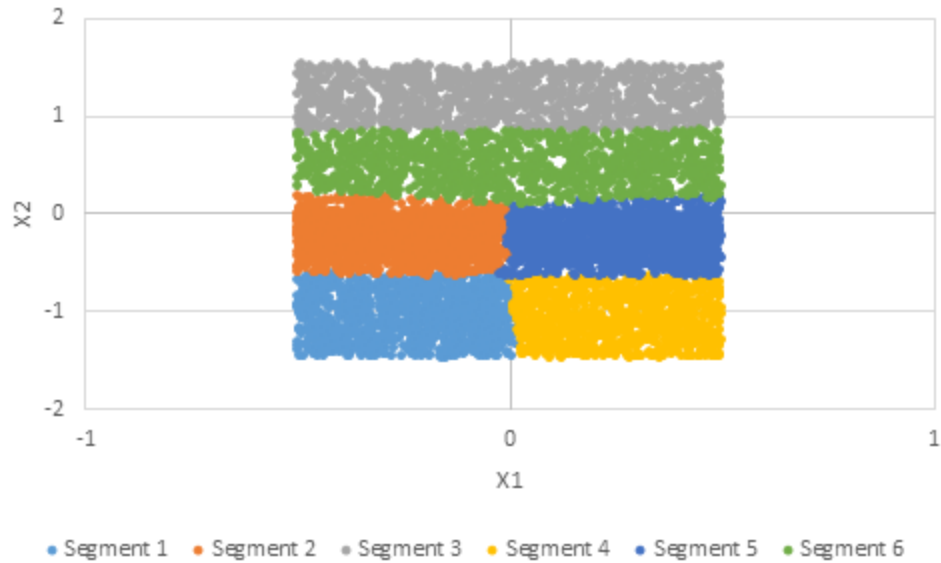


FIGURE 4.11: Unsupervised segmentation result for the second data set

3 and 6, but are able to relatively accurately find the endpoints of the rest of the segments. It can be seen that the sizes of the markers differ between the two charts. This is driven by differences in the number of iterations required by the two algorithms. The unsupervised algorithm required many more iterations when compared to the semi-supervised algorithm, and the starting size of its markers therefore needed to be larger.

From the results of the second data set, we have learned that the addition of the supervised element can affect the final segment results in two ways - it affects both the movement and final positions of the centroids, but can also overcome the "equal size" limitation of the standard k-means algorithm in the right circumstances.

It must be noted that we do not imply through this observation that more than one

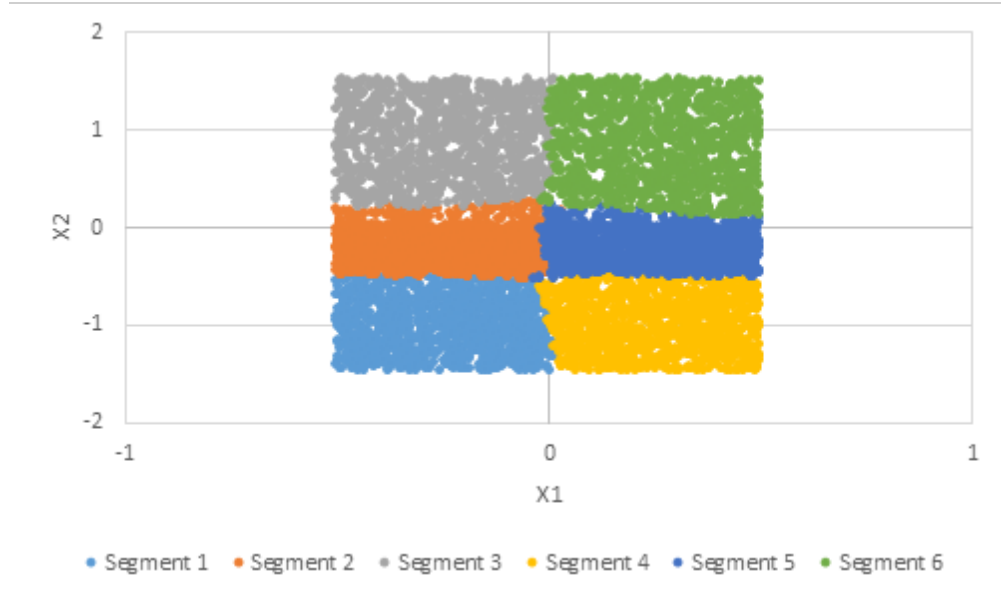
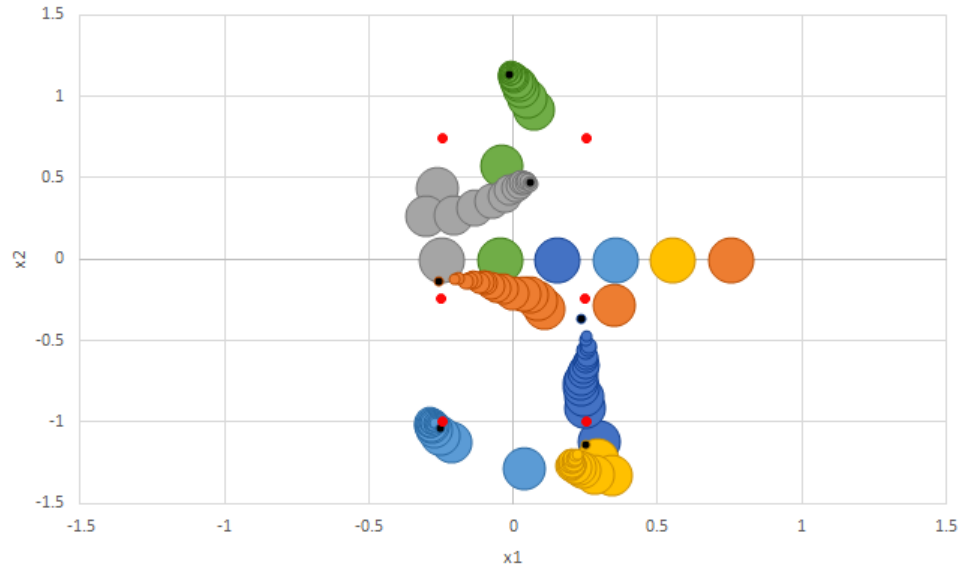
FIGURE 4.12: Semi-supervised segmentation result for the second data set ($w = 0.9$)

FIGURE 4.13: Unsupervised centroid movements for the second data set

element was changed when comparing the two methodologies. This would mean that the well-known principle of *ceteris paribus* (i.e. a true comparison keeps all things equal except for the element that is tested), was compromised. The only element in this comparison that was changed, was the technique used to identify the original segments (i.e. standard *k*-means vs. SSSKMIV). We can therefore conclude that this is a valid comparison and conclusion. The *result* we observed was that both the movement and final positions of the centroids were impacted. That is, the impact of changing the segmentation technique, resulted in more than one difference in the outcome.

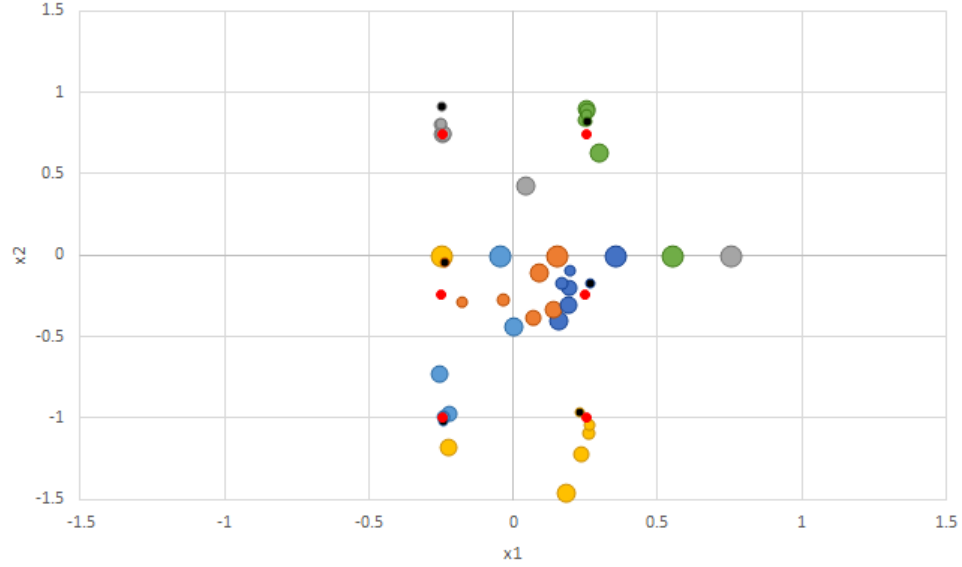


FIGURE 4.14: Semi-supervised segmentation centroid movements for the second data set ($w = 0.9$)

4.4.3 Results data set 3: Two concentric circles

Table 4.9 shows a summary of the accuracy with which segments were identified for the third data set. As expected, standard k-means clustering is unsuccessful with finding the generated segments due to its non-globular properties. However, once the distortion weight, w , exceeds 0.5, the supervised element allows the algorithm to identify the segments perfectly.

Distortion weight (w)	Average PPA	Average ARI	Nr. Iterations
0 (Standard k-means clustering)	51.96%	0.11%	20
0	51.96%	0.11%	20
0.1	51.96%	0.11%	20
0.2	51.96%	0.11%	20
0.3	51.96%	0.11%	20
0.4	51.96%	0.11%	20
0.5	79.9%	35.96%	20
0.6	100.%	100.%	20
0.7	100.%	100.%	20
0.8	100.%	100.%	20
0.9	100.%	100.%	20

TABLE 4.9: Results for the third data set

Table 4.10 shows the segment-specific accuracy using PPA. Note that percentages are based on the number of observations in each original data set. The accuracy from

$w = 0.6$ is clear from these statistics as well.

Distortion weight (w)	Segment 1	Segment 2
0 (Standard k-means clustering)	50.61%	53.25%
0.1	50.61%	53.25%
0.2	50.61%	53.25%
0.3	50.61%	53.25%
0.4	50.61%	53.25%
0.5	66.36%	92.86%
0.6	100.0%	100.0%
0.7	100.0%	100.0%
0.8	100.0%	100.0%
0.9	100.0%	100.0%

TABLE 4.10: Segment-specific accuracy results using PPA

Figures 4.15 and 4.16 show examples of results of unsupervised and semi-supervised segmentation analysis on the third data set respectively. The weight used for this example is $w = 0.6$. The reason for the low accuracy obtained by the unsupervised k-means algorithm is clear from the graphs, whilst the semi-supervised algorithm overcomes the limitations completely. Distortion weight $w = 0.6$ is used in Figure 4.16 since (as can be seen from Table 4.9) this is the lowest value for w where the semi-supervised algorithm identifies both segments perfectly.

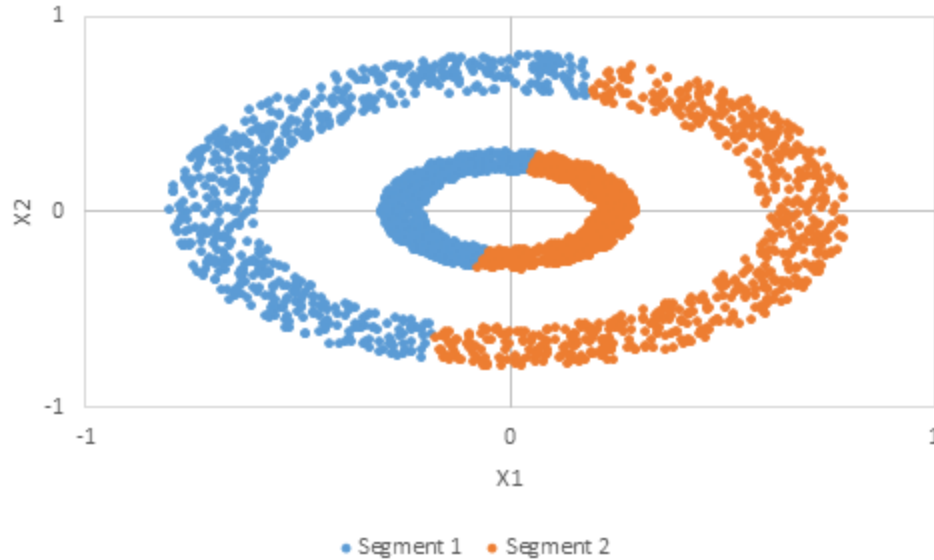
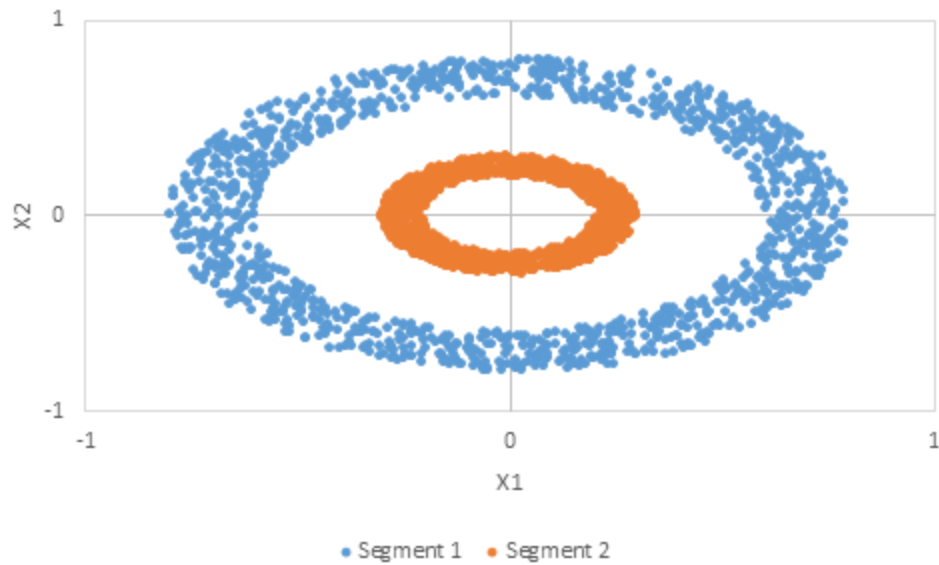


FIGURE 4.15: Unsupervised segmentation result for the third data set

From Figures 4.17 and 4.18 the large discrepancy in the positions of the final segment centroids of the unsupervised k-means algorithm compared to the actual centroids can be clearly seen. Due to the standard k-means algorithm's limitations, the centroids

FIGURE 4.16: Semi-supervised segmentation result for the third data set ($w = 0.6$)

move to a position that is completely incorrect, whilst the semi-supervised segmentation algorithm positions its centroids very close to the generated means of (0,0).

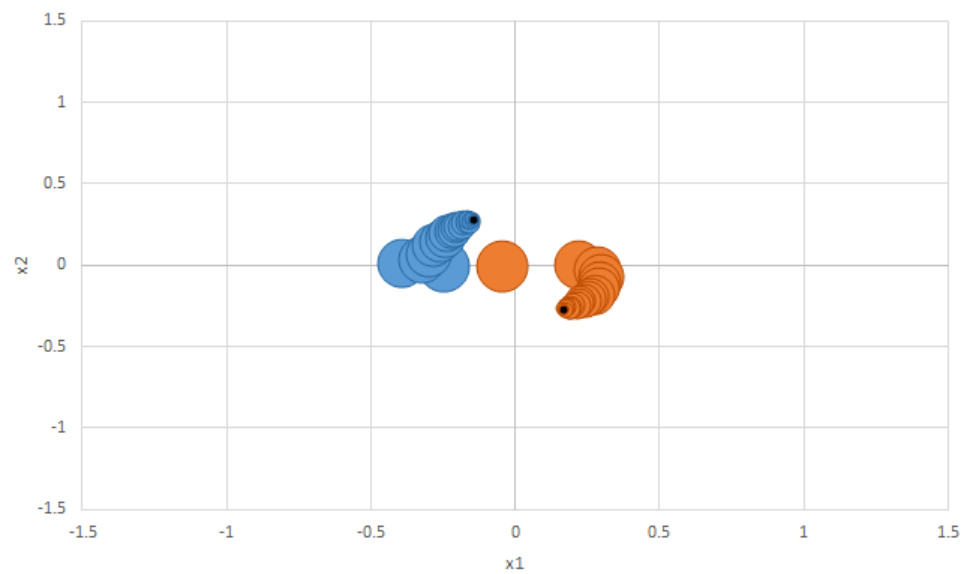


FIGURE 4.17: Unsupervised centroid movements for the third data set

The third data set showed us that the addition of the supervised element to the k-means algorithm's objective function allows it to overcome its requirement for globular segments in the right circumstances.

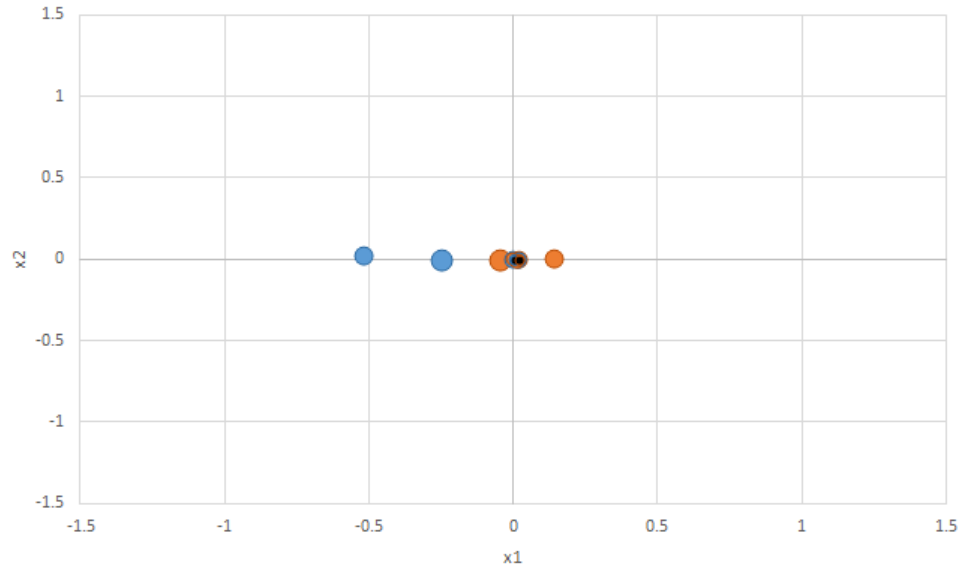


FIGURE 4.18: Semi-supervised segmentation centroid movements for the third data set ($w = 0.6$)

4.5 Conclusion

This section centred on the impact that the addition of the supervised element to the k-means algorithm's objective function has on segmentation results. Results were shown graphically in two dimensions and we have seen that, in the right circumstances, the semi-supervised algorithm is able to identify segments more accurately due to the additional information incorporated into its objective function. We have also seen that the semi-supervised algorithm is able to overcome some of the known shortcomings of the standard k-means algorithm. In summary, we have seen that (in the right circumstances):

- the addition of a supervised element to the objective function can improve the accuracy with which segments are identified, even if they are difficult to recognise spatially;
- the addition of a supervised element to the objective function can allow the k-means algorithm to overcome its "equal size" shortcoming; and
- the addition of a supervised element to the objective function can allow the k-means algorithm to overcome its reliance on globular segments.

The next chapter addresses the actual purpose of the semi-supervised segmentation algorithm, which is segmenting of data sources into groups on which separate predictive models can be developed.

Chapter 5

Comparison of SSSKM model accuracy to current segmentation techniques and alternative methodologies

5.1 Introduction

Chapter 4 shows that the semi-supervised segmentation algorithm is able to improve the accuracy with which segments in a data set are identified, as well as overcome some of the known weaknesses of k-means clustering. Although the semi-supervised segmentation algorithm is not a clustering algorithm, it is good to see from Chapter 4 that the additional information can be useful in improving cluster accuracy.

In this chapter we focus on the actual purpose of the semi-supervised segmentation algorithm, which is segmenting of data sources into groups, from which separate predictive models can be developed. As noted in Section 3.16, when the information value, or IV, is used as the distortion factor for the SSSKM algorithm, we call the algorithm SSSKMIV (semi-supervised segmentation as applied to k-means using information value). When chi-squared is the distortion function, we call the algorithm SSSKMCSQ (semi-supervised segmentation as applied to k-means using chi-squared).

We compare the success of using the SSSKMIV and the SSSKMCSQ algorithms for segmentation to the results obtained by both supervised and unsupervised techniques currently used in industry applications. We then also compare the results obtained from

the segmentation-based techniques to popular non-linear classification techniques, to understand how well these compare in terms of performance.

5.1.1 Research contributions

In this chapter we will contribute the following:

- We apply the proposed SSSKMIV and SSSKMCSQ algorithms to 6 data sets from various disciplines.
- We compare the success of segmenting, by means of the SSSKMIV and SSSKMCSQ methodologies to segmentation, using supervised segmentation with decision trees and unsupervised segmentation with standard k-means. Success is measured based on the improvement in the validation (or "holdout") set's Gini coefficient, when models are separately developed on the proposed segments.
- We analyse and compare the results and point out observed trends of the SSSKMIV and SSSKMCSQ algorithms.
- We apply popular alternative, non-linear classification techniques to the six data sets and compare the performance of these techniques to the segmentation-based techniques.
- We indicate observed trends of the segmentation-based techniques' performance compared to popular non-linear techniques.

5.2 Data used

We will make use of six data sets. The data used here will be representative of industry-related data from various disciplines. This means that the number of characteristics, or explanatory variables (m) that are considered will be larger than two (compare Section 4.2), which also means that segments cannot be illustrated visually as was done in Chapter 4. The data sets were mainly obtained from repositories that are freely available on the internet, but one data set was also obtained from a local South African bank. Target variables will be binary in all cases so as to facilitate the use of the SSSKMIV algorithm.

The six data sets we will use are itemised below, with the number of observations (n) and the number of explanatory variables (m) shown next to the descriptions.

- A data set from one of South Africa's largest four banks, containing information on customers and their responses to direct marketing campaigns ($n = 4720$, $m = 24$).
- A data set on computational biology, obtained from the UCI Machine Learning Repository (Lichman, 2013), containing information about physicochemical properties of protein tertiary structures ($n = 45730$, $m = 9$).
- A data set containing credit application data, obtained from kaggle.com (*Kaggle Website*, 2016), used in its public competition "Give me some credit" ($n = 150000$, $m = 10$).
- A data set from the UCI Machine Learning Repository (Lichman, 2013) that contains properties of red and white wines as well as the wine quality as evaluated by expert tasters ($n = 6497$, $m = 11$).
- A data set on game theory, obtained from the UCI Machine Learning Repository (Lichman, 2013), called "Chess (king-rook vs. king) Data Set". We also refer to it as the chess endgame data set ($n = 28056$, $m = 6$ before adding derived variables and $m = 18$ after adding derived variables).
- A data set on bodily injury liability insurance (*Kaggle Website*, 2016) for customers operating a motor vehicle, which predicts whether an insurance claim will be made and paid out based mostly on information about the vehicle ($n = 7.75\text{million}$ before sampling, $n = 14782$ after sampling, $m = 12$).

We will define each data set in more detail in the next sections.

5.2.1 Direct marketing data set

The first data set to be analysed was obtained from one of South Africa's largest four banks. The data set contains information about the bank's customers, the products they have with the bank, and their utilisation and behaviour on those products. The target variable is binary, and indicates whether the customer responded to a direct marketing campaign for a personal loan or not.

Predictive models are used in direct marketing to predict the propensity of customers responding positively to a personalised mail (or some other form of personalised solicitation) promoting some of the bank's products. These models help the bank save money by enabling it to target only customers who will most likely be interested in the product that is advertised, and avoid frustrating customers who will not be interested. Segmentation within direct marketing analytics is done with a number of objectives in

mind. Obviously improvement in the power of the predictive models will improve the efficiency of marketing campaigns, but in many instances segmentation allows the bank to be more effective in their marketing campaigns by communicating differently to one segment compared to the next. For this reason, clustering algorithms are quite popular in direct marketing analytics (Brito et al., 2015, Casabayó et al., 2015, Dutta et al., 2015).

Tables B.1 to B.3 in Appendix B provide a data dictionary of the fields that are used within this data set. Appendix B is used in this Chapter for all information that cannot be practically incorporated into the main body of the chapter.

5.2.2 Protein tertiary structures data set

In computational biology one of the persistent challenges continues to be the prediction of tertiary structures of proteins (Fraenkel, 1993, Mishra et al., 2014, Rana et al., 2014, Salzberg et al., 1998, Unger and Moulton, 1993). Proteins assume three dimensional tertiary structures and are therefore complex in nature. Structures are further influenced by a number of physicochemical properties which further complicates the task of accurate prediction (Anfinsen, 1973). Protein structure prediction algorithms (PSPAs) are algorithms that attempt to predict the tertiary structure of proteins (Mishra et al., 2014). These prediction algorithms have been refined over a number of years (Dhingra and Jayaram, 2013, Jayaram et al., 2012, Kim et al., 2004, Lambert et al., 2002), but will still deviate when compared to samples of actual, experimentally determined structures. One way of measuring such deviations are through the root-mean-square-deviation (RMSD) (Cozzetto et al., 2009, Mishra et al., 2014, Rana et al., 2014). Note that the protein structure prediction algorithms are in no way related to predictive modelling as defined in this thesis, since it is specific to the field of protein assessment.

The protein structure data set that will be used for this study was obtained from the UCI machine learning repository and contains results of experiments performed by the Protein Structure Prediction Center (PSPC, 2015) on the latest protein structure prediction algorithms. These experiments were labeled the "Critical Assessment of protein Structure Prediction" (CASP) experiments (Garcia-Moreno, 1995–2014). The data set contains various physicochemical properties of proteins, and the target variable is based on the RMSD measurement, indicating how much the predicted protein structures deviate from experimentally determined structures. The target that will be used for our purposes is binary, and indicates that the RMSD has exceeded a certain value ($\text{RMSD} > 7.5$), which implies an inaccuracy that is greater than normal. The goal in this study

is therefore to determine what physicochemical properties cause PSPAs to deviate more than the norm from experimentally determined protein structures.

Table B.4 shows the data dictionary for this data set. Other than standardising the explanatory variables and deriving the binary target variable, no further changes are made to this data set before segment analysis.

5.2.3 Credit application data set

The next data set that is used was obtained from Kaggle.com (*Kaggle Website*, 2016). The data set is publicly available, and was used in its competition "Give me some credit" which ran from September to December 2011. The data set contains a number of characteristics of customers who applied for credit, and the target variable is binary, indicating whether the customer experienced a 90 days or worse delinquency. The data set is used in a number of studies covering various areas of predictive modelling (Bahnsen et al., 2014, Sharma, 2013, Sitar et al., 2014, Zhou and Wang, 2012). Table B.5 was obtained from the website, and details the information contained in the data set.

In order to ensure that the data can be utilised fully for predictive modelling, some small adjustments were made to the data set:

- From Table B.5 it can be seen that the fields `NumberOfDependents` and `Monthly-Income` are expected to be integer values. However, analysis has shown that some of the data points contain the text values "NA". For these observations the values were substituted with a 0 value. Many methods of imputing or otherwise dealing with such data exist, but since this is not the focus of this study and as it was seen that the target rate of the observations with 0 values compared relatively well with the observations with "NA" values, the substitution was deemed suitable for our purposes.
- In line with Section 3.6, the values of all characteristics were standardised.

The data consist of a training set with 150,000 observations. This is more than what is needed to perform accurate cluster and predictive model analysis, so a random sample of 45,000 was drawn to reduce computation time. Given the number of iterations that are tested, this sampling saved weeks in computation time without any indication of an impact on the accuracy of the results.

5.2.4 Wine quality data set

This data set was obtained from the UCI Machine Learning Repository (Lichman, 2013). The data contain physicochemical properties of wines that is extracted through analytical tests that can be easily performed on most wines. The data set was first used by Cortez et al. (2009) and was collected between May 2004 and February 2007 (Cortez et al., 2009). The target variable is derived from a score between 0 and 10 which indicates the quality of the wine as scored by tasting experts. The binary target variable that is used for this analysis indicates whether the score is greater than six, thereby indicating a great quality wine (only 20% of the wines score greater than 6).

Table B.6 shows the data dictionary for the input variables used in this data set.

The repository consisted of two data sets, one for white wines and one for red. For the purposes of this exercise, the two data sets were combined.

Other than normalising the input variables, no other adjustments to the data sets were made.

5.2.5 Chess (king-rook vs. king) data set

The next data set that will be considered is based on game theory and was obtained from the UCI Machine Learning Repository (Lichman, 2013). The data set is an "Endgame database", which is a table of stored game-theoretic values for the legal positions of the pieces on a chess board. In this endgame, first described by Clarke (1977), the white player has both its king and its rook left, whilst the black player only has its king left - it is widely known as the "KRK endgame" and is still the focus of many studies (Bain, 2014, Bramer, 2014, Cohen, 2014). The database stores the positions of each piece as well as the number of moves taken to finish the game from those positions assuming minimax-optimal play (black to move first). Minimax-optimal play is an algorithm often used by computers to obtain the best combination of moves in a chess game and is based on the minimax game theory introduced by Neumann (1928). More information on this can be found in a number of texts (Casti and Casti, 1996, Osborne and Rubinstein, 1994, Russell and Norvig, 2003). Table B.7 details the information available in the data set. The target variable is binary, and indicates whether the game will be completed within 12 moves or less - which is less than the average number of moves recorded.

From Table B.7 it can be seen that the information provided is clearly not in the right format to use in a segmentation analysis as it contains characters and the target variable is in text format. The information provided can also be further expanded and enriched

through additional derived characteristics. Algorithms B.1, B.2 and 5.1 summarise the steps taken to prepare the data for segment analysis and predictive modelling. Algorithm B.1 shows how the "Rank" values are enumerated (i.e. changed from text to the represented numerical values). Algorithm B.2 enumerates the target variable whilst Algorithm 5.1 derives some additional characteristics as described in Table 5.1. The variables derived in Algorithm 5.1 can be grouped into four types:

- Row distances: The number of rows between each of the parts. Since the king can only move one block at a time, the distance between the parts (especially the kings), can play a large part in how long the game could still take.
- Column distances: This is added for the same reason as the row distances, as described above.
- Total distances: The total distance adds the distance in rows to the distance in columns. This provides another indication of how long the game could still progress.
- Diagonal indicators: Where the total distances above are more relevant to the king pieces, the diagonal indicators are more relevant to the rook. These indicate if the two pieces are on the same diagonal of the chess board (the rook can only move diagonally). If the white king and white rook are currently on the same diagonal, the white rook could be protecting the white king from attack, which provides valuable information. If the black king and white rook are on the same diagonal, the black king may already be in check - also valuable information which may have an impact on the expected length of the remaining game.

5.2.6 Insurance claim prediction data set

Another data set that was obtained from *Kaggle Website* (2016) contains information about bodily injury liability insurance. The competition was named "Claim Prediction Challenge (Allstate)" and concluded in 2011. The binary target that will be used in this data set indicates whether a claim payment was made. The explanatory variables have been hidden, but according to the website, it contains information about the vehicle to which the insurance applies as well as some particulars about the policy itself. Because the firm who donated this data required anonymity of the data fields, it is not possible to provide a detailed data dictionary, but Table B.8 summarises the known information about the data.

Algorithm 5.1 Deriving further characteristics for chess (king-rook vs. king) data set

$$\text{RankBlocks}_{WKBK} = WKR - BKR$$

$$\text{RankBlocks}_{WRBK} = WRR - BKR$$

$$\text{RankBlocks}_{WRWK} = WRR - WKR$$

$$\text{FileBlocks}_{WKBK} = N_WKF - N_BKF$$

$$\text{FileBlocks}_{WRBK} = N_WRF - N_BKF$$

$$\text{FileBlocks}_{WRWK} = N_WRF - N_WKF$$

$$\text{TotalBlocks}_{WKBK} = \text{abs}(\text{RankBlocks}_{WKBK}) + \text{abs}(\text{FileBlocks}_{WKBK})$$

$$\text{TotalBlocks}_{WRBK} = \text{abs}(\text{RankBlocks}_{WRBK}) + \text{abs}(\text{FileBlocks}_{WRBK})$$

$$\text{TotalBlocks}_{WRWK} = \text{abs}(\text{RankBlocks}_{WRWK}) + \text{abs}(\text{FileBlocks}_{WRWK})$$

If $\text{abs}(\text{RankBlocks}_{WKBK}) = \text{abs}(\text{FileBlocks}_{WKBK})$ **Then** $WKBK_{Diag} = 1$

else $WKBK_{Diag} = 0$

If $\text{abs}(\text{RankBlocks}_{WRBK}) = \text{abs}(\text{FileBlocks}_{WRBK})$ **Then** $WRBK_{Diag} = 1$

else $WRBK_{Diag} = 0$

If $\text{abs}(\text{RankBlocks}_{WRWK}) = \text{abs}(\text{FileBlocks}_{WRWK})$ **Then** $WRWK_{Diag} = 1$

else $WRWK_{Diag} = 0$

The data set itself that was provided has many observations, but events are rare (probability of occurrence around 1%). In order to reduce unnecessary computation time, the data set was therefore oversampled, which increased the event rate to around 33%. Oversampling in cases where events are rare is a common technique applied in the industry (Chang et al., 2013, Taft et al., 2009, Yap et al., 2014).

No significant changes were made to the data for this data set. Continuous input variables were standardised based on the oversampled set.

5.3 Comparing improvement in Gini coefficient due to segmentation

In order to compare the effectiveness of the three segmentation methodologies, namely supervised segmentation through decision trees (no unsupervised element), unsupervised segmentation ($w = 0$) through k-means clustering and semi-supervised segmentation (w ranging from 0.1 to 0.9) through SSSKMIV and SSSKMCSQ, each algorithm needs to have an opportunity to segment data sets, and then compare the improvement in accuracy obtained. This process is relatively generic no matter what segmentation methodology is used, and is illustrated in Figure 5.1

Variable Name	Variable Description	Data Type
$RankBlocks_{WKBK}$	The number of rows between the white king and the black king.	Integer
$RankBlocks_{WRBK}$	The number of rows between the white rook and the black king	Integer
$RankBlocks_{WRWK}$	The number of rows between the white rook and the white king	Integer
$FileBlocks_{WKBK}$	The number of columns between the white king and the black king	Integer
$FileBlocks_{WRBK}$	The number of columns between the white rook and the black king	Integer
$FileBlocks_{WRWK}$	The number of columns between the white rook and the white king	Integer
$TotalBlocks_{WKBK}$	The total number of blocks that the white king and black king are apart	Integer
$TotalBlocks_{WRBK}$	The total number of blocks that the white rook and black king are apart	Integer
$TotalBlocks_{WRWK}$	The total number of blocks that the white rook and white king are apart	Integer
$WKBK_{Diag}$	The white king and black king are on the same diagonal	Binary
$WRBK_{Diag}$	The white rook and black king are on the same diagonal	Binary
$WRWK_{Diag}$	The white rook and white king are on the same diagonal	Binary

TABLE 5.1: Additional derived characteristics for the chess (king-rook vs. king) data set

As illustrated, first of all, the data is divided into a development and a validation set. The development set is used to perform the segmentation on and develop the predictive models, whilst the validation set is used to test the lift in model accuracy (as measured by the Gini coefficient). We refer to the lift in the Gini coefficient obtained through segmentation on the validation set simply as "lift" throughout the thesis. The development set and validation set were generally sampled with even sizes (i.e. 50% each). Although the norm is to usually use larger samples for the development set, the validation Gini is used in this study as the ultimate measure of success, and the larger validation set size was therefore preferred to ensure Ginis are not affected by low sample size.

Once the segmentation is done on the development set, a single, unsegmented logistic regression model is fit on the entire development set. To this end stepwise regression was applied using SAS[®] software's Proc Logistic. The significance level for entry of parameters was set at 0.1, whilst the significance level for removal was set at 0.05.

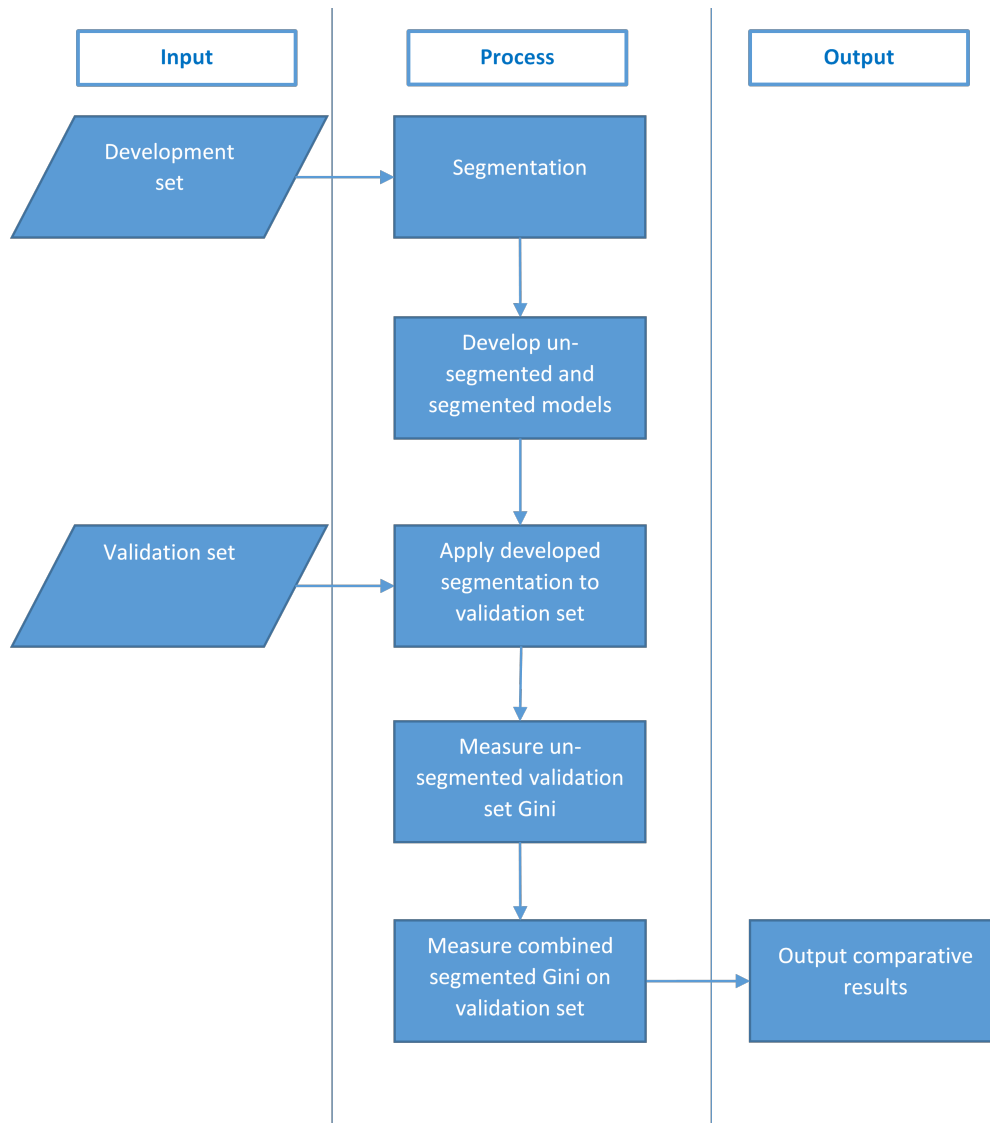


FIGURE 5.1: Illustration of segmentation comparison process

The resulting model provides the reference model against which the segmented models' accuracy will be tested to measure lift. The development set is also split into the different segments, on which separate logistic models are developed (using the same settings as described above).

Once all models have been developed, they are applied to the validation set. The un-segmented model is applied to the full validation set to obtain the reference Gini, whilst the segmented models are applied individually to each corresponding segment. In order to measure the combined Gini of the segmented models on the validation set, the predicted probabilities of all segments are combined, and the Gini is calculated on the overall, combined set. Once this is done, the unsegmented Gini can be compared to the combined, segmented Gini and the lift calculated.

5.4 Segmenting using a decision tree (supervised segmentation)

Decision trees and its use in segmentation is discussed in Section 2.7.1. SAS[®] software's Proc Split was used to segment the data sets using decision trees. Since the goal is to develop predictive models (which cannot be done effectively on very small samples), the "Splitsize" option was used to set the minimum number of observations in a leaf and control the number of segments created. The "Splitsize" option in SAS[®] software's Split procedure does not allow a node to split if the split will result in a leaf that will contain less than the indicated number of observations. The procedure will still consider other options for splitting the node, but will simply eliminate those that result in leaves that will breach the indicated "Splitsize" value. More information on this can be found in the help files for the SAS[®] Enterprise Miner[™] software. The value that was passed to the "Splitsize" option is initially calculated as $Sp_{size} = \frac{n_{dev}}{2K}$ where n_{dev} is the number of observations in the development set and K is the selected number of segments. This is simply done to initiate the decision tree process and is based on the logic that the number of observations per segment will be directly proportional to the size of the data set and indirectly proportional to the number of segments.

Algorithm 5.2 Do decision tree segmentation

Input: K {as number of segments}

Input: n_{dev} {as number of observations in the development set}

Input: {Development set}

Input: {Validation set}

$$Sp_{Size} = \frac{n_{dev}}{2K}$$

Do initial decision tree on development set with SplitSize = Sp_{Size}

Ascertain number of resulting segments K_{Dtree}

while $K_{Dtree} < K$ **do**

$$Sp_{Size} = Sp_{Size} \times \frac{K_{Dtree}}{K}$$

Redo decision tree on development set with SplitSize = Sp_{Size}

Ascertain number of resulting segments K_{Dtree}

end while

while $K_{Dtree} > K$ **do**

Prune the leaf that minimises information loss

$$K_{Dtree} = K_{Dtree} - 1$$

end while

Apply resulting tree with $K_{Dtree} = K$ to validation set

Output: Validation set split into K decision tree segments

Algorithm 5.2 illustrates the logic used to segment the data sets using a decision tree. The value of K is provided by the user, and in this case will be a value between two and ten (see Section 5.7). The first while loop reduces the value of Sp_{Size} continually until the number of segments produced by the decision tree is at least equal to the number of segments required, K . In many instances after the first while loop, the number of segments produced by the decision tree may be more than K . In this case the decision tree is pruned one leaf at a time. The impact of pruning each leaf on the information value is calculated in this step, and the leaf that minimises information loss is the one that is pruned. When both these loops are completed, the decision tree has formed K segments. With these steps, the algorithm aims to encourage the decision tree to create segments that are as large as possible to avoid the situation where segments are too small to facilitate robust predictive model development.

5.5 Segmenting using k-means (unsupervised segmentation)

Segmentation using k-means will be performed in a similar way as is described in Chapter 3. An unsupervised k-means algorithm (i.e. with $w = 0$) will be applied to the data set with the value of K pre-defined. This segmentation result of the development set will then be applied to the validation set through KNN smoothing (see Section 3.14 for details).

5.6 Segmenting using SSSKM (semi-supervised segmentation)

The process of segmenting using SSSKM is described in detail in Chapter 3. In order to ensure that a representative range of values for w will be investigated, we will run iterations on values of $w = 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8$ and 0.9 for each data set.

5.7 Comparison of segmented model development results

In order to obtain meaningful and robust results, the data sets described above will be put through several segmentation analysis iterations. Because the optimal number of segments are unknown for each data set, the value of K (i.e. the number of required segments) was iteratively increased from two to ten for each data set. For each semi-supervised segmentation algorithm, 20 segmentation exercises were performed to provide

the algorithm with the best chance to find a globally optimal solution (see Section 3.5). For the same reason, unsupervised segmentation (k-means) and decision tree segmentation was also performed multiple times to provide the best chance for globally optimal solutions. This means that close to 2000 iterations of segmentation were performed and almost 12000 models fit. The run time to complete a single data set was up to seven days¹ (depending on the size of the data set).

The reason why ten segments were seen as a reasonable maximum number of segments is due to the fact that segmentation for predictive models is mainly done to facilitate the use of simple, transparent predictive model methodologies (like logistic regression) that are easy to maintain. If more than ten models have to be maintained, the argument for using these techniques become weak. In cases where the increased accuracy is more important than the transparency and future maintenance of models, it is likely that other predictive modelling techniques, that do not require segmentation, will be preferred. Where the available data sets are small, segmenting the sets into many segments also reduces the number of observations available for model development in individual segments, which reduces model robustness. Similar sentiments on segmentation in practical applications are put forward by a number of texts (Anderson, 2007, Siddiqi, 2006).

As also described in Section 3.15, a number of statistics are available for each segmentation iteration after performing the semi-supervised segmentation algorithm. Most of these statistics are retained to be used for comparative purposes in the iterative result described above. Table B.9 summarises these statistics that are available.

We will now summarise and analyse the results found for each data set.

5.7.1 Direct marketing data set results

The results in this chapter will be similarly presented as was done in Chapter 4. This first data set will be used to explain the specific graphs and statistics used to summarise the results, and these explanations can be used to understand the results of all the additional data sets (in Sections 5.7.2 to 5.7.6) as well.

The first aspect that will be explored is the impact that an increase in the distortion weight has on relevant segment characteristics for both SSSKMCSQ and SSSKMIV. Figure 5.2 and Table B.10 illustrate this. On the x-axis, the label indicates the distortion weight. The first label (i.e. "unsupervised") represents the average results for an unsupervised (i.e. $w = 0$) run of both SSSKMCSQ and SSSKMIV. If enough iterations of each set of input parameters into the algorithm could be performed, all values of IV

¹The computer used had the following processing specifications: Intel® Core™ i5-3330 CPU @ 3.00GHz, 3001MHz, 4 Cores, 4 Logical Processors

and CH would be equal for $w = 0$. This is because both SSSKMCSQ and SSSKMIV are completely unsupervised for $w = 0$, and the algorithms are therefore identical. The small difference seen is due to the fact that the number of iterations had to be limited to reduce run time. As can be seen clearly from the graph, the differences are, however, negligible indicating that the number of iterations are sufficient to obtain meaningful results.

The last label (i.e. "supervised") represents the results for segmentation using a decision tree as described in Section 5.4. The labels in-between represents results from both SSSKMCSQ and SSSKMIV algorithms as applied to the data set with the indicated distortion weights (w). Note that all averages (of the measures explained below) are measured over results for all values of K (i.e. two segments up to ten segments). For the clustering algorithms, each data point therefore represent results from 180 (20×9) segmentation exercises.

The characteristics that are illustrated here are as follows:

- Percentage k-means decisions overridden for both SSSKMIV and SSSKMCSQ: These statistics show the average percentage of segment allocations that was changed due to the addition of the supervised element of either the SSSKMIV or the SSSKMCSQ algorithm in the last iteration of the algorithm (this is measured on the set used for segmentation). If the unsupervised element (i.e. Euclidian distance) corresponds to the supervised element in terms of what segment needs to be allocated to the observation, the value of this characteristic will be zero. It will only be set to one if the final assignment that would have been made by an unsupervised algorithm was changed due to the addition of the supervised element and the distortion weight. In short therefore it indicates the percentage of observations that was assigned to a different segment than unsupervised segmentation. An example of such a case is discussed in Section 3.9.6. These values do not make sense for unsupervised or fully supervised segmentation for obvious reasons. The average override percentage (over all values of K) is illustrated by bars in Figure 5.2.
- Average CH value: This is an average of the CH value (or pseudo-f statistic - see Section 2.12) for all iterations of the segmentation approach indicated, as applied to the validation set. As discussed in Section 2.12, the CH value provides an indication of the success of the segmentation from an unsupervised perspective.
- Average IV: This is an average of the information value (see Section 2.11.1) of the segmentation for all iterations of the segmentation approach indicated, as applied

to the validation set. The information value provides an indication of the success of the segmentation from a supervised perspective.

In Figure 5.2 and Table B.10 it can be seen that neither of the two semi-supervised segmentation algorithms are particularly successful in affecting the IVs or the CH values of the segmentation result. From the bars in Figure 5.2, it is clear that the increase in the supervised weight does have a large impact on the percentage of assignments overridden, but this does not seem to provide a big lift in IV, nor much of an effect on the CH value. A slightly increasing trend in the average IV is visible and a very slight decrease in the average CH value can be observed. There are no clear signs of overfit (see Section 3.14) for this data set as the average IV (as applied to the validation set) never decreases much, even for values of $w = 0.9$.

On the other hand, the decision tree succeeded in segmenting the population into segments that separate the target variable well. On average, an IV of 0.225 was obtained by decision tree segmentation. As can be seen from the figure and the table, this came at the cost of a lower average value for the CH value (as can be expected for supervised segmentation).

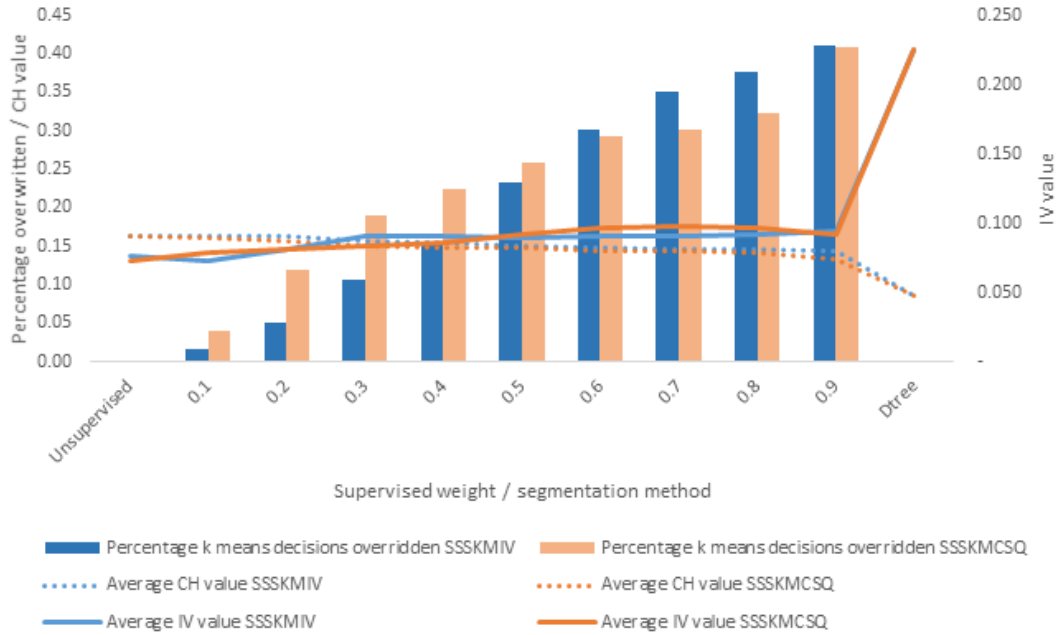


FIGURE 5.2: Comparison of segmentation characteristics by supervised weight

Figure 5.3 and Table B.11 compare the success of achieving a lift in the IV during segmentation with the improvement in Gini. The x-axis of Figure 5.3 is similar to Figure 5.2, but the y-axis on the left indicates the average Gini improvement which is compared to the average IV (represented by bar graphs) on the right axis. We have observed in Figure 5.2 and Table B.10 that the SSSKM techniques applied to this data

set were not particularly successful in obtaining high IVs. It can be seen from Figure 5.3 and Table B.11 that they were also not, on average, successful in generating an increase in Gini either. We can see from the figure that an increase in the distortion weight, w , actually does more damage than good, and that the combined average Gini of the segmented sets are lower than the Gini on the total data set for high values of w . The same cannot be said for the decision tree, though. As was seen in Figure 5.3 and Table B.11, the decision tree is able to generate high IVs and from Figure 5.3 and Table B.11 it is clear that it is also more successful in improving the Gini through segmentation as well.

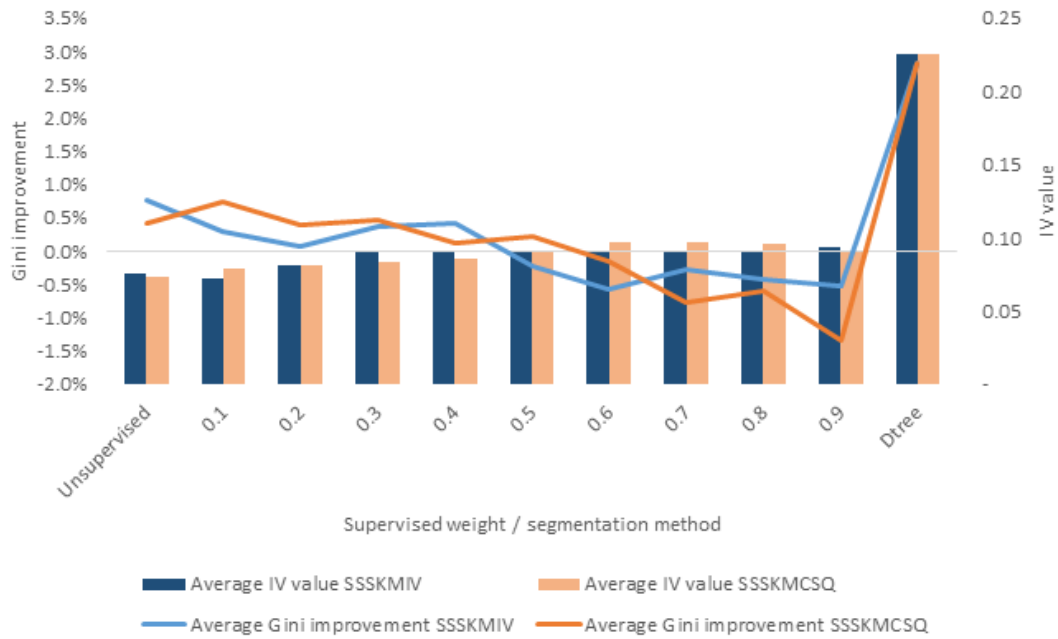


FIGURE 5.3: Success of target separation compared to Gini improvement by supervised weight

Figures 5.4 and 5.5 shows the success (as measured by improvement in Gini on the validation set) of segmentation by number of segments for each of the segmentation techniques compared here. The values on the x-axis indicate the number of segments, whilst the improvement in Gini due to segmentation is illustrated on the y-axis. Figure 5.4 is arguably of most value, as it shows the best (i.e. global optimum) solution for each technique, whilst Figure 5.5 shows the average. Tables B.12 and B.13 summarise these values in a table format. It is clear from these figures and tables that the decision tree is decidedly the best methodology that can be used to segment this data set for predictive modelling between the methodologies applied here. The decision tree is able to produce an improvement in Gini on the validation set of between 8% and 12%. The reference Gini (i.e. the unsegmented Gini) for this data set was only 22%, so this is a relative improvement of more than 35%.

Although we have seen the decision tree decisively outperforming other segmentation techniques, we will see later in this section that other techniques outperform on some of the other data sets.

When looking at Figure 5.4 and Table B.12, one can deduce that the SSSKMCSQ technique outperforms the SSSKMIV technique when more segments are created. Note that in order to make a formal conclusion, one would need to consider that the Gini coefficient is random, and include a hypothesis test. This was not done here, although the results are from a large number of iterations. When looking at the results overall the two techniques seem to be obtaining similar results.

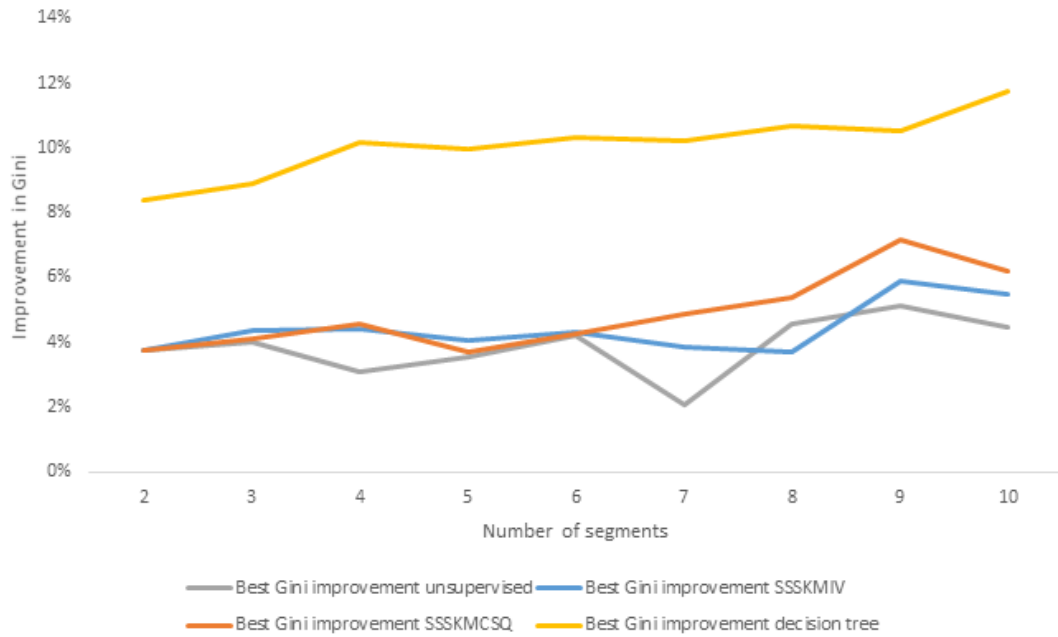


FIGURE 5.4: Comparison of best improvements in Gini

5.7.2 Protein tertiary structures data set results

Figure 5.6 and Table B.14 show a similar graph and table as was shown for the direct marketing data set in Figure 5.2 and Table B.10.

The first aspect we consider is the impact on the percentage k-means decisions that are overridden due to the value of the distortion weight, w . A clear increasing trend can be observed as the value of w increases. It is also clear that the SSKMCSQ algorithm tends to override more decisions compared to the SSSKMIV algorithm in all cases.

The increase in overrides causes the SSSKMCSQ algorithm to achieve higher average IVs at first, but the average IVs converge at $w = 0.6$. Overfitting starts to become evident from $w = 0.6$ for both SSSKMIV and SSKMCSQ as we can clearly see the

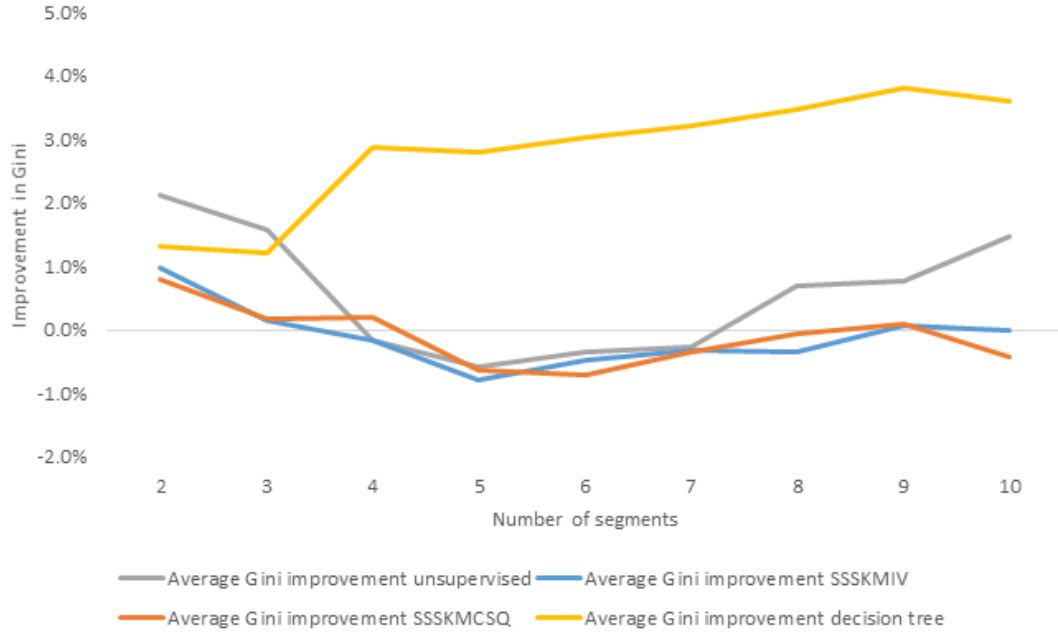


FIGURE 5.5: Comparison of average improvements in Gini

IVs (as measured on the validation set) starting to decrease from that point onward. At $w = 0.6$, where the average IVs are at its maximum, there is no clear distinction between the average IVs of either SSSKMIV or SSKMCSQ ²

The increased k-means override percentage of the SSKMCSQ algorithm causes the average CH values of this algorithm to be lower than the average CH values of the SSSKMIV algorithm. As discussed, this seems to coincide with higher IVs at first, but at $w = 0.6$, the SSSKMIV algorithm achieves higher CH values and similar IVs compared to the SSKMCSQ algorithm.

What is at first peculiar about the result shown in Figure 5.6 and Table B.14, is that it seems that the SSSKM algorithms actually achieve higher IVs (for optimal values of w) than the decision tree on this data set. This may seem quite surprising at first, as the decision tree is fully supervised and therefore exclusively focused on the target variable. However, since the process of segmentation differs quite substantially between the SSSKM algorithms and the decision tree, it is entirely plausible that this could be the case. Furthermore, not only do the SSSKM algorithms achieve higher IVs, but the CH values they achieve are much better than the decision tree at these high IVs.

Figure 5.7 and Table B.15 compares the IVs achieved to the improvement in average Gini. For the SSSKM algorithms, the correlation is apparent. The highest improvements

²No overfit was evident in Figure 5.2 for the direct marketing data set. Presumably this has more to do with the SSSKM algorithms' inability to separate the target variable than anything else.

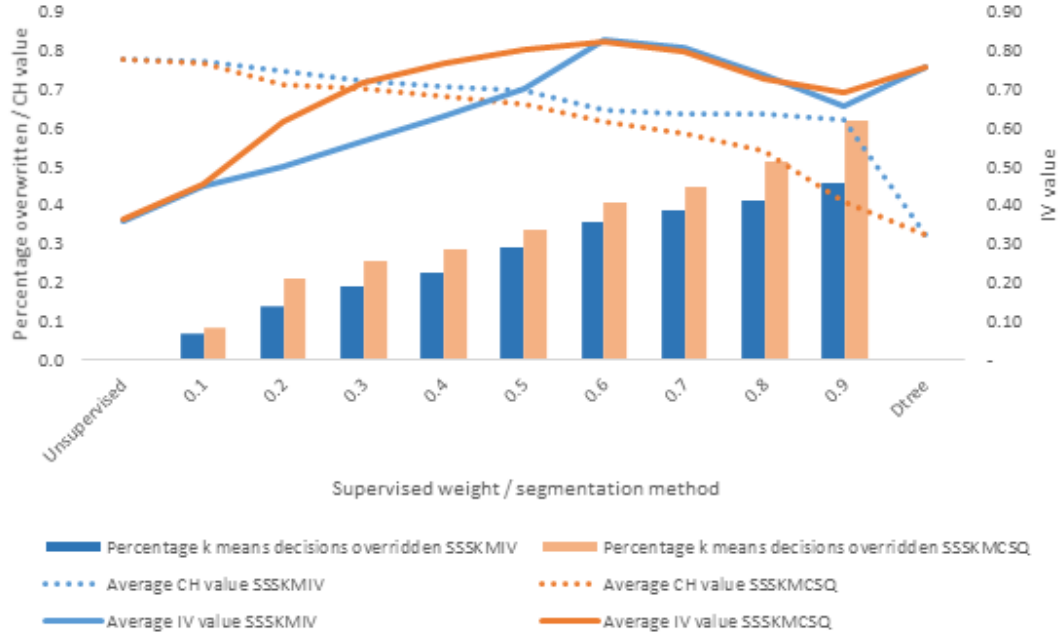


FIGURE 5.6: Comparison of segmentation characteristics by supervised weight

in Gini is obtained for the same values of w where the best IVs are obtained in Figure 5.6 and Table B.14.

What can also be seen is that the SSSKMIV algorithm outperforms the SSSKMCSQ algorithm at higher values of w . Ostensibly, this could be due to the fact that it maintains (on average) higher CH values whilst achieving similar IVs as observed from Figure 5.6 and Table B.14. At optimal values of w , it can be observed that the SSSKM algorithms outperform both unsupervised k-means and decision trees.

The superiority of the SSSKM algorithms on this data set as assessed by Gini improvement of segmented predictive models is confirmed by Figures 5.8 and 5.9 and Tables B.16 and B.17. In fact, the unsupervised k-means algorithm also outperforms the decision tree from six segments and higher. In these graphs and tables, it can also be seen that the SSSKMIV algorithm tends to outperform the SSSKMCSQ algorithm on average, with a clear outlier improvement in Gini when eight segments are selected. The optimal Gini improvement is between 2% and 11% on the overall model, which has a Gini of 59% on the validation set. Given the difficulty in improving models that are already quite predictive, this improvement could be valuable in many areas of application.

5.7.3 Credit application data set results

When analysing Figure 5.10 and Table B.18, an abnormal trend in the override percentage can be observed for the SSSKMIV algorithm. The percentage overridden decisions

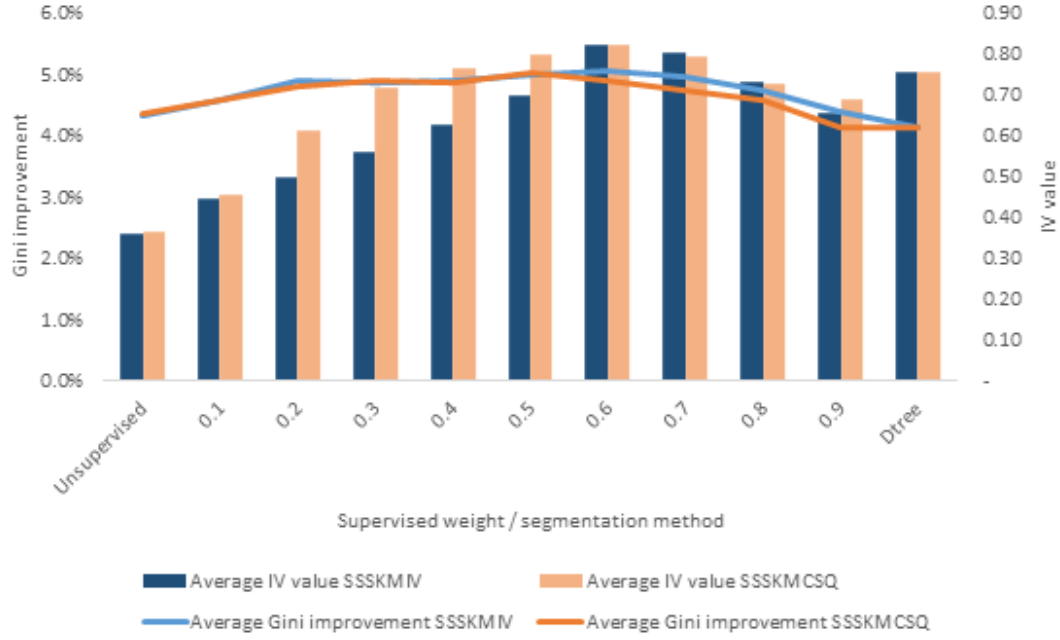


FIGURE 5.7: Success of target separation compared to Gini improvement by supervised weight

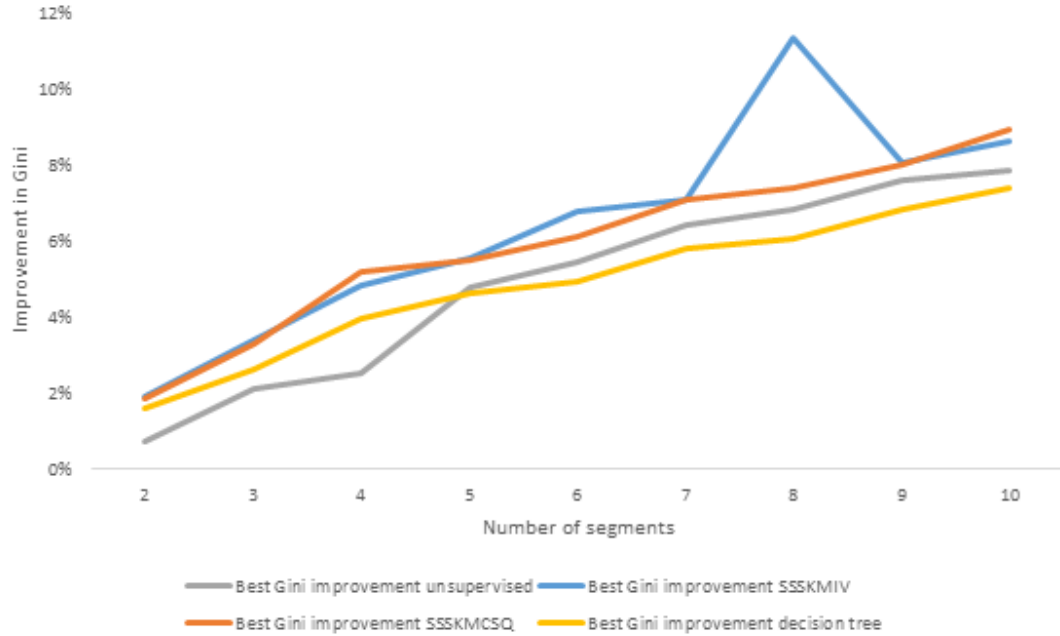


FIGURE 5.8: Comparison of best improvements in Gini

increases steeply up to $w = 0.5$ and then drops suddenly at $w = 0.6$. This warranted further investigation. It was observed that the result was due to the fact that, when the value of w exceeds 0.5, the actual position of the segment centroids (and therefore the position of the segments themselves) are affected. This means that, for values of w exceeding 0.5, the algorithm no longer overrides as many segment allocations, because

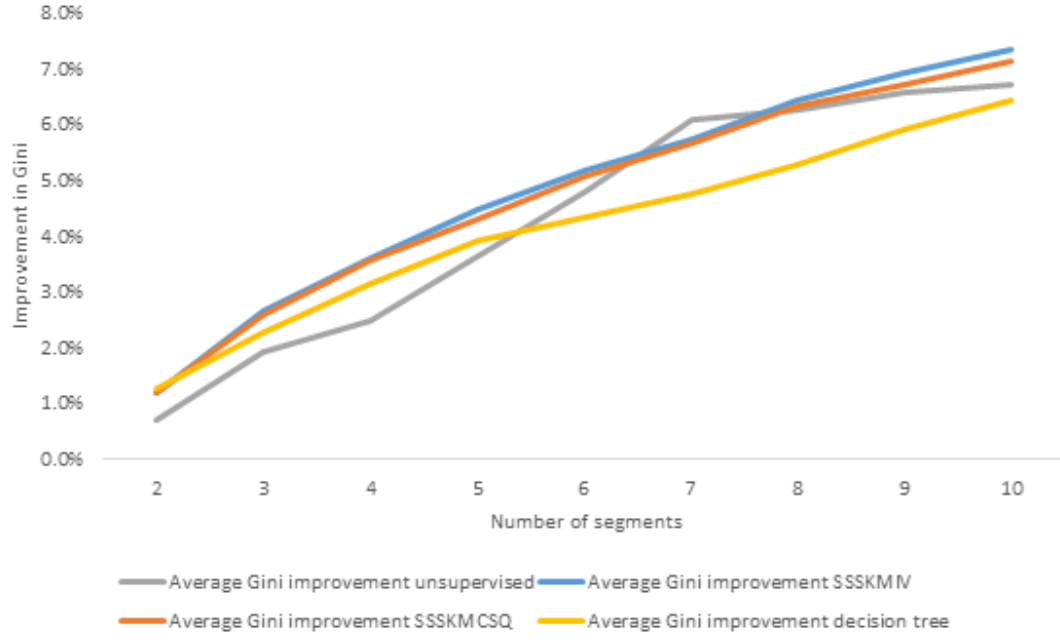


FIGURE 5.9: Comparison of average improvements in Gini

the position of the centroid is such that the unsupervised and supervised elements agree more often regarding segment allocation.

Similar to the results observed in Section 5.7.1, it is obvious from Figure 5.10 and Table B.18 that the SSSKM algorithms are not very effective in improving IVs as the value of w increases. The decision tree is, however, very successful, and obtains IVs that are, on average, more than three times higher than the SSSKM algorithms (even though it is achieved at expense of the CH value).

Very little overfit can be observed from the IVs for both SSSKM algorithms, even at high values of w .

While we can observe a high correlation between the success of achieving higher IVs and improving the Gini coefficient for the SSSKM algorithms in Figure 5.11 and Table B.19, the semi-supervised algorithms are not able to match the average Gini improvement achieved by decision tree segmentation.

In Figure 5.12 and Table B.20, it can be seen that both SSSKM algorithms outperform the decision tree in terms of Gini improvement when three segments are selected, but following that, the decision tree decidedly outperforms all other algorithms. It is interesting to see that the best improvement in Gini as achieved by the decision tree shows only marginal increases from values of K greater than four (the marginal increases can be confirmed from Table B.20).

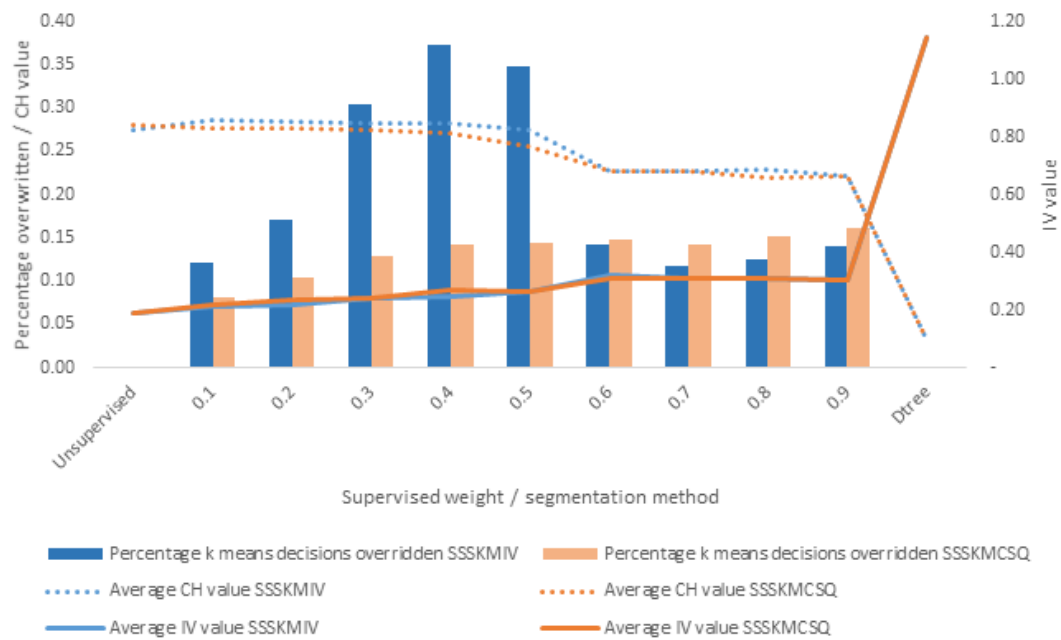


FIGURE 5.10: Comparison of segmentation characteristics by supervised weight

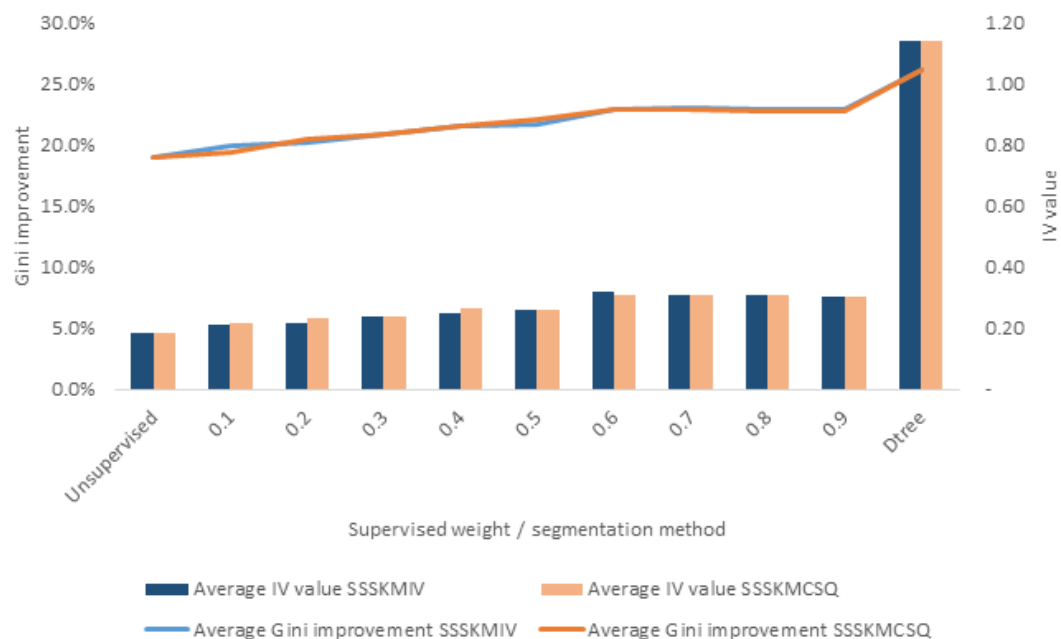


FIGURE 5.11: Success of target separation compared to Gini improvement by supervised weight

Figure 5.13 and Table B.21 illustrates that, on average, the decision tree seems to outperform both unsupervised segmentation and the SSSKM algorithms consistently. The improvement by decision tree segmentation of 33% on the benchmark of 38% on the validation set is quite remarkable, and shows that segmentation can, in some instances almost double the predictive accuracy as measured by the Gini coefficient.



FIGURE 5.12: Comparison of best improvements in Gini



FIGURE 5.13: Comparison of average improvements in Gini

5.7.4 Wine quality data set results

We analyse the impact of an increase in distortion weight on relevant segment statistics in Figure 5.14 and Table B.22 for the wine quality data set. A few recurring trends from Section 5.7.2 can be observed.

Firstly, the SSSKMCSQ algorithm again seems to override more segment allocations for lower values of w . This, in line with Section 5.7.2, causes the SSSKMCSQ algorithm to find segments with higher IVs for lower values of w , but (also similar to Section 5.7.2) at the cost of lower CH values. We can also observe some degree of overfitting from $w = 0.6$, which causes the validation set IVs to decrease.

On this data set, the decision tree finds slightly higher IVs compared to the SSSKM algorithms, but again at the price of very low CH values.

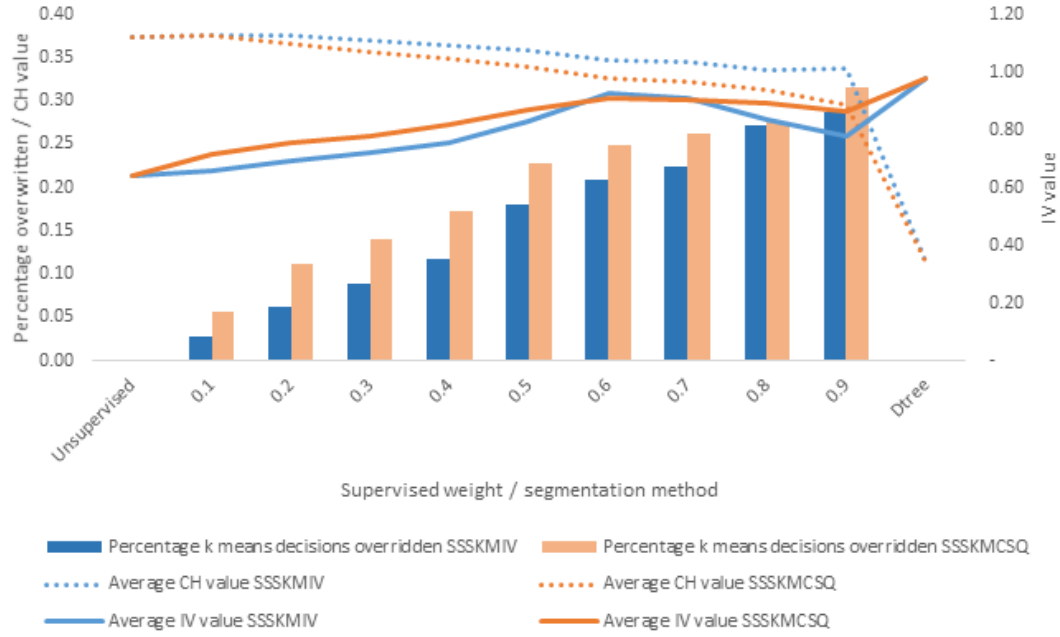


FIGURE 5.14: Comparison of segmentation characteristics by supervised weight

Since we have, with the previous few data sets, often observed that the success of the SSSKM algorithms in finding higher average IVs are strongly correlated with the average improvement in Gini, it is quite interesting to observe from Figure 5.15 and Table B.23 that the SSSKMCSQ algorithm manages to find its highest average Gini improvements for values of w between 0.2 and 0.3. When analysing the individual results underlying the averages shown in Figure 5.15 and Table B.23, it can be seen that the optimal value of w (as measured by maximum Gini improvement) differs depending on the number of segments selected. The reason for this could be that, at low values of w , the SSSKM algorithms are still able to preserve the CH value (unsupervised measure of success) quite well, whilst already improving the IVs (supervised measure of success).

What is also clear from Figure 5.15 and Table B.23 is that the average Gini improvement of the decision tree segmentation methodology is worse compared to both unsupervised and semi-supervised segmentation.

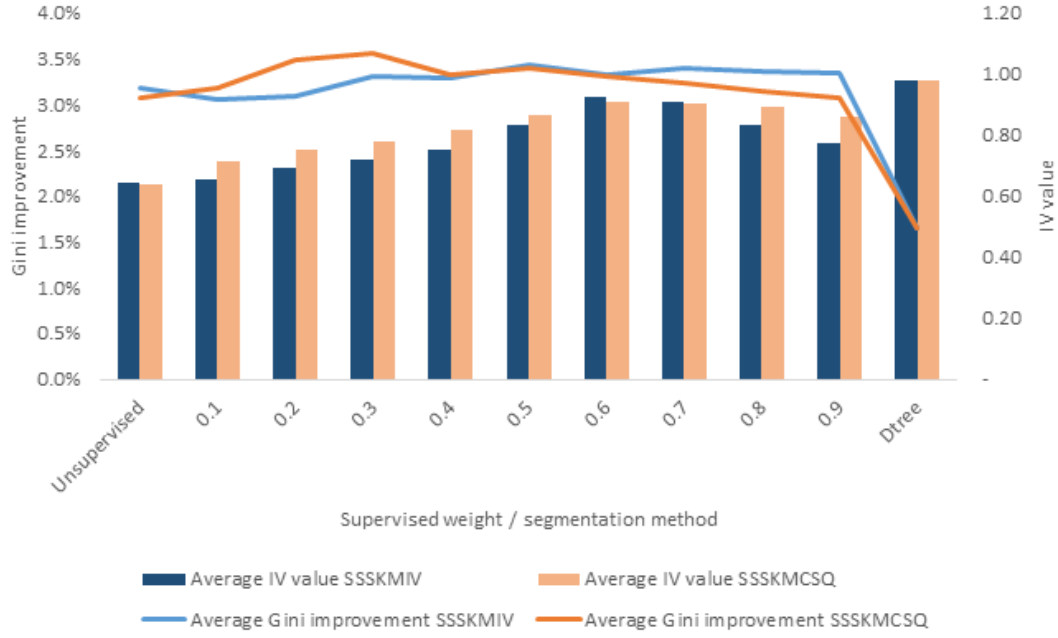


FIGURE 5.15: Success of target separation compared to Gini improvement by supervised weight

In Figure 5.16 and Table B.24, the best Gini improvements for each of the segmentation methodologies are shown by number of segments. Although the SSSKM techniques generally outperform unsupervised and fully supervised segmentation, unsupervised segmentation results are generally close to the SSSKM algorithms, and actually outperforms it in this sample for 5, 9 and 10 segments. It is clear to see from both Figure 5.16 and Figure 5.17 (as well as Table B.25) that the decision tree is not a suitable segmentation technique for this data set, as it is consistently outperformed by both unsupervised k-means and the SSSKM algorithms. A clear distinction between SSSKMIV and SSSKMCSQ can not be made in the results as they seem to achieve similar results.

For this data set, the reference validation set Gini (as developed on the entire set with no segmentation) is already high at 60.5%. Generating a further 6% to 7% in Gini (as is done here) is often quite challenging.

5.7.5 Chess (king-rook vs. king) data set results

When analysing Figure 5.18 and Table B.26, there are quite a few similarities to the same statistics illustrated in Figure 5.6 and Table B.14 from Section 5.7.2. Similar to the protein tertiary structure data, the SSSKMCSQ is able to form segments with high IVs for relatively low values of w . However, in this case, the SSSKMIV algorithm is not able to match any of the SSSKMCSQ average IVs for any value of w . The difference in IV between the highest average SSSKMCSQ IV and the decision tree average IV is also

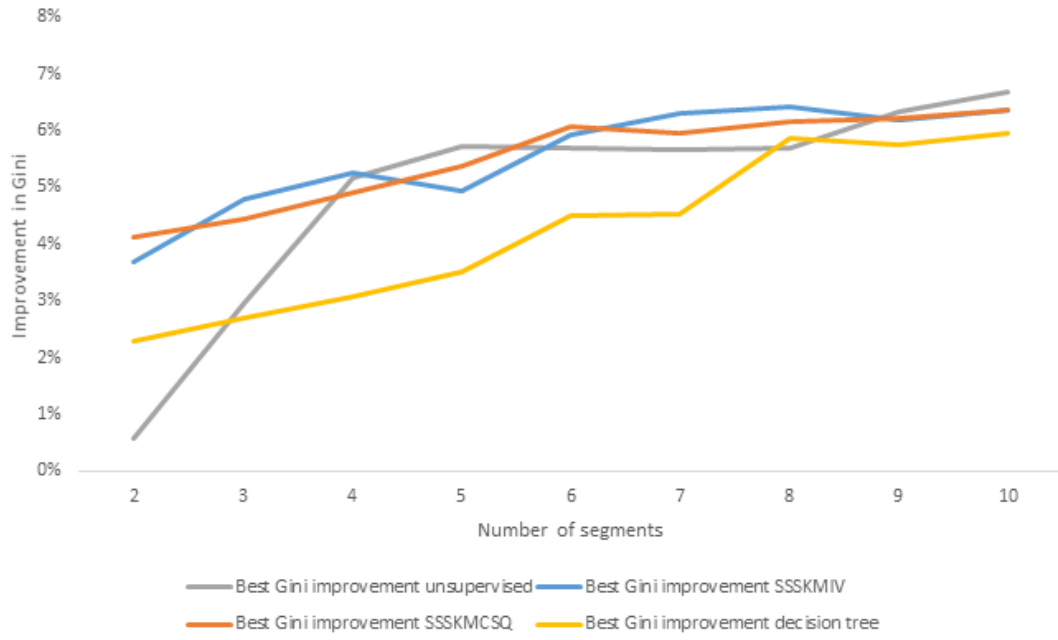


FIGURE 5.16: Comparison of best improvements in Gini

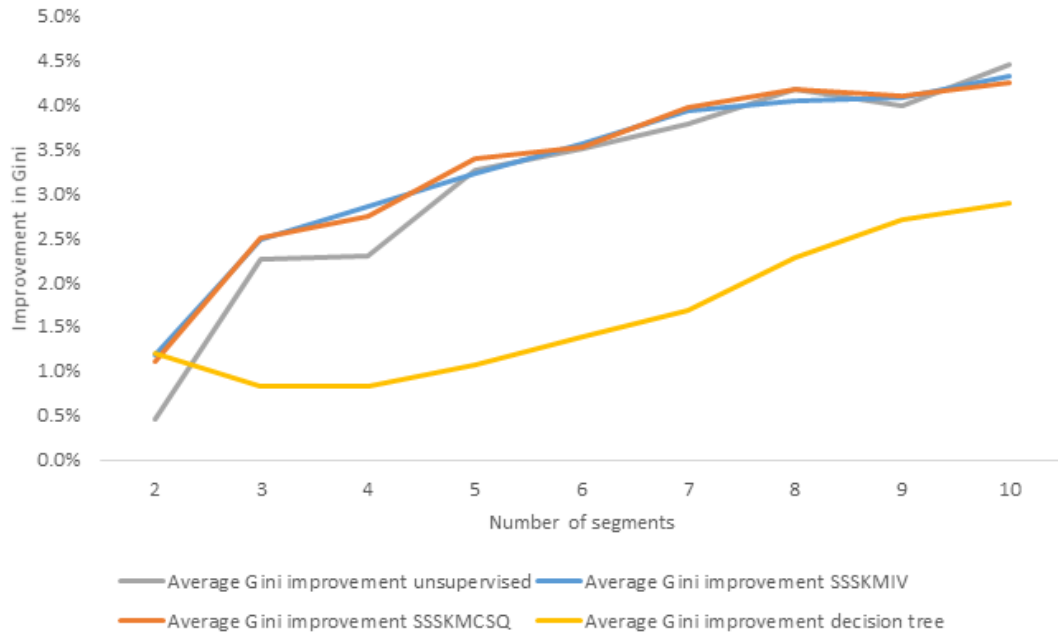


FIGURE 5.17: Comparison of average improvements in Gini

not big. Again, the SSSKMCSQ algorithm generally overrides more segment allocations compared to the SSSKMIV algorithm, and therefore produces slightly better CH values for higher values of w . Overfit is evident for SSSKMCSQ from $w = 0.6$ and for SSSKMIV from $w = 0.7$.

Surprisingly, it can be seen from Figure 5.19 and Table B.27 that the large gains in IV obtained by the SSSKM algorithms do not translate into any gain in average Gini for

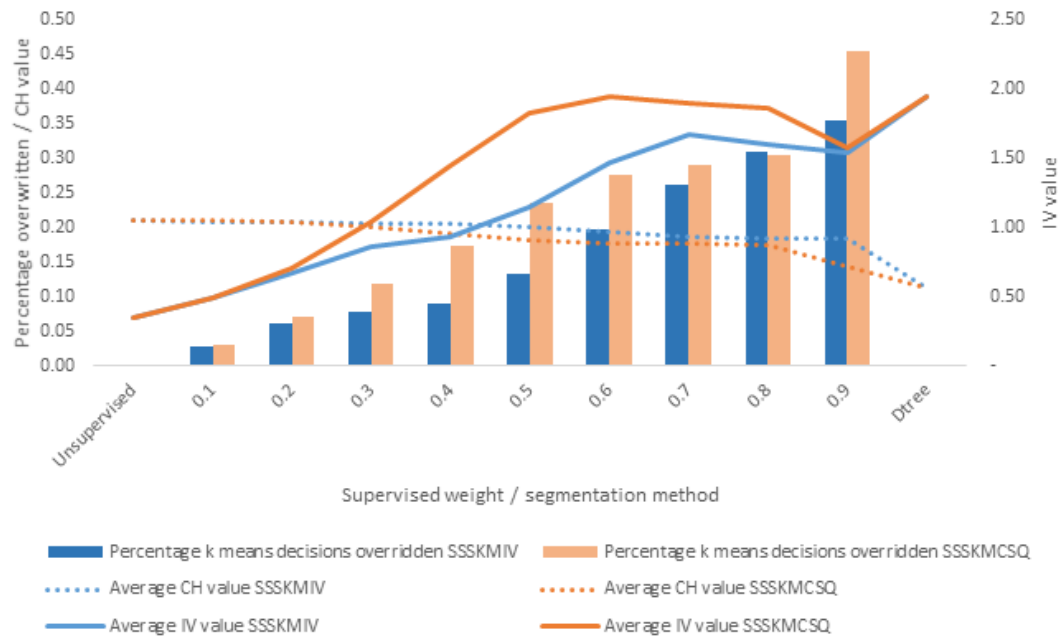


FIGURE 5.18: Comparison of segmentation characteristics by supervised weight

this data set. If anything, there is a slight downward trend in average Gini, whilst the decision tree segmentation obtains the best average values for Gini.

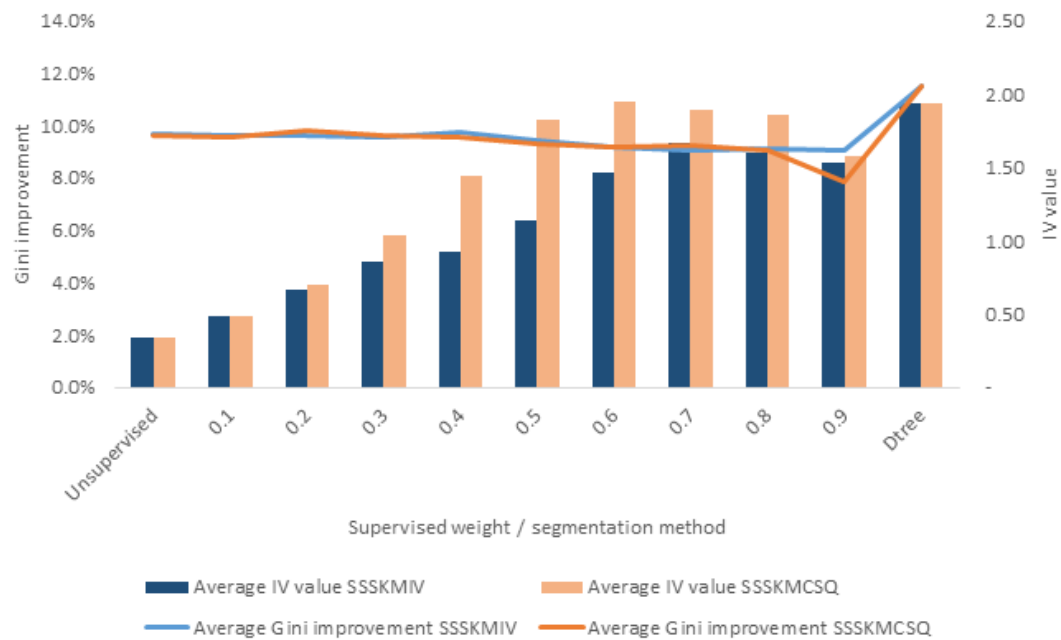


FIGURE 5.19: Success of target separation compared to Gini improvement by supervised weight

The stronger performance in Gini improvement by the decision tree is also evident from Figure 5.20 and Table B.28, however, it seems that both the unsupervised and SSSKM algorithms outperform decision trees (albeit with a very small margin) when fewer than

four segments are required. When considering average improvements (seen in Figure 5.21 and Table B.29), the decision tree consistently produces the best results.

The improvement in Gini of between 5% and 17% for this data set is noteworthy, especially since the reference validation Gini is already 71%.



FIGURE 5.20: Comparison of best improvements in Gini

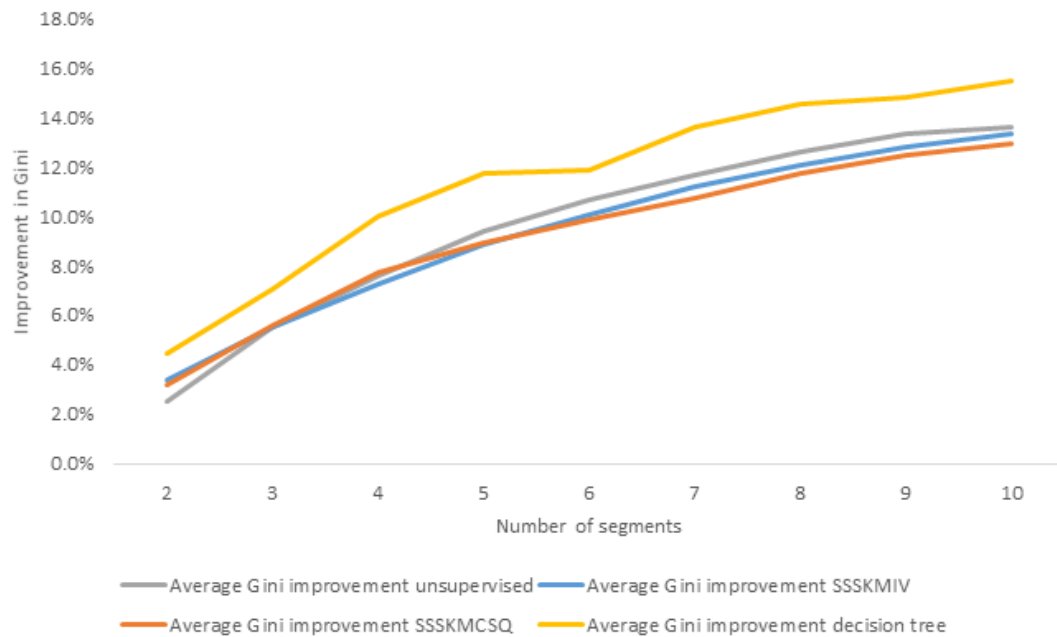


FIGURE 5.21: Comparison of average improvements in Gini

5.7.6 Insurance claim prediction data set results

Figure 5.22 and Table B.30 summarise the segment statistics for the last data set. We observe similar trends regarding overridden segment allocations between SSSKMIV and SSSKMCSQ compared to the previous data sets. Here, we see that overfit seems to start for lower values of w , somewhere between $w = 0.3$ to $w = 0.5$, and that the SSSKM algorithms are not very successful in improving IV over unsupervised segmentation. Decision tree segmentation do, however, succeed in obtaining high IVs.

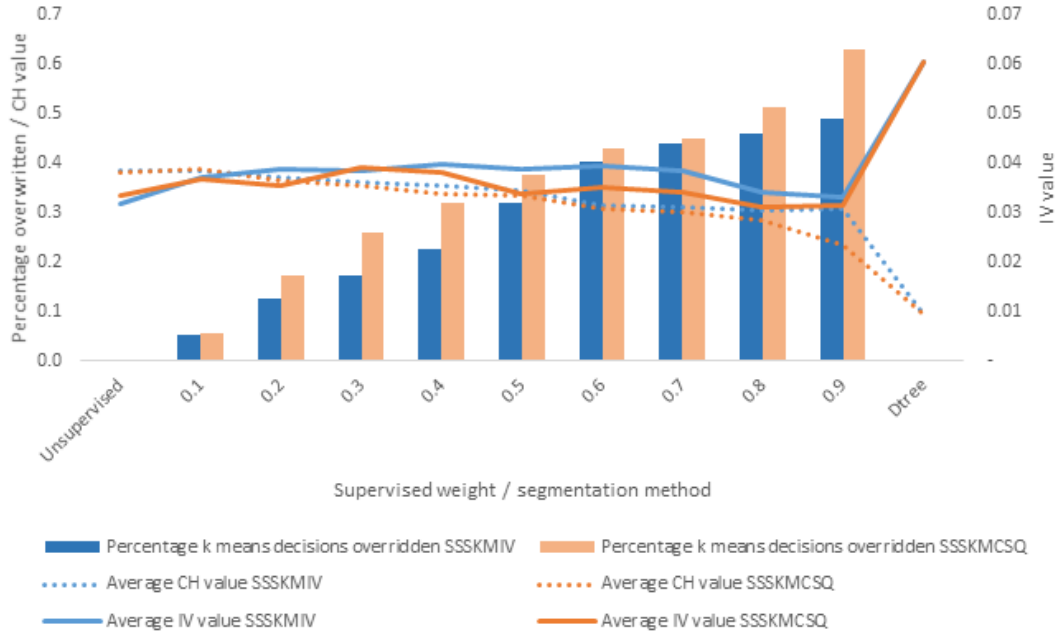


FIGURE 5.22: Comparison of segmentation characteristics by supervised weight

From Figure 5.23 and Table B.31, it can clearly be seen that, even though the SSSKM algorithms were not very successful in increasing IV by much, they are successful in improving the average Gini. SSSKMIV seems to perform the best based on average Gini, and we can also observe a correlation between IV and Gini improvement.

Figure 5.24 and Table B.32 confirm the results seen in Figure 5.23 and Table B.31 for this data set based on the best Gini per segment for each segmentation algorithm. Here, the SSSKMIV algorithm seems to have less of an edge over the SSSKMCSQ algorithm, but both clearly outperforms the decision tree.

The results are even clearer when analysing Figure 5.25 and Table B.33, which illustrate the average Ginis per segment for each algorithm. Interestingly, the decision tree actually reduces the Gini on average when two segments are selected.

The improvement of between 4% and 7% is achieved over the very low reference validation Gini of 8%. Clearly this target variable is not easy to predict well.

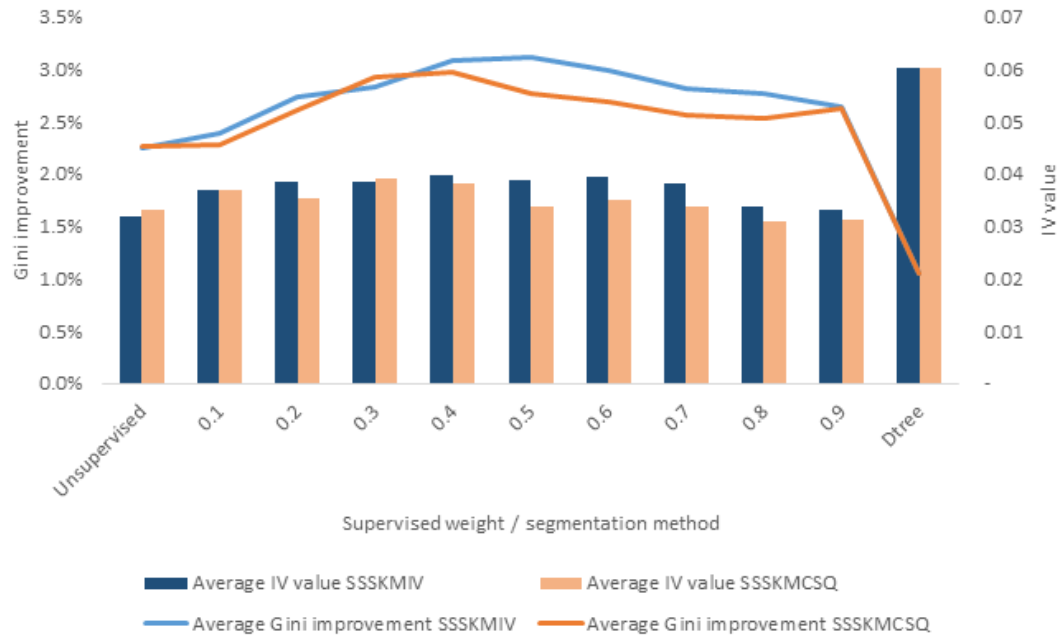


FIGURE 5.23: Success of target separation compared to Gini improvement by supervised weight

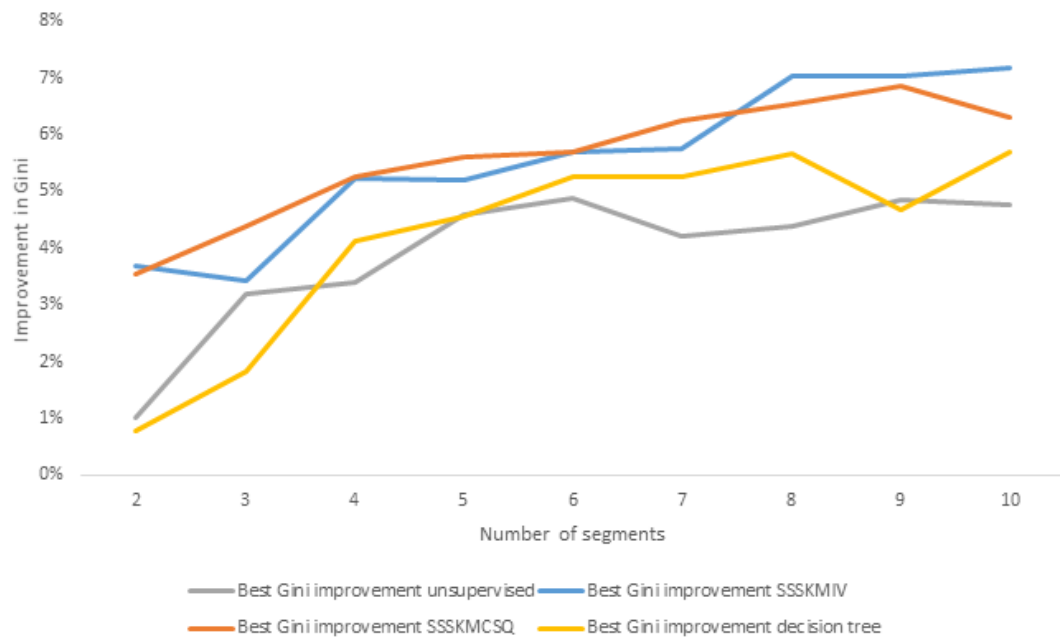


FIGURE 5.24: Comparison of best improvements in Gini

5.8 Summary of segmentation results

In Section 5.7 we compared the success of using the SSSKMIV and the SSSKMCSQ algorithms for segmentation to the results obtained by both supervised and unsupervised techniques currently used in industry applications. We have compared a number of

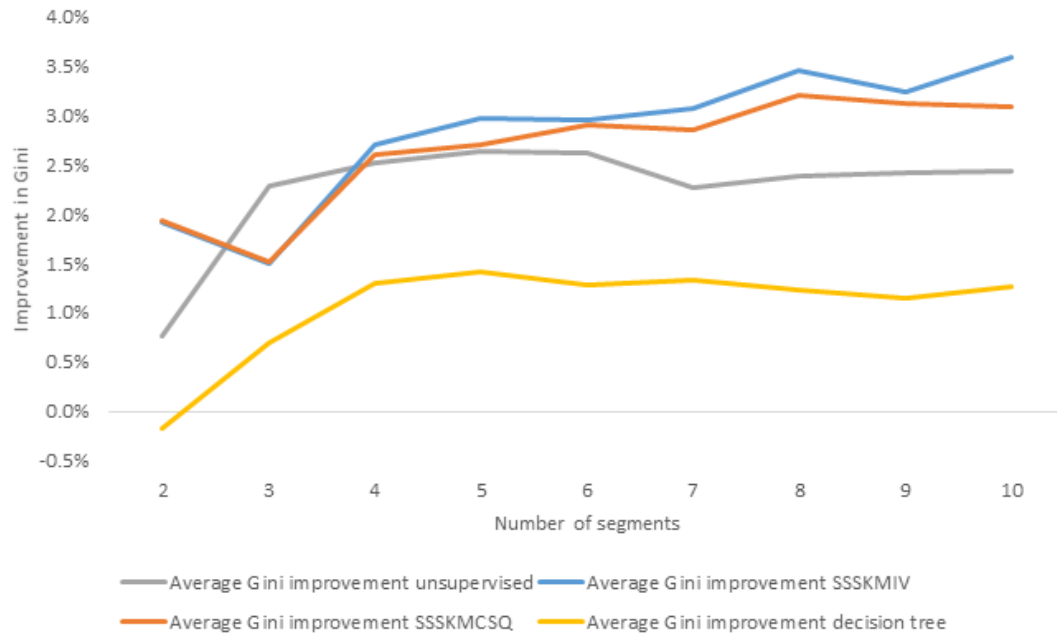


FIGURE 5.25: Comparison of average improvements in Gini

quantitative results and we have compared the success of the segmentation algorithms in improving predictive model accuracy as measured by the Gini coefficient.

Table 5.2 summarises the results obtained within this section.

Data set	Unsegmented Gini on validation set	Range of best improvements in Gini	Best suited methodology (in general)
Direct marketing	22%	8%-12%	Decision tree
Protein tertiary structures	59%	2%-11%	SSSKM
Credit applications	38%	23%-33%	Decision tree
Wine quality	61%	4%-6%	SSSKM
Chess (king-rook vs. king)	71%	5%-17%	Decision tree
Insurance claim prediction	8%	4%-7%	SSSKM

TABLE 5.2: Summary of results

From the results it can be seen that there is not one single segmentation algorithm that always outperforms when segmenting for predictive modelling. Depending on the characteristics of the data set, one method may outperform another. We have also seen that it is not always the case that a single methodology will always outperform other methodologies on a specific data set, but that the results may differ depending on the number of segments required.

From the six data sets analysed, we have seen a number of common trends regarding the SSSKM algorithms. We have seen that, as can be expected, the percentage of overridden segment assignments tend to increase as the value of w increases. In most cases, the increase in the value of w also coincides with an increase in the separation of the target variable (as measured by the IV in this study). However, for high values of w , some degree of overfit can be observed as is evident from a decrease in the IV when segmentation is applied to a validation set. We have seen that, in general, the SSSKMCSQ algorithm tends to override more segment assignments for lower values of w , which, in some cases, causes it to achieve higher target separation earlier. However, we have also found that the SSSKMCSQ algorithm generally creates segments with lower CH values when compared to the SSSKMIV algorithm.

We have also observed that in many cases the improvement in target rate separation on the validation set is correlated with the Gini improvement achieved, indicating that this measure may be used as an indication of the best value of w without the need to develop models.

Table 5.3 compares the overall average (across all values for K) Gini improvement of the SSSKMIV and SSSKMCSQ algorithms. The tendency of the SSSKMCSQ algorithm to override k-means assignments quicker is clear from the results as it can be seen that the algorithm (on average) obtains its highest Gini values for lower values of w compared to the SSSKMIV algorithm. From the results it is evident that this cannot be seen as a positive trait, as the SSSKMIV algorithm outperforms the SSSKMCSQ algorithm in every set (as measured by the average improvement in Gini on the validation set). The reason for this could be linked to overfit as well as the lower CH values of the SSSKMCSQ algorithm. The fact that the algorithm overfits, may cause it to miss important interactions in the data that is potentially identified by the SSSKMIV algorithm. This is also potentially the reason why the algorithm obtains lower CH values, it is not able to identify segments that are very similar from an explanatory variable perspective, which may mean the the models that are fitted performs worse on average.

Although we have observed the general trends described above in many cases, we have also found quite a number of exceptions. We have seen cases where the override percentage increases sharply along with an increase in the value of w and then subsequently drop off by a lot. We have also seen cases where the SSSKM algorithms are very successful in improving target separation but utterly fails in improving Gini.

In summary we can conclude that the SSSKM algorithms, although not always outperforming unsupervised and supervised techniques, can be a valuable tool to improve segmentation for predictive modelling, and do in many cases provide better segmentation than the more traditional segmentation methodologies.

Data set	Average Gini improvement SSS-KMIV	Average Gini improvement SSSKM-CSQ	Best value of w for SSS-KMIV	Best value of w for SSSKM-CSQ	Best SSSKM algorithm
Direct marketing	-0.02%	-0.05%	0.4	0.1	SSSKMIV
Protein tertiary structures	4.78%	4.69%	0.6	0.5	SSSKMIV
Credit applications	21.52%	21.51%	0.7	0.7	SSSKMIV
Wine quality	3.28%	3.28%	0.5	0.3	SSSKMIV
Chess (king-rook vs. king)	9.42%	9.29%	0.4	0.2	SSSKMIV
Insurance claim prediction	2.76%	2.62%	0.5	0.4	SSSKMIV

TABLE 5.3: Comparison of average SSSKM improvements in Gini between data sets

The reason why one segmentation technique outperforms another on a specific data set, but not on another, is the logical next question that can be asked. This has to be related to the characteristics and interactions of the variables considered and how the segmentation technique used either compliment or undermine these characteristics. We will explore these reasons further in Chapter 6.

5.9 Alternative modelling methodologies

As illustrated in Section 5.7 and summarised in Section 5.8, regression models can benefit from segmentation. However, it is well known that quite a number of modelling methodologies do not require the same linearity assumptions as regression techniques (also see Section 2.4). In Section 2.4 it is discussed why these nonlinear techniques are not very popular in practical applications, but for the sake of completeness, we will compare the result of some of the well known non-linear techniques on the data sets introduced in this chapter.

We will compare the results obtained on the data sets from Section 5.7 (based on segmented regression models) with results obtained from the following, non-linear modelling techniques:

- Neural networks.
- Support vector machines.

- K-nearest-neighbour models (or memory based reasoning).
- Decision trees (used here as the final model, not as segmentation).
- Gradient boosting (which is a boosting variation of random forests).

The model results are obtained through SAS[®] Enterprise Miner[™] software, and uses modelling nodes made available by the software. Below is a description of each node used:

- AutoNeural tool.
 - Neural networks have been researched since the early 1940’s (McCulloch and Pitts, 1943) and are very well known in the supervised classification field today. The AutoNeural tool is provided by the SAS[®] Enterprise Miner[™] software to automatically configure a neural network and conducts searches to optimise the settings of a neural network for the specific data set used. Given that a neural network has so many variables to consider, and that neural networks are not the focus of this study, this tool was deemed appropriate for our purposes. Table B.34 provides an overview of the modelling properties that can be set for the AutoNeural tool. The values that were used are also provided. For this node, the default input values proposed by Enterprise Miner[™] was used.
- Support vector machine (SVM) tool.
 - Support vector machines are relatively new (compared to other well-known modelling techniques) and have been introduced in the nineties (Cortes and Vapnik, 1995). It has been researched quite extensively over the last number of years (Cristianini and Shawe-Taylor, 2000, Li, 2015, Meyer and Wien, 2014). SVM’s do not produce probabilities of classification, but assigns observations to a binary class by maximising the margin between classes through hyperplanes (Cortes and Vapnik, 1995). The Enterprise Miner[™] SVM tool facilitates the development of support vector machines for supervised binary classification. Table B.35 summarises the modelling properties and the values that were used for developing SVM models for this study. Only one change was made to the default Enterprise Miner[™] settings, the “Estimation Method” was set to “Least Squares SVM” as opposed to “Decomposed Quadratic Programming” as the DQP setting failed to find a conclusive result on the claim prediction data set. Note that the estimation method is printed in **bold** in the table to indicate the change.

- The memory based reasoning node.
 - The MBR (memory based reasoning) node uses k-nearest-neighbour principles to classify observations in a data set. When a new observation is evaluated, the algorithm allows the k nearest observations of the development set to "vote" regarding that observation's classification (their votes are based on the values of their target variables). The votes then represent the probabilities of the new observation belonging to that specific target value. The MBR node is also used here with the default values provided by the SAS[®] Enterprise Miner[™] software. See Table B.36 for details of the properties and the values used for the MBR node.
- The decision tree node.
 - Decision trees, as discussed in Section 2.7.1, are simple classifiers that produce results that are easy to interpret and apply. For this reason they are also quite popular in the industry (Anderson, 2007). We used decision trees to segment data for predictive modelling in Section 5.7. Here the SAS[®] Enterprise Miner[™] node will use the decision tree for predictive modelling. Two changes were made to the default settings in Enterprise Miner[™]. The splitting criterion for nominal input variables was changed from chi-squared probability to Gini, as the Gini is the measure we use for model success / accuracy in this study. In addition, the number of branches or subsets that a splitting rule can produce was increased from two (binary) to six, which is equal to the default value for the depth of a decision tree. This was done as it allows the decision tree to segment far more granularly, which allows it to be more competitive with the other non-linear techniques used that allow very granular results. See Table B.37 for details of all the properties and the values used. Note that the properties that were changed as described here are indicated in **bold**.
- The gradient boosting node.
 - The gradient boosting node in the SAS[®] Enterprise Miner[™] software draws its concept from the greedy decision tree approach proposed by Friedman (2001). The algorithm creates a number of small decision trees on the development set, and these trees are combined to produce the model's output. The technique can be linked to the techniques used in random forests (Breiman, 2001) in that a number of different trees are developed. It differs from random forests in that the subsequent trees are fitted on the residual of the prediction of the earlier trees, rather than being randomly refitted to re-optimize

the target variable. It is therefore more similar to boosting algorithms, and less similar to bagging, as is the case with pure random forest techniques (Friedman et al., 2001) as described in Section 2.4.6. Similar to the other nodes, the gradient boosting node is used with its default setting in Enterprise MinerTM. See Table B.38 for the details of all the properties and the values used.

Note that the descriptions of the properties as described in Tables B.34 to B.38 were taken directly from the information provided by the SAS[®] Enterprise MinerTM software in their help files ³.

In Sections 5.7 and 5.8, the graphs and tables used focused on the improvement in Gini coefficient that was obtained through segmentation. In this section, the best Ginis obtained in Section 5.7 will be compared to the Ginis obtained from the alternative techniques. In order to link the improvements shown in Section 5.7 to the results shown in the next sections, Tables B.39 to B.44 are provided. The tables effectively add the maximum Gini improvements shown in Tables B.12, B.16, B.20, B.24, B.28 and B.32 to the unsegmented Ginis obtained on the specific data sets, and therefore shows the best Ginis obtained for each segmentation technique (unsupervised k-means, SSSKMIV, SSSKMCSQ and decision trees) and each number of segments (two to ten). The overall best Gini obtained for each technique is printed in **bold**. These will effectively be the Ginis that will be compared to the Ginis obtained from the alternative techniques.

5.9.1 Direct marketing data set results

Table 5.4 summarises the performance of the predictive modelling methodologies discussed in Section 5.9 when applied to the direct marketing data set (as measured by the Gini coefficient). The gradient boosting methodology, which is similar to random forests, achieved the best result on this data set, with decision tree segmentation running a close second. Neural networks could not converge to a model without overfitting, and the resulting Gini on the validation set is therefore effectively equal to 0. What can be seen additionally from Table 5.4 is that segmentation-based techniques take in positions two through to five as ranked by the Gini coefficient on the validation set.

Table 5.5 will be shown for all data sets and focuses on the relative performance of the segmentation-based techniques (when compared to the alternative predictive modelling methodologies analysed). It provides the rank of each technique again (compared to the

³Copyright ©SAS Institute Inc., Cary, NC, USA. All Rights Reserved. Reproduced with permission of SAS Institute Inc., Cary, NC.

Modelling technique	Best Gini obtained	Rank
Unsupervised segmentation (k-means) with logistic regression	27.11%	5
SSSKMIV segmentation with logistic regression	27.89%	4
SSSKMCSQ segmentation with logistic regression	29.14%	3
Decision tree segmentation with logistic regression	33.70%	2
Neural networks	0%	9
Support vector machines	24.46%	6
Memory based reasoning (KNN)	21.95%	8
Decision trees	22.94%	7
Gradient boosting (random forests)	35.31%	1

TABLE 5.4: Comparison of performance of alternative modelling methodologies to segmentation with logistic regression for the direct marketing data set

alternative methodologies shown in Table 5.4), and then considers the results obtained for each number of segments in Section 5.7 for this data set. In many practical instances where segmentation is used for predictive modelling, the number of segments that is used needs to strike a balance between accuracy and complexity (Andersen, 2008, Thomas, 2009). The more segments that are used, the more models need to be developed and maintained. For this reason, Table 5.5 indicates what the minimum number of segments are that will ensure that each technique retains its indicated rank.

To clarify this further, consider decision tree segmentation with logistic regression in Table 5.5. With a Gini of 33.7% on the validation set, this methodology obtains the second-highest Gini of all the methodologies considered. In order to obtain 33.7% Gini, the data set was segmented into ten different segments (as indicated in the fourth column of Table 5.5). However, by using only two segments, the methodology already achieves a Gini of 30.34%, which allows it to keep its second-ranked position. In short therefore, the decision tree segmentation methodology can achieve the second-best Gini of all techniques with only two segments. This means that the model developer will need to weigh up the additional complexity of implementing and maintaining a random forest type model, with the slight reduction in Gini obtained from two simple logistic regression models.

5.9.2 Protein tertiary structures data set results

Table 5.6 summarises the Gini results of the various methodologies as applied to the data set on protein tertiary structures. As evidenced by the table, the ranking order of the methodologies is completely different compared to the order seen in Section 5.9.1.

Modelling technique	Best Gini obtained	Rank	Best nr of segments	Minimum nr segments to retain position	Minimum Gini that retains position
Unsupervised segmentation (k-means) with logistic regression	27.11%	5	9	2	25.72%
SSSKMIV segmentation with logistic regression	27.89%	4	9	9	27.89%
SSSKMCSQ segmentation with logistic regression	29.14%	3	9	9	29.14%
Decision tree segmentation with logistic regression	33.7%	2	10	2	30.34%

TABLE 5.5: Further detail on segmentation performance compared to alternative techniques for the direct marketing data set

As a start, the gradient boosting methodology ranks third from the bottom, at number seven. The methodology that achieves the best results in this case is the memory based reasoning node, which uses k nearest neighbour logic to assign target values to observations. In Section 5.9.1, this methodology was ranked at position eight.

Modelling technique	Best Gini obtained	Rank
Unsupervised segmentation (k-means) with logistic regression	66.88%	5
SSSKMIV segmentation with logistic regression	70.37%	2
SSSKMCSQ segmentation with logistic regression	67.99%	4
Decision tree segmentation with logistic regression	66.43%	6
Neural networks	47.32%	9
Support vector machines	57.04%	8
Memory based reasoning (KNN)	80.33%	1
Decision trees	69.17%	3
Gradient boosting (random forests)	57.89%	7

TABLE 5.6: Comparison of performance of alternative modelling methodologies to segmentation with logistic regression for the protein tertiary structure data set

The best-ranked segmentation-based methodology for this data set is SSSKMIV in position two. This can also be seen in Table 5.7. From this table it can also be seen that it will require eight segments for SSSKMIV to retain its second position.

A literature review revealed that a similar data set to this was used by Rana et al. (2014), albeit with a slightly different modelling approach. They attempted to predict

the nominal value of the (amongst other things) root mean square error using techniques such as random forests. Their results are therefore not directly comparable to our results (which targets a binary variable), but they achieved an R-squared value of 84% on the RMSE target.

Modelling technique	Best Gini obtained	Rank	Best nr of segments	Minimum nr segments to retain position	Minimum Gini that retains position
Unsupervised segmentation (k-means) with logistic regression	66.88%	5	10	9	66.66%
SSSKMIV segmentation with logistic regression	70.37%	2	8	8	70.37%
SSSKMCSQ segmentation with logistic regression	67.99%	4	10	9	67.03%
Decision tree segmentation with logistic regression	66.43%	6	10	2	60.62%

TABLE 5.7: Further detail on segmentation performance compared to alternative techniques for the protein tertiary structure data set

5.9.3 Credit application data set results

Table 5.8 shows that, for the data set with credit application data, neural networks outperform all other methodologies. In Sections 5.9.1 and 5.9.2, neural networks were ranked last each time. In both cases, the neural network struggled to find a solution that does not overfit. However, in this case the structure of the data set evidently suited the methodology well.

Similar to what was seen in Section 5.9.1, segmentation-based techniques take up positions two to five for this data set, with decision tree segmentation performing best. At this point a trend is emerging that segmentation-based techniques may not always render the best results, but seem to deliver results that are consistently amongst the top.

Table 5.9 shows that the decision tree segmentation methodology used can retain its position with only four segments. However, given that positions two to five are taken by other segmentation-based techniques, decision tree segmentation can outperform all remaining non-segmentation-based methodologies (i.e. all non-segmentation-based techniques other than neural networks) with only two segments (see Table B.20). Again, the

Modelling technique	Best Gini obtained	Rank
Unsupervised segmentation (k-means) with logistic regression	63.11%	5
SSSKMIV segmentation with logistic regression	66.25%	3
SSSKMCSQ segmentation with logistic regression	65.39%	4
Decision tree segmentation with logistic regression	70.89%	2
Neural networks	72.20%	1
Support vector machines	31.47%	9
Memory based reasoning (KNN)	43.80%	8
Decision trees	48.41%	7
Gradient boosting (random forests)	53.96%	6

TABLE 5.8: Comparison of performance of alternative modelling methodologies to segmentation with logistic regression for the credit scoring data set

model developer will need to take this into consideration when selecting the best suited methodology during model development.

Modelling technique	Best Gini obtained	Rank	Best nr of segments	Minimum nr segments to retain position	Minimum Gini that retains position
Unsupervised segmentation (k-means) with logistic regression	63.11%	5	9	2	60.47%
SSSKMIV segmentation with logistic regression	66.25%	3	9	9	66.25%
SSSKMCSQ segmentation with logistic regression	65.39%	4	9	7	64.9%
Decision tree segmentation with logistic regression	70.89%	2	10	4	70.17%

TABLE 5.9: Further detail on segmentation performance compared to alternative techniques for the credit scoring data set

As mentioned before, this data set was used in an online competition on the Kaggle website (*Kaggle Website*, 2016) where hundreds of teams competed to find the highest predictive power. The top performers all used very sophisticated, hybrid models which did not only use a single methodology, but combined various different models to obtain the final prediction (Sharma, 2013). The accuracy was measured by the AUC statistic on a test data set of which the target variables were not made available. The winning team's AUC on the test set was 0.869558 (*Kaggle Website*, 2016, Sharma, 2013), which can be transformed (see Hand and Till (2001)) to a Gini of 73.9%. This is only slightly

higher than the Gini obtained by neural networks in Table 5.8, and it must be noted that the Gini in Table 5.8 is calculated on a different validation data set, since the target variable was not made available for the data set on which the competitors of the competition was evaluated.

5.9.4 Wine quality data set results

For the wine quality data set, Table 5.10 shows that segmentation-based techniques occupy the top three positions, with decision tree segmentation in position five. Results are generally very close, with only decision trees and support vector machines not doing particularly well.

Modelling technique	Best Gini obtained	Rank
Unsupervised segmentation (k-means) with logistic regression	67.21%	1
SSSKMIV segmentation with logistic regression	66.97%	2
SSSKMCSQ segmentation with logistic regression	66.92%	3
Decision tree segmentation with logistic regression	66.50%	5
Neural networks	66.64%	4
Support vector machines	59.66%	9
Memory based reasoning (KNN)	66.10%	6
Decision trees	60.86%	8
Gradient boosting (random forests)	63.34%	7

TABLE 5.10: Comparison of performance of alternative modelling methodologies to segmentation with logistic regression for the wine quality data set

From Table 5.11 it can be seen that, although the segmentation-based methodologies perform best on this data set, many segments are required to retain these positions given how close the results are. The SSSKMIV methodology can achieve a Gini higher than 65% with only three segments, however, which many may believe is close enough given the relative simplicity.

Quite a number of examples of studies using this, or similar data sets, can be found in literature (Cortez et al., 2009, Horák, 2009, Nachev and Hogan, 2013), but all the studies that could be found predict the exact expert score (i.e. 1 - 10), and not a binary target as was done here, which therefore makes meaningful comparison with our results impossible. Cortez et al. (2009) applied multiple regression, neural networks and SVM's to the data set and found the SVM to perform best. SVM's also outperformed a number of neural networks in the study performed by Nachev and Hogan (2013), whilst Horák (2009) limited their study to regression trees. SVM's did not perform well in our

Modelling technique	Best Gini obtained	Rank	Best nr of segments	Minimum nr segments to retain position	Minimum Gini that retains position
Unsupervised segmentation (k-means) with logistic regression	67.21%	1	10	10	67.21%
SSSKMIV segmentation with logistic regression	66.97%	2	8	8	66.97%
SSSKMCSQ segmentation with logistic regression	66.92%	3	10	8	66.7%
Decision tree segmentation with logistic regression	66.5%	5	10	8	66.41%

TABLE 5.11: Further detail on segmentation performance compared to alternative techniques for the wine quality data set

analysis, as can be seen from Table 5.10, but the target variable as well as the way in which success is measured (i.e. by the Gini as opposed to accuracy of multiple class predictions) in our study differs vastly from the other studies and therefore not much can be read into these differences.

5.9.5 Chess (king-rook vs. king) data set results

Table 5.12 shows that decision tree logic is best suited for the non-linear nature of the chess (king-rook vs. king) data set. This is the first data set where segmentation based techniques fail to be one of the top 2 methodologies, with decision tree segmentation in third place.

From Table 5.13, it can be seen that the structure and properties of the chess (king-rook vs. king) data set requires many segments in order for the segmentation-based techniques to achieve good results. For this data set the segmentation-based techniques are an unlikely choice as the decision tree model achieves the best results whilst being relatively simple to implement, maintain and understand.

This data set was used by Bain (1994) who used an inductive learning algorithm called GCWS to predict the number of moves directly, again targeting a different target variable than we do. He shows that the algorithm can successfully identify the number of moves for a subset of the data base, but does not show his results in a format that makes it comparable to this study. Quinlan and Cameron-Jones (1995) applies another inductive learning algorithm called FOIL to the data set, but also attempts to predict the exact

Modelling technique	Best Gini obtained	Rank
Unsupervised segmentation (k-means) with logistic regression	86.95%	5
SSSKMIV segmentation with logistic regression	86.60%	6
SSSKMCSQ segmentation with logistic regression	86.49%	7
Decision tree segmentation with logistic regression	88.34%	3
Neural networks	25.47%	9
Support vector machines	74.81%	8
Memory based reasoning (KNN)	90.63%	2
Decision trees	93.34%	1
Gradient boosting (random forests)	87.25%	4

TABLE 5.12: Comparison of performance of alternative modelling methodologies to segmentation with logistic regression for the chess (king-rook vs. king) data set

Modelling technique	Best Gini obtained	Rank	Best nr of segments	Minimum nr segments to retain position	Minimum Gini that retains position
Unsupervised segmentation (k-means) with logistic regression	86.95%	5	10	10	86.95%
SSSKMIV segmentation with logistic regression	86.6%	6	10	10	86.6%
SSSKMCSQ segmentation with logistic regression	86.49%	7	10	2	76.56%
Decision tree segmentation with logistic regression	88.34%	3	10	9	88.3%

TABLE 5.13: Further detail on segmentation performance compared to alternative techniques for the chess (king-rook vs. king) data set

number of moves. Again the results are not in a format that can be directly compared to this study.

5.9.6 Insurance claim prediction data set results

Table 5.14 shows the result on the last data set to be analysed, the claim prediction data set. It can be seen from the table that the first three positions are again held by segmentation-based techniques, with SSSKMIV achieving best results. The best non-segmentation-based technique is gradient boosting in position four followed by unsupervised k-means segmentation.

The Ginis for this application are low, so the relative differences between the 15.18% obtained by SSSKMIV is quite high when compared to the 12.92% of gradient boosting.

Modelling technique	Best Gini obtained	Rank
Unsupervised segmentation (k-means) with logistic regression	12.92%	5
SSSKMIV segmentation with logistic regression	15.19%	1
SSSKMCSQ segmentation with logistic regression	14.89%	2
Decision tree segmentation with logistic regression	13.72%	3
Neural networks	10.22%	6
Support vector machines	10.06%	7
Memory based reasoning (KNN)	9.39%	8
Decision trees	8.69%	9
Gradient boosting (random forests)	12.92%	4

TABLE 5.14: Comparison of performance of alternative modelling methodologies to segmentation with logistic regression for the claim prediction data set

Table 5.15 shows that SSKMIV can retain its first position with eight segments. Note that SSSKMCSQ is in second position, and SSSKMIV achieves a Gini of 13.27% with only four segments (see Table B.32), which is better than any of the non-segmentation-based techniques. The model developer will need to consider whether the additional complexity is worth the additional Gini improvement.

Modelling technique	Best Gini obtained	Rank	Best nr of segments	Minimum nr seg-ments to retain position	Minimum Gini that retains position
Unsupervised segmentation (k-means) with logistic regression	12.92%	5	6	3	11.23%
SSSKMIV segmentation with logistic regression	15.19%	1	10	8	15.07%
SSSKMCSQ segmentation with logistic regression	14.89%	2	9	6	13.73%
Decision tree segmentation with logistic regression	13.72%	3	10	6	13.29%

TABLE 5.15: Further detail on segmentation performance compared to alternative techniques for the claim prediction data set

This data set was used in another Kaggle (*Kaggle Website*, 2016) competition, with the winner achieving a Gini of just over 20%. Again, the test data set's target variables

are not available, so the comparison is not completely like-for-like, but a Gini of 15.07% (as achieved by SSSKMIV) would have achieved an overall tenth position out of 102 participants. The techniques applied by the winner could not be found on the web site or in any papers, but Dal Pozzolo et al. (2010) did a full dissertation where they compared the results of many techniques (decision tree, random forest, naive bayes, k-nearest neighbour models, neural networks, SVM's and linear discriminant analysis) on the data, obtaining at best a Gini of 11.5% on the training data set.

5.9.7 Summary of alternative methodology results

Comparative studies regarding the success of the modelling methodologies analysed in this Section are being performed continuously and on a wide variety of fields (Baba et al., 2015, Fernández et al., 2015, Maroco et al., 2011, Rodriguez-Galiano et al., 2015, Stylianou et al., 2015, Tao et al., 2015, Were et al., 2015). Although it is not the focus of this study to do an exhaustive comparison of modelling methodologies, this section provides an overview of how some of the more popular supervised classification methodologies perform when compared to segmented logistic regression. The analysis was performed in the SAS® Enterprise Miner™ software and, for the most part, the default settings as provided by the software was used for the modelling nodes.

Perhaps due to the diverse nature of the data sets used in this study, it was interesting to see that no single methodology dominated the top position. What was also clear, was that the four segmentation-based methodologies explored always performed well, even though other methodologies demonstrated some large inconsistencies. Table 5.16 summarises the best performing methodologies for each data set. In addition, the table also shows the position, or rank, of the best performing segmentation-based methodology. The consistency is clear from the fact that these methodologies usually take in either position one or two, with only a single third place.

Table 5.17 provides a different view on the consistency of the segmentation-based methodologies. For each of the data sets, the rank of each methodology is shown. The last column provides the average rank of each methodology. The table has been sorted from lowest average rank to highest average rank. As expected, the segmentation-based techniques do very well, taking in the first four positions. SSSKMIV is rated first with an average rank of three.

We have also seen that there are many examples where segmentation-based methodologies can achieve acceptable results with relatively low number of segments. In these cases, the reduced complexity of segmented logistic regression models can be seen as an advantage.

Data set	Best methodology	Position of best segmentation-based methodology
Direct marketing	Gradient boosting (random forests)	2
Protein tertiary structures	Memory based reasoning (KNN)	2
Credit application data	Neural networks	2
Wine quality	Unsupervised segmentation (k-means) with logistic regression	1
Chess (king-rook vs. king)	Decision trees	3
Insurance claim prediction	SSSKMIV segmentation with logistic regression	1

TABLE 5.16: Summary of results of alternative techniques compared to segmentation with logistic regression

5.10 Conclusion

In this chapter we focused on the actual purpose of the semi-supervised segmentation algorithm, which is segmenting of data sources into groups on which separate predictive models are developed. We compared the success of using the SSSKMIV and the SSSKMCSQ algorithms for segmentation to the results obtained by both supervised and unsupervised techniques currently used in industry applications. We then also compared the results obtained from the segmentation-based techniques to popular non-linear classification techniques to understand how well these compare in terms of performance.

In this chapter we contributed the following:

- We applied the proposed SSSKMIV and SSSKMCSQ algorithms to six data sets from various disciplines.
- We compared the success of segmenting by means of the SSSKMIV and SSSKMCSQ methodologies to segmentation, using supervised segmentation with decision trees and unsupervised segmentation with standard k-means. Success was measured based on the improvement in the validation set's Gini coefficient, when models are separately developed on the proposed segments.
- We analysed and compared the results and pointed out observed trends of the SSSKMIV and SSSKMCSQ algorithms.

Modelling technique	Rank: Direct marketing	Rank: Protein tertiary structures	Rank: Credit scoring	Rank: Wine quality	Rank: Chess end game	Rank: Claim prediction	Average rank
SSSKMIV segmentation with logistic regression	4	2	3	2	6	1	3
Decision tree segmentation with logistic regression	2	6	2	5	3	3	3.5
SSSKMCSQ segmentation with logistic regression	3	4	4	3	7	2	3.8
Unsupervised segmentation (k-means) with logistic regression	5	5	5	1	5	5	4.3
Gradient boosting (random forests)	1	7	6	7	4	4	4.8
Memory based reasoning (KNN)	8	1	8	6	2	8	5.5
Decision trees	7	3	7	8	1	9	5.8
Neural networks	9	9	1	4	9	6	6.3
Support vector machines	6	8	9	9	8	7	7.8

TABLE 5.17: Summary of ranking position of alternative methodologies

- We then applied popular alternative, non-linear classification techniques to the 6 data sets and compared the performance of these techniques to the segmentation-based techniques.
- We indicated observed trends of the segmentation-based techniques' performance compared to popular non-linear techniques.

Chapter 6

Segmentation performance on simulated data sets

6.1 Introduction

Chapter 5 compares the success of using the SSSKMIV and the SSSKMCSQ algorithms for segmentation to the results obtained by both supervised and unsupervised techniques currently used in industry applications. It also compares the results obtained from the segmentation-based techniques to popular non-linear classification techniques, to understand how well these compare in terms of performance.

In this chapter we continue to explore the phenomenon observed in Chapter 5, where one segmentation technique outperforms another, depending on the selected data set used. In order to do so, we simulate data with predefined characteristics. This makes it possible to understand what characteristics cause one methodology to outperform another. It must be noted that we do not endeavour to simulate all possible data characteristics here (indeed that would be impossible), but simply some of the more obvious ones that may explain the phenomenon observed.

6.1.1 Research contributions

In this chapter we contribute the following:

- We test the impact of segmentation for predictive modelling on a base-case scenario, where a data set will be simulated for the purposes of segmentation for predictive modelling. We analyse the impact of the segmentation techniques introduced in Chapter 5 on the simulated base-case data set (using the techniques

applied in Chapter 5), and measure how successful these techniques are in comparison to the true (actual) Gini obtained from the simulation.

- We then expand on the base-case data set by generating scenarios of data sets. This is done through random sampling of data set characteristics such as segment-event rate, segment-specific explanatory variable distribution and segment complexity. The outcome is a large number of data sets with different characteristics that can be investigated to understand what specific characteristics cause one segmentation technique to outperform another. Representative samples of data sets with known characteristics are tested.
- We apply the segmentation techniques discussed in Chapter 5 to the selected simulated data sets (with known characteristics) and analyse what characteristics cause one segmentation technique to outperform another.
- We consider how underlying characteristics (like the true IV) influence what value of w performs well during SSSKMIV segmentation.

6.2 Base case scenario

In order to facilitate a good platform to explain the data simulation experiment, we first establish a base case for simulating a data set, after which we discuss the additional elements that are varied for further exploration. The approach described here is similar to approaches of simulating data sets with binary outcomes followed by Shifa and Rashid (2003) and Venter and De la Rey (2007).

The goal of this base case scenario is to show that it is possible to simulate a data set on which segmentation for logistic regression modelling will have a positive impact on accuracy, compared to the case where no segmentation is done. We will simulate six segments, with six being chosen again since it offers enough diversity without becoming overly complex. Recall from Section 2.4.1 that, in instances where $\mathbf{y}$ is binary, the logistic regression model is given by Equation 2.2. From Section 2.4, $\hat{y}_i$ is defined as the probability of an event occurring given $\mathbf{x}_i$, such that

$$\log \left(\frac{\Pr(y_i = 1 \mid \mathbf{X} = \mathbf{x}_i)}{1 - \Pr(y_i = 1 \mid \mathbf{X} = \mathbf{x}_i)} \right) = a\beta_0 + \boldsymbol{\beta}^T \mathbf{x}_i. \quad (6.1)$$

In the base case scenario, we set $K = 6$, and for each of the six segments (S_j) we use different parameters for $\boldsymbol{\beta}$ (note that m is set to 20):

$$\begin{aligned}
S_1 : \boldsymbol{\beta} &= [1, 1, -0.5, 0.5, -1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0] \\
S_2 : \boldsymbol{\beta} &= [1, 1, -0.5, 0.5, 0, 0, -1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0] \\
S_3 : \boldsymbol{\beta} &= [1, 1, -0.5, 0.5, 0, 0, 0, 0, -1, 1, 0, 0, 0, 0, 0, 0, 0] \\
S_4 : \boldsymbol{\beta} &= [1, 1, -0.5, 0.5, 0, 0, 0, 0, 0, 0, -1, 1, 0, 0, 0, 0, 0] \\
S_5 : \boldsymbol{\beta} &= [1, 1, -0.5, 0.5, 0, 0, 0, 0, 0, 0, 0, 0, -1, 1, 0, 0, 0] \\
S_6 : \boldsymbol{\beta} &= [1, 1, -0.5, 0.5, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, -1, 1, 0].
\end{aligned} \tag{6.2}$$

In all cases, β_0 is set to 0.

For $\mathbf{X}$, all values, except for the values of $\mathbf{X}_3$ and $\mathbf{X}_4$ are drawn from $N(0, 1)$ distributions. In order to distinguish the segments, the values of $\mathbf{X}_3$ and $\mathbf{X}_4$ were drawn from separate distributions for each segment,

$$\begin{aligned}
S_1 : \mathbf{X}_3 &\in N(4, 1) \text{ and } \mathbf{X}_4 \in N(6, 2) \\
S_2 : \mathbf{X}_3 &\in N(10, 2) \text{ and } \mathbf{X}_4 \in N(12, 1) \\
S_3 : \mathbf{X}_3 &\in N(-4, 2) \text{ and } \mathbf{X}_4 \in N(-2, 1) \\
S_4 : \mathbf{X}_3 &\in N(-2, 1) \text{ and } \mathbf{X}_4 \in N(-4, 2) \\
S_5 : \mathbf{X}_3 &\in N(1, 5) \text{ and } \mathbf{X}_4 \in N(10, 5) \\
S_6 : \mathbf{X}_3 &\in N(5, 5) \text{ and } \mathbf{X}_4 \in N(5, 5).
\end{aligned} \tag{6.3}$$

The number of observations per segment were also varied,

$$\begin{aligned}
S_1 : 1000 \\
S_2 : 200 \\
S_3 : 500 \\
S_4 : 1000 \\
S_5 : 1000 \\
S_6 : 500.
\end{aligned} \tag{6.4}$$

Therefore the simulated probability vectors $\mathbf{p}_1$ to $\mathbf{p}_6$ will be represented as

$$\begin{aligned}
\mathbf{p}_1 &= \frac{1}{1 + e^{-(\mathbf{X}_1 + \mathbf{X}_2 - 0.5\mathbf{X}_3 + 0.5\mathbf{X}_4 - \mathbf{X}_5 + \mathbf{X}_6)}} \\
\mathbf{p}_2 &= \frac{1}{1 + e^{-(\mathbf{X}_1 + \mathbf{X}_2 - 0.5\mathbf{X}_3 + 0.5\mathbf{X}_4 - \mathbf{X}_7 + \mathbf{X}_8)}} \\
\mathbf{p}_3 &= \frac{1}{1 + e^{-(\mathbf{X}_1 + \mathbf{X}_2 - 0.5\mathbf{X}_3 + 0.5\mathbf{X}_4 - \mathbf{X}_9 + \mathbf{X}_{10})}} \\
\mathbf{p}_4 &= \frac{1}{1 + e^{-(\mathbf{X}_1 + \mathbf{X}_2 - 0.5\mathbf{X}_3 + 0.5\mathbf{X}_4 - \mathbf{X}_{11} + \mathbf{X}_{12})}} \\
\mathbf{p}_5 &= \frac{1}{1 + e^{-(\mathbf{X}_1 + \mathbf{X}_2 - 0.5\mathbf{X}_3 + 0.5\mathbf{X}_4 - \mathbf{X}_{13} + \mathbf{X}_{14})}} \\
\mathbf{p}_6 &= \frac{1}{1 + e^{-(\mathbf{X}_1 + \mathbf{X}_2 - 0.5\mathbf{X}_3 + 0.5\mathbf{X}_4 - \mathbf{X}_{15} + \mathbf{X}_{16})}}.
\end{aligned} \tag{6.5}$$

Since the aim of the simulation exercise is to generate data sets, and therefore the values of both $\mathbf{X}$ and $\mathbf{y}$, we generate $\mathbf{y}$ as

$$y_i = \begin{cases} 1, & \text{if } u_i \leq p_i \\ 0, & \text{if } u_i > p_i, \end{cases} \tag{6.6}$$

where $\mathbf{u} = \{u_1, \dots, u_n\}$ is drawn from a $U(0, 1)$ distribution, and p_i is the element in $\mathbf{p}_j$ for simulated vector $\mathbf{x}_i$.

In this case $\mathbf{p}_1$ to $\mathbf{p}_6$ can be seen as the true fit for the simulated data sets against which any other estimate can be measured. In Chapter 5 we introduced the pseudo-F statistic, or CH value to measure the degree of separation from an explanatory variable perspective as well as the IV which measures target variable separation. Since the simulation described here is drawn from an independent sample in each case, the results will differ marginally, however Table 6.1 summarises the average results obtained from 100 simulation sets performed in the described way.

Average event rate Seg1	65.1%
Average event rate Seg2	61.5%
Average event rate Seg3	60.9%
Average event rate Seg4	34.9%
Average event rate Seg5	91.6%
Average event rate Seg6	50.3%
Average IV	0.91
Average CH value	0.27

TABLE 6.1: Average characteristic values of 100 simulated base case data sets

Comparing the average IV and CH value for each segment obtained in Table 6.1 to the values obtained on the industry data sets analysed in Chapter 5, we see that the simulated values are reasonable (see specifically Figures 5.2, 5.6, 5.10, 5.14, 5.18 and 5.22).

We also observe different event rates for each of the segments. The differences are caused by the differences in the distributions of explanatory variables $\mathbf{X}_3$ and $\mathbf{X}_4$ for each segment, since the values of $\mathbf{p}_j$ are all simulated to be dependent on (amongst others) $\mathbf{X}_3$ and $\mathbf{X}_4$. Given that all other variables were simulated with the same distribution, and the weights for $\mathbf{X}_3$ and $\mathbf{X}_4$ are identical for each segment, only the variance in distribution of $\mathbf{X}_3$ and $\mathbf{X}_4$ between segments can introduce a variance in the event rate.

Next we explore the Gini values obtained from using $\mathbf{p}_1$ to $\mathbf{p}_6$ as predictor (denoted as $\mathbf{p}$ when all segments are combined into a single data set) and compare it with the Ginis obtained when predictive models are fit separately on the segments. The Gini measures how well the predictive model is able to separate the values of the actual outcome (indicated by $\mathbf{y}$). To calculate the *true Gini*, we compare the value of $\mathbf{p}$ (see Equation 6.5) with $\mathbf{y}$ (see Equation 6.6). Note that the betas (β) used in Equation 6.5 are known (see Equation 6.2). To calculate the *fitted Gini*, we first estimate the betas (β) with logistic regression, denoted by $\hat{\beta}$, for each segment individually, and then calculate $\hat{\mathbf{p}}_j$ for each segment. Finally the combined vector, $\hat{\mathbf{p}}$ is compared with $\mathbf{y}$.

Logistic regression is used to fit the models as described in Section 5.3. The average of 100 simulated data sets is shown in Table 6.2. The important thing to note here is that, in this case, the segment allocation is provided, so segments are perfectly identified. We can see from Table 6.2 that on this data set, the fitted Gini is improved by more than 10% (in absolute terms) through perfect segmentation (from 71.8% to 81.9%). What can also be seen is that the segments with a low number of observations, like Segment 2 and Segment 6 shows a large decrease in Gini. This can be due to larger model uncertainty given the lower volume (although this is not the case for Segment 3). Note that for the results in Table 6.2, the data sets were not divided into build and validation sets.

Gini Type	Fitted Gini	True Gini
Seg1	77.3%	78.2%
Seg2	61.5%	75.6%
Seg3	70.6%	71.9%
Seg4	77.7%	78.0%
Seg5	83.9%	85.4%
Seg6	72.2%	76.8%
Combined	81.9%	83.9%
No Segmentation	71.8%	

TABLE 6.2: Fitted Ginis compared to true fit

Lastly we apply the segmentation algorithms explored in Chapter 5 to the generated data. Table 6.3 summarises the result. It can be seen that the SSSKMIV algorithm performs the best here. In this chapter we will only use the SSSKMIV algorithm (and

not the SSSKMCSQ algorithm as well) given the fact that we observed very close results for the two in Chapter 5, with the SSSKMIV algorithm being the stronger of the two in all cases we used.

Method	Best validation set Gini	Best Gini improvement	Average IV	Average CH value
Unsupervised clustering	74.45%	2.46%	0.35	0.090
SSSKMIV	74.89%	2.90%	0.64	0.087
Decision trees	73.42%	1.43%	1.01	0.069

TABLE 6.3: Success of segmentation algorithms

The values shown in Table 6.3 are as expected, with unsupervised clustering forming segments with the highest average CH values, but the lowest average IVs, whilst the decision tree obtains the highest average IV and the lowest average CH value. The SSSKMIV algorithm strikes a good balance between the two, but can only achieve a 2.9% Gini improvement at best (this is achieved with $w = 0.7$). This is lower than the optimal improvement of more than 10%. Note that Table 6.3 compares Gini improvements on the validation set, so the unsegmented Gini will be slightly different to the one shown in Table 6.2. The average IV of 1.01 achieved by the decision tree algorithm is slightly higher than the 0.91 of the actual segmentation, but none of the algorithms got close to the average CH value of 0.27 of the simulated segmentation.

6.3 Scenario simulation

In Section 6.2 we presented some of the concepts used to simulate data sets for segmented predictive modelling by providing an example of a base case scenario. The base case scenario was based on two main principles:

- Simulation of explanatory variables ($\mathbf{X}$) with known distributions. For most variables, the distributions were drawn from $N(0, 1)$, but the distributions of $\mathbf{X}_3$ and $\mathbf{X}_4$ were varied.
- Simulation of the dependent variable, or target variable ($\mathbf{y}$), with a known relationship with (or dependency on) $\mathbf{X}$. For all cases, $\mathbf{y}$ was simulated in such a way that it is dependent on 6 variables, with the last two being different for each segment.

As mentioned in the introduction, our aim with this chapter is to explore some of the reasons why one segmentation technique can outperform another depending on the characteristics of the data set used. By simulating data sets we are able to control some characteristics of the data on which segmentation for predictive modelling is performed. This gives us the opportunity to explore links between data set characteristics and dominance of a specific segmentation technique. Because practical data sets are in most cases made up of real-world data, which are extremely complex and diverse, we can never expect to find an exhaustive list of reasons for one technique outperforming another. We can however attempt to explore some of the more obvious links.

To this end, we will explore the impact of varying three main characteristics in simulated data sets, and we will explore their impact on the dominance of either unsupervised, supervised, or semi-supervised segmentation:

- Target rate separation between segments.
 - We know that supervised techniques are focused on finding segments that differ with regard to the target variable. For this reason we will explore how the degree of target rate separation between segments influences the success of our segmentation techniques.
 - As was done throughout this thesis, target rate separation between segments will be measured by the IV.
- Differences in explanatory variable distribution.
 - We know that unsupervised techniques are focused on finding segments that differ with regard to the explanatory variables. For this reason we will explore how the level of difference between the explanatory variable distributions of segments influence which segmentation technique is successful.
 - We will continue to use the pseudo-F statistic, or CH value to measure explanatory variable distribution differences between segments.
- Segment complexity.
 - Another dimension that could play a part in the success of segmentation techniques is the complexity of the underlying segments. As an example, distinguishing Segment 1 from Segment 2 could be as easy as saying: "If $\mathbf{X}_1 > 0.5$ then *Segment* = 1, Else *Segment* = 2", or it could require numerous rules using many explanatory variables.
 - We will define segment complexity by the number of explanatory variables that are used to define a segment. This will be simulated by varying the number of variables whose distributions are different for each segment.

Some assumptions presented in Section 6.2 will be retained, such as:

- We will continue to simulate six segments for each data set.
- We will continue to generate $\mathbf{y}$ such that it is dependent on six explanatory variables, with $\mathbf{X}_1 - \mathbf{X}_4$ being the first four, and the last two differing for each segment.
- We will continue to use $\{1, 1, -0.5, 0.5\}$ as the weights for $\mathbf{X}_1 - \mathbf{X}_4$ and $\{-1, 1\}$ for the last two explanatory variables on which $\mathbf{y}$ is dependent.

The process that will be followed to generate the results in this chapter is illustrated in the flow diagram in Figure 6.1.

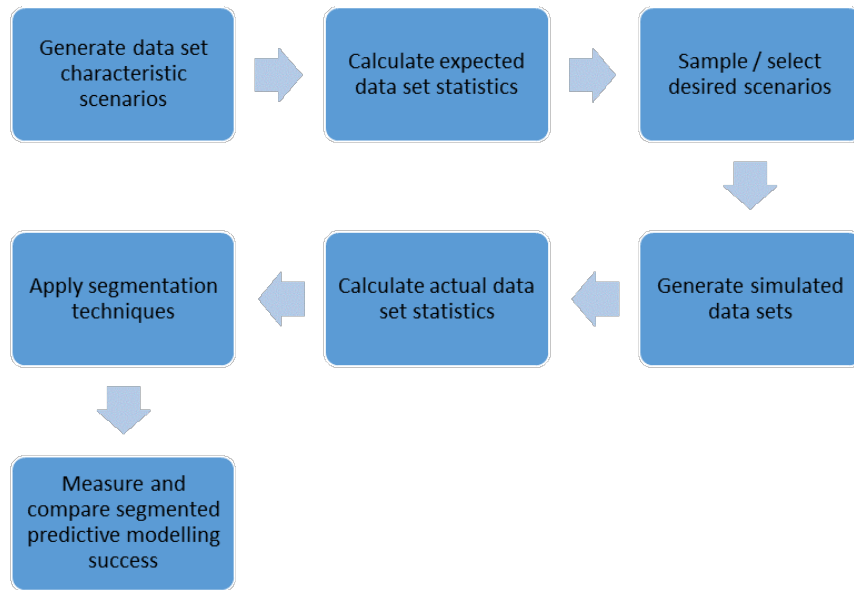


FIGURE 6.1: Process flow diagram for simulating data and comparing results

As can be seen, the process consists of seven steps:

1. Generate data set characteristic scenarios.
 - This step does not generate actual simulated data sets, but is responsible for generating a number of scenarios which prescribes the characteristics of data sets that can be simulated.
 - Each scenario that is generated contains information about how the explanatory variables should be simulated, the target rate for each segment as well as the segment complexity.
2. Calculate expected data set statistics.
 - Based on the characteristics generated in the first step, the expected values of the CH value and the IV can be calculated for each generated scenario.

3. Sample / select desired scenarios.
 - Since we will be exploring the characteristics that cause one segmentation technique to outperform another, we can now select the scenarios that will be simulated and used for comparison.
4. Generate simulated data sets.
 - The next step is to generate the simulated data sets selected in Step 3, according to the prescriptions generated in Step 1.
5. Calculate actual data set statistics.
 - Since the data are simulated and cannot be infinitely large, actual results for the CH and IVs will vary slightly from the expected results. After generating the required simulated data sets, the actual CH and IVs can be calculated.
6. Apply segmentation techniques.
 - The segmentation techniques explored in Chapter 5 are applied to the simulated data sets. As was done in Chapter 5, the combined Gini improvement of the segmented models is compared with the Gini obtained with no segmentation.
7. Measure and compare segmented predictive modelling success.
 - Since the true Gini for each segment as well as the true combined Gini is known, the success of each segmentation technique can be compared with the true Gini improvement, as was done in Section 6.2.
 - By studying the impact of simulated data set characteristics on segmentation technique success, possible reasons for one segmentation technique outperforming another can be explored.

We will discuss Steps 1 to 5 in the next sections after which we will summarise and evaluate the results. The methodologies used to segment and measure segmentation success in Steps 6 and 7 are identical to those described in Chapters 3 (see Sections 3.3 to 3.15) and 5 (see Sections 5.3 to 5.7), and will therefore not be repeated here.

6.3.1 Generate data set characteristic scenarios

The first step in the simulation process is to generate scenarios that prescribe the characteristics for a number of data sets to be simulated. As discussed, we are concerned with varying the scenarios according to the target variable, the explanatory variable

and the complexity. To this end, the algorithm consumes the following information as inputs:

- Number of scenarios (SC).
 - Indicates the number of scenarios to be generated.
- Number of segments (K).
 - Indicates the number of segments for which information should be generated. For our analysis we set $K = 6$.
- Average segment size ($\overline{SS}$).
 - Indicates what the average size of a segment should be.
- Complexity (O).
 - Complexity is measured by the number of explanatory variables that will be used to define a segment S_j , where $j = 1, \dots, 6$.

The algorithm uses this information to produce a table with SC rows, containing the following data set characteristics:

- SegSize.
 - SegSize is a $SC \times K$ matrix containing the segment size (SS_j) for each of the K segments and each of the SC scenarios.
 - Each SS_j is drawn from a normal distribution such that $SS_j = N(\overline{SS}, 0.2 \times \overline{SS})$, $SS_j \in (0, \infty)$ with $SS_j \in \mathbb{N}$. This is done to ensure that segments are varying in size. The value of 0.2 was derived through empirical analysis.
- Eventrate.
 - Eventrate is an $SC \times K$ matrix prescribing the average target rate for each segment S_j .
 - Eventrate is drawn from normal distribution $N(0.5, 0.2)$ such that $Eventrate_j \in (0.02, 0.98)$. The limits of 2% and 98% are set to ensure an IV can always be calculated for each simulated data set, since IVs are not defined for bad rates of 100% or 0% (Siddiqi, 2006).

- XMean.
 - XMean is a $SC \times K \times O$ matrix containing the mean values of the O variables that will be used to describe each of the K segments. If, for example, X_{15} is used, then X_{15} will have a different mean value for each segment S_j . Values are generated for each of the SC scenarios.
 - The mean values are drawn from the uniform distribution $U(0, 3)$.
 - As will be discussed in Section 6.3.4, the final simulated data sets will consist of 26 explanatory variables. However, the distributions of all other variables will be drawn from $N(0, 1)$ for each segment, which means there is no need for this information to be contained within the scenario information.
- XStd.
 - XStd is an $SC \times K \times O$ array containing the standard deviations of the O variables that will be used to describe each of the K segments. Similar to XMean, the standard deviation of the selected variable will differ for each segment.
 - The standard deviations are also drawn from the uniform distribution $U(0, 3)$ for each variable that will be varied.

Table 6.4 shows an example of the information that is generated for a single scenario. In this example, the number of segments required was six ($K = 6$), the average required segment size was 200 ($\overline{SS} = 200$) and the complexity was set to three ($O = 3$).

6.3.2 Calculate expected data set statistics

In Sections 2.11.1 and 2.12, the formulas used to calculate the IV and CH value are provided. Since we have the size of each segment as well as the event rate for each segment from the scenarios generated in Step 1 (described in Section 6.3.1), it is easy to derive P_j^T and P_j^F for each segment, and therefore also the expected IV for each scenario using Equation 2.15.

The information for Equation 2.18 (the CH value) is also available from the generated scenarios. Recall from Section 2.12 that the CH value is calculated as the ratio between the separation (or "between cluster sum-of-squares") to the cohesion (or "within cluster sum-of-squares").

The between cluster sum-of-squares, which is the numerator of Equation 2.18, is calculated as $\sum_{j=1}^K |S_{cj}| (\mathbf{u} - \mathbf{u}_{cj})$, and therefore requires the centre (average) of each segment

Field	Value
SegSize1	266
SegSize2	213
SegSize3	225
SegSize4	194
SegSize5	195
SegSize6	175
Eventrate1	49.0%
Eventrate2	31.0%
Eventrate3	78.0%
Eventrate4	54.0%
Eventrate5	88.0%
Eventrate6	72.0%
$X_u-S_1-X_1$	0.83
$X_u-S_1-X_2$	0.92
$X_u-S_1-X_3$	2.18
$X_u-S_2-X_1$	2.56
$X_u-S_2-X_2$	1.63
$X_u-S_2-X_3$	1.52
$X_u-S_3-X_1$	1.64
$X_u-S_3-X_2$	0.79
$X_u-S_3-X_3$	1.69
$X_u-S_4-X_1$	2.31
$X_u-S_4-X_2$	1.92
$X_u-S_4-X_3$	2.7
$X_u-S_5-X_1$	2.02
$X_u-S_5-X_2$	0.34
$X_u-S_5-X_3$	0.92
$X_u-S_6-X_1$	1.32
$X_u-S_6-X_2$	1.97
$X_u-S_6-X_3$	2.9
$\sigma^2-S_1-X_1$	2.24
$\sigma^2-S_1-X_2$	2.43
$\sigma^2-S_1-X_3$	2.5
$\sigma^2-S_2-X_1$	2.03
$\sigma^2-S_2-X_2$	1.25
$\sigma^2-S_2-X_3$	1.12
$\sigma^2-S_3-X_1$	1.7
$\sigma^2-S_3-X_2$	0.85
$\sigma^2-S_3-X_3$	0.06
$\sigma^2-S_4-X_1$	2.82
$\sigma^2-S_4-X_2$	2.08
$\sigma^2-S_4-X_3$	1.39
$\sigma^2-S_5-X_1$	1.3
$\sigma^2-S_5-X_2$	1.05
$\sigma^2-S_5-X_3$	2.93
$\sigma^2-S_6-X_1$	1.34
$\sigma^2-S_6-X_2$	1.69
$\sigma^2-S_6-X_3$	2.37

TABLE 6.4: Example of the information generated for a scenario

as well as the centre (average) of the entire data set. Since we know the centres of each of the O variables that will be varied, and we know that all other variables will have a mean of zero, the centre of each segment $\mathbf{u}_{cj}$ is simply the vector of the means of each of the variables for that segment. The centroid of the entire data set, $\mathbf{u}$ is a $1 \times m$ vector with each element, μ_{cp} , calculated as $\mu_{cp} = \frac{\sum_{j=1}^K |S_{cj}| u_{cjp}}{\sum_{j=1}^K |S_{cj}|}$, where u_{cjp} is the p^{th} element of vector $\mathbf{u}_{cj}$.

To calculate the "within cluster sum-of-squares" of each segment, we use the standard deviation of each of the variables that is varied for each segment. We also know that the standard deviation of the remaining variables will be equal to one. The within cluster sum-of-squares can now be calculated as the sum of the standard deviations of each of the variables in that segment, or $WSS_j = \sum_{p=1}^K \sigma_p^2$.

At the end of this step we therefore have SC scenarios which contain information about the event rate per segment as well as the explanatory variables to be generated. We also have expected values of IV and the CH value for each scenario.

Figure 6.2 shows a histogram which illustrates the distribution of IVs as generated by the algorithm discussed in Section 6.3.1. 20000 scenarios were generated to construct the histogram. As can be seen, the distribution has a lognormal shape, peaking between IVs of 0.2 and 0.4. Many examples of scenarios can be found with similar IVs (ranging between 0 and 1.5), which correlates well with the ranges of IVs obtained in Chapter 5 (see again Figures 5.2, 5.6, 5.10, 5.14, 5.18 and 5.22).

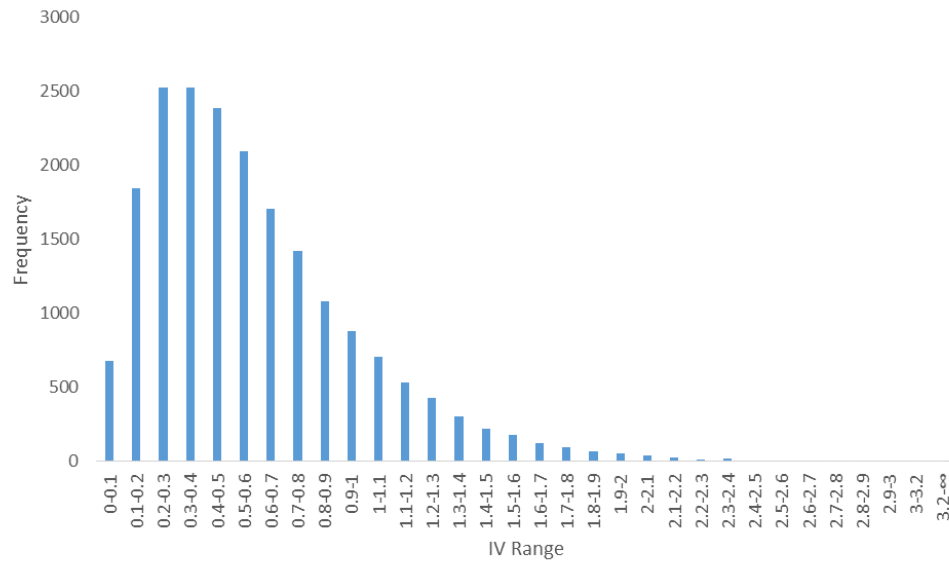


FIGURE 6.2: Distribution of IVs of 20000 scenarios

Since the complexity of a data set is measured by the number of variables that are varied, it also has an influence on the CH values generated. For this reason, Figure 6.3

shows the difference in the CH value distribution for two scenarios, one where $O = 1$ and one where $O = 15$. It can be seen that both distributions have a normal shape, but the average CH value differs depending on the value of O .

The CH values generated seems lower than those in Chapter 5, but analysis has shown that generating these data with higher CH values makes it too easy for the algorithms to identify the segments, which hinders our specific goal in this chapter.

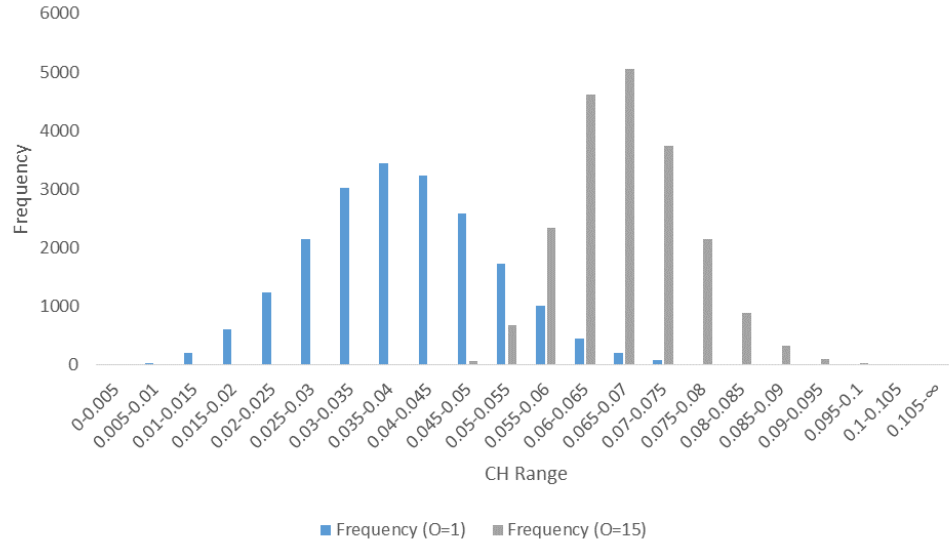


FIGURE 6.3: Distribution of CH values of 20000 scenarios, with $O = 1$ and $O = 15$

6.3.3 Sample / select desired scenarios

Because we want to be selective of the data set characteristics we simulate to ensure we can test a wide range of values for IV, CH and complexity, we need to carefully select the scenarios that will be tested. This may require that we generate many scenarios in Step 1, since we have seen from Figures 6.2 and 6.3 that the probability distributions are not uniform, and the combination of the specific criteria (i.e. combinations of specific CH values and IV's) we need in scenarios may be scarce.

As discussed in Section 6.3.1, the complexity, O , is an input into the algorithm that generates the scenarios. New scenarios will therefore be generated for each value of O that is required (in our case we will simulate data sets with $O \in \{5, 10, 15\}$). Since the value of O is therefore already established in the scenarios, the focus of this step is on selecting a representative range of CH and IVs from the scenarios generated. To this end, we make use of stratified sampling.

The IVs are stratified in four groups. Group 1 consists of all scenarios with IVs in the range $(0, 0.05]$ to ensure there is an adequate sample from really low IVs. Group 2

contains all scenarios with IVs in the range $(0.05, 0.5]$; Group 3, values within $(0.5, 0.8]$; and the last group consists of all IVs above 0.8. The values used for the ranges of each of the groups were based on analysis of the results observed for the different segmentation techniques on data sets with IVs in these ranges. As will be seen in the results section, grouping the IVs in this way provides us with enough volume in IV areas where different segmentation techniques perform well. This provides a comparative view of how IVs influence the effectiveness of specific segmentation techniques. The specific IV groupings were determined by considering the ranges of IVs obtained in Chapter 5 along with iterative analysis of the data and results. The goal was to find ranges where the differences in results can be easily observed, and therefore ensure that the results can be clearly and concisely conveyed in the next sections.

For CH we have the additional complexity that the range of the CH values differ depending on the value of O . We therefore follow a slightly different approach compared to IV:

- We firstly divide all the scenarios that are generated for a specific value of O into deciles (ranked groups consisting of 10% of the total number of scenarios) based on the CH value.
- We only select scenarios from the first, fifth and tenth decile for a specific value of O to facilitate a good spread of CH values without the need to do too many iterations. The first decile contains the highest CH values, and is therefore called the "High" group. The fifth decile contains mid-range values of the CH value, and is therefore called "Mid". The tenth decile contains the lowest CH values, and is subsequently called "Low".

Proc Surveyselect is used with stratified sampling to sample the scenarios that will be generated. To facilitate this, a strata field is generated that is a combination of the IV group and the CH group. A total of 12 (4×3) groups are therefore generated. An equal number of scenarios is selected from each strata group. This ensures optimal distribution of characteristics for the analysis.

Table 6.5 shows an example of the selected strata, and average IV and CH values for a number of scenarios that was generated with $O = 3$. In this case, each group contains 30 scenarios, which means this analysis would require 360 data sets. The table is shown here to illustrate how the IV groups and CH groups are combined and subsequently used for stratification.

Strata (IVGroup.-CHGroup)	Nr Scenarios	Average IV	Average CH value
0.0-0.05_High	30	0.036	0.090
0.0-0.05_Mid	30	0.030	0.070
0.0-0.05_Low	30	0.036	0.052
0.05-0.5_High	30	0.304	0.088
0.05-0.5_Mid	30	0.297	0.070
0.05-0.5_Low	30	0.300	0.051
0.5-0.8_High	30	0.630	0.088
0.5-0.8_Mid	30	0.661	0.070
0.5-0.8_Low	30	0.637	0.052
0.8-Inf_High	30	1.153	0.088
0.8-Inf_Mid	30	1.174	0.070
0.8-Inf_Low	30	1.125	0.052

TABLE 6.5: Example of strata use and average IV and CH values for $O = 3$

6.3.4 Generate simulated data sets

Now that we have selected the scenarios to be simulated, the next step is the simulation itself. As discussed in Section 6.3.1, the scenarios contain information about segment sizes, number of variables to be varied (complexity), the means and standard deviations of those variables for each segment as well as the event rate per segment. These are now treated as inputs, with the goal being to generate data sets that conform as closely as possible to the scenarios.

In Section 6.2, we observed how changes in explanatory variable distributions can influence the generated event rate of a segment (see Table 6.1). Due to the fact that $\mathbf{p}_1$ to $\mathbf{p}_6$ are functions of selected explanatory variables, and $\mathbf{y}$ is generated from $\mathbf{p}_1$ to $\mathbf{p}_6$, it will be very complex to control for a specific required event rate if the explanatory variables on which $\mathbf{p}_1$ to $\mathbf{p}_6$ are based is varied. Since control of the event rate per segment is an important part of this analysis, we will separate the variables used to identify segments from the variables that are used to generate the target variable. This concept is shown in Figure 6.4. As can be seen, for the purposes of this analysis, variables $\mathbf{X}_1$ to $\mathbf{X}_{11}$ will be used to generate $\mathbf{p}_1$ to $\mathbf{p}_6$ as described in Section 6.2. Variables $\mathbf{X}_{12}$ to $\mathbf{X}_{26}$ will then be available to be varied depending on the complexity selected. This means that, for this exercise, O can vary between 1 and 15. 26 variables were chosen in this instance as this allows us to increase the complexity (O) to 15, and analysis has shown that a complexity of 15 (i.e. $O = 15$) is enough to provide conclusive results (see Sections 6.5.1 to 6.5.2).

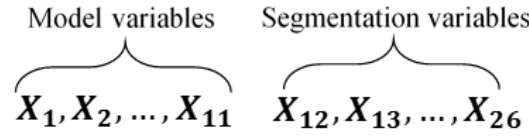


FIGURE 6.4: Variable roles

The data sets for each scenario is generated as follows:

1. Randomly identify O variables from $\mathbf{X}_{12}$ to $\mathbf{X}_{26}$ to apply the standard deviation and mean inputs to.
 - This means that any of $\mathbf{X}_{12}$ to $\mathbf{X}_{26}$ can be varied for any data set.
2. Generate explanatory variables $\mathbf{X}_1$ to $\mathbf{X}_{26}$.
 - Note here that all variables, other than the O selected ones will be drawn from $N(0, 1)$.
3. Generate the dependent variable $\mathbf{y}$.
 - Since the average event rate of $\mathbf{y}$ needs to be controlled, the way in which $\mathbf{y}$ is generated differs from the base case scenario. This is discussed below.

In Section 6.2, the way in which $\mathbf{p}_j$ is generated is shown in Equation 6.5. A similar approach is taken here, but in order to use as few variables as possible, the last two variables are allowed to overlap, as can be seen in Equation 6.7. Even though these variables overlap, it can be seen that the weights are different in each case, and therefore a similar impact is achieved. The probabilities are defined as

$$\begin{aligned}
 \mathbf{p}_1 &= \frac{1}{1 + e^{-(\mathbf{X}_1 + \mathbf{X}_2 - 0.5\mathbf{X}_3 + 0.5\mathbf{X}_4 - \mathbf{X}_5 + \mathbf{X}_6)}} \\
 \mathbf{p}_2 &= \frac{1}{1 + e^{-(\mathbf{X}_1 + \mathbf{X}_2 - 0.5\mathbf{X}_3 + 0.5\mathbf{X}_4 - \mathbf{X}_6 + \mathbf{X}_7)}} \\
 \mathbf{p}_3 &= \frac{1}{1 + e^{-(\mathbf{X}_1 + \mathbf{X}_2 - 0.5\mathbf{X}_3 + 0.5\mathbf{X}_4 - \mathbf{X}_7 + \mathbf{X}_8)}} \\
 \mathbf{p}_4 &= \frac{1}{1 + e^{-(\mathbf{X}_1 + \mathbf{X}_2 - 0.5\mathbf{X}_3 + 0.5\mathbf{X}_4 - \mathbf{X}_8 + \mathbf{X}_9)}} \\
 \mathbf{p}_5 &= \frac{1}{1 + e^{-(\mathbf{X}_1 + \mathbf{X}_2 - 0.5\mathbf{X}_3 + 0.5\mathbf{X}_4 - \mathbf{X}_9 + \mathbf{X}_{10})}} \\
 \mathbf{p}_6 &= \frac{1}{1 + e^{-(\mathbf{X}_1 + \mathbf{X}_2 - 0.5\mathbf{X}_3 + 0.5\mathbf{X}_4 - \mathbf{X}_{10} + \mathbf{X}_{11})}}.
 \end{aligned} \tag{6.7}$$

Since the program consumes as input a target event rate for each segment ($Eventrate_j$), we need to adjust Equation 6.6 under the assumption of $\mathbf{X}_1$ to $\mathbf{X}_{11}$ being drawn from $N(0, 1)$ as follows:

If $Eventrate_j < 0.5$:

$$y_i = \begin{cases} 1, & \text{if } u_i \leq p_{ij} \times Eventrate_j \times 2, \quad y_i \in S_j \\ 0, & \text{if } u_i > p_{ij} \times Eventrate_j \times 2, \quad y_i \in S_j, \end{cases} \quad (6.8)$$

else if $Eventrate_j \geq 0.5$:

$$y_i = \begin{cases} 1, & \text{if } u_i \leq 1 - 2(1 - Eventrate_j) + p_{ij} \times 2(1 - Eventrate_j), \quad y_i \in S_j \\ 0, & \text{if } u_i > 1 - 2(1 - Eventrate_j) + p_{ij} \times 2(1 - Eventrate_j), \quad y_i \in S_j, \end{cases} \quad (6.9)$$

where u_i is drawn from $U(0, 1)$ as per Equation 6.6 and p_{ij} is the i^{th} observation of $\mathbf{p}_j$. The purpose of Equations 6.8 and 6.9 is to adjust the probability of an event ($Pr(y_i = 1)$) occurring according to the event rate required.

Note that our approach to simulate data sets with known characteristics differs in some ways from standard data simulation exercises (see for instance Shifa and Rashid (2003) and Venter and De la Rey (2007)). This is done to ensure we are able to control the three characteristics (target rate separation between segments, differences in explanatory variable distribution and segment complexity) as independently as possible. In the base case scenario (Section 6.2), the target rate was influenced by the distributions of the explanatory variables (specifically $\mathbf{X}_3$ and $\mathbf{X}_4$). If this is done here, then controlling both the first two characteristics simultaneously (i.e. target rate separation and differences in explanatory variable distribution) will be extremely complex.

In order to explain Equations 6.8 and 6.9 further, we will make use of an example generated data set.

We focus first on the formulas used to calculate $\mathbf{p}_1$ to $\mathbf{p}_6$ in Equation 6.7. Since we ensure that $\mathbf{X}_1$ to $\mathbf{X}_{11}$ are all drawn from random normal distributions with means equal to zero, the average values of $\mathbf{p}_1$ to $\mathbf{p}_6$ are therefore expected to equal 0.5 (since $\frac{1}{1+e^{-(0+0-0.5 \times 0+0.5 \times 0-0+0)}} = 0.5$). We also know that the values of $\mathbf{p}_1$ to $\mathbf{p}_6$ will be distributed according to the standard logistic s-shaped curve and have values between zero and one (Hosmer et al., 2013, Neter et al., 1996).

To illustrate this, we use the results of a simulation with six segments each with 2000 observations, where the values of $\mathbf{p}_1$ to $\mathbf{p}_6$ were generated according to Equation 6.7. As an example, the event rate requested ($Eventrate_j$) for each segment can be set as

summarised in Table 6.6. Note that this is simply an example to illustrate Equations 6.8 and 6.9, the data sets we will simulate will each have unique event rates.

$Eventrate_1$	10%
$Eventrate_2$	20%
$Eventrate_3$	40%
$Eventrate_4$	50%
$Eventrate_5$	70%
$Eventrate_6$	90%

TABLE 6.6: Average requested event rate for example data set

In Figure 6.5 the cumulative distribution of $\mathbf{p}_1$ to $\mathbf{p}_6$ for the example data set is shown. It can be seen that these are virtually identical, and they only differ due to statistical variation (confirming our expectation that the average value of $\mathbf{p}_j$ would always be approximately equal to 50%).

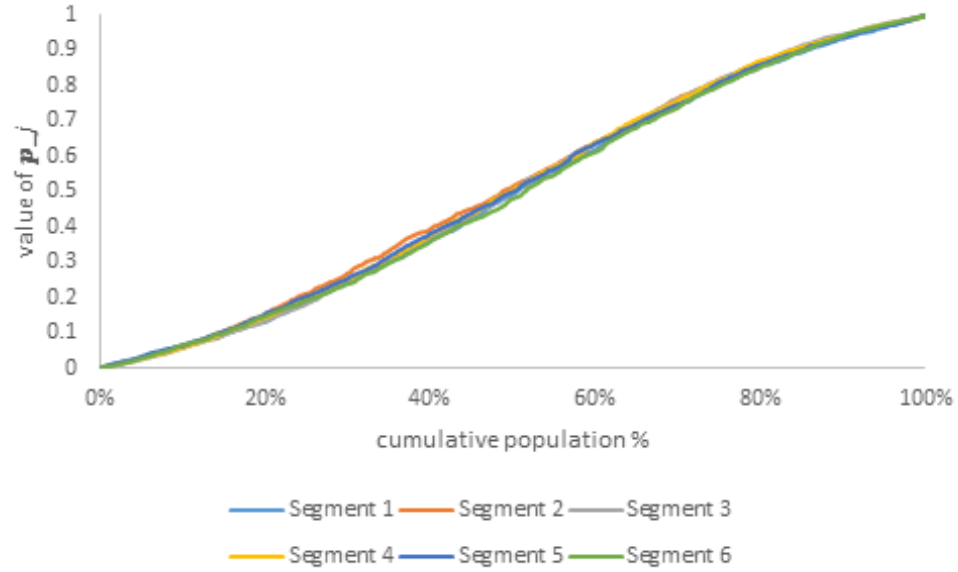
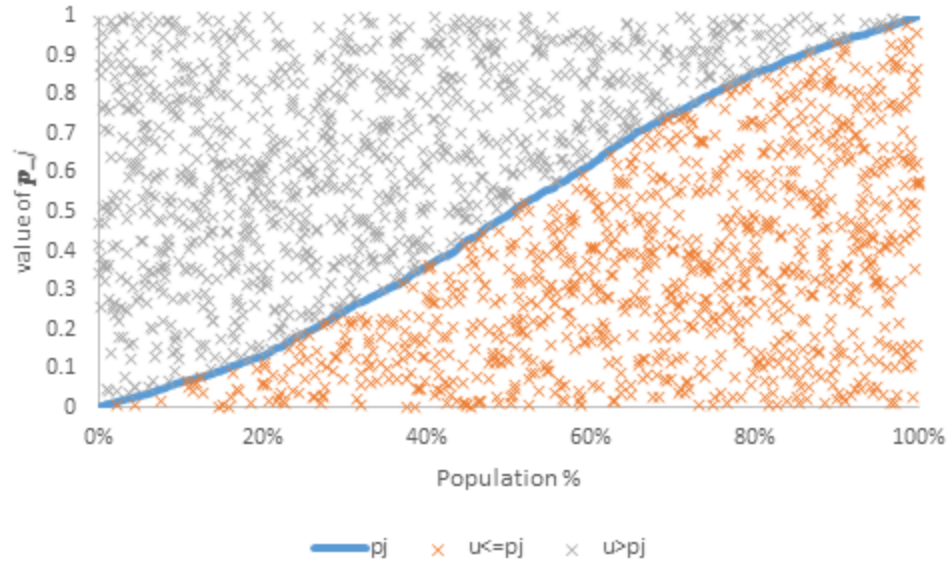


FIGURE 6.5: Cumulative population distribution of six simulated segments

In our base case scenario discussed in Section 6.2, we assigned $y_i = 1$ if $u_i \leq p_i$ (see Equation 6.6). Figure 6.6 illustrate the logic of this visually if we were to apply it to the segments in the example data set. All observations with values of $\mathbf{u}$ that are greater than $\mathbf{p}_j$ will be assigned a value of 1 to the target variable, $\mathbf{y}$. These are represented by the grey crosses. The orange crosses are where $\mathbf{u} \leq \mathbf{p}_j$. It is clear from the figure that, since the logistic function is symmetrical, the number of grey crosses are similar to the number of orange crosses, resulting in an event rate of close to 50%.

In order to visually demonstrate how Equations 6.8 and 6.9 adjust the final event rate, we will use Segment 1 and Segment 6 as examples. This is done to illustrate the impact of both equations, and since the event rate requested for Segment 1 in this example is

FIGURE 6.6: Illustrating how y is assigned

below 50%, Equation 6.8 will be used, whilst Equation 6.9 will be used for Segment 6. For Segment 1 in this example, Equation 6.8 solves to

$$y_i = \begin{cases} 1, & \text{if } u_i \leq 0.2 \times p_{ij}, \quad y_i \in S_j \\ 0, & \text{if } u_i > 0.2 \times p_{ij}, \quad y_i \in S_j. \end{cases} \quad (6.10)$$

For Segment 6 in this example, Equation 6.9 solves to

$$y_i = \begin{cases} 1, & \text{if } u_i \leq 0.8 + 0.2 \times p_{ij}, \quad y_i \in S_j \\ 0, & \text{if } u_i > 0.8 + 0.2 \times p_{ij}, \quad y_i \in S_j. \end{cases} \quad (6.11)$$

The results of Equations 6.10 and 6.11 are visually illustrated in Figures 6.7 and 6.8.

In order to confirm that the required bad rates (i.e. 10% and 90%) are achieved by Equations 6.10 and 6.11, we re-arrange the formulae as

$$y_i = \begin{cases} 1, & \text{if } 0 \leq 0.2 \times p_{ij} - u_i, \quad y_i \in S_j \\ 0, & \text{if } 0 > 0.2 \times p_{ij} - u_i, \quad y_i \in S_j, \end{cases} \quad (6.12)$$

$$y_i = \begin{cases} 1, & \text{if } 0 \leq 0.8 + 0.2 \times p_{ij} - u_i, \quad y_i \in S_j \\ 0, & \text{if } 0 > 0.8 + 0.2 \times p_{ij} - u_i, \quad y_i \in S_j. \end{cases} \quad (6.13)$$



FIGURE 6.7: Graphical representation of how the target variable is assigned for Segment 1

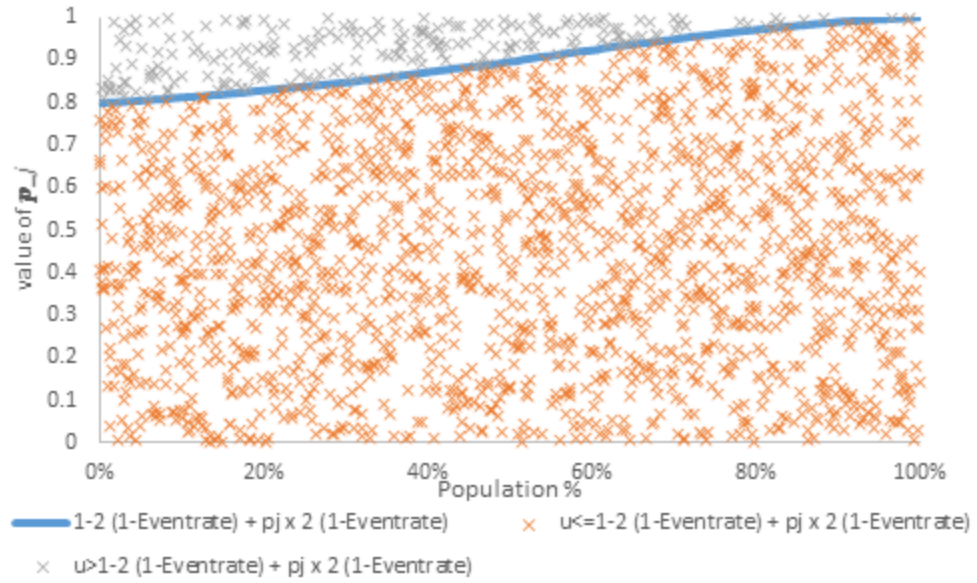


FIGURE 6.8: Graphical representation of how the target variable is assigned for Segment 6

We are therefore looking for the percentage of the population where Equations 6.12 and 6.13 are greater than 0. The cumulative distribution curve of Equations 6.12 and 6.13 are respectively plotted in Figures 6.9 and 6.10.

In Figure 6.9 we can see that, for Segment 1, only 10% of the population achieves values for Equation 6.12 that are greater than 0, which means that 10% of the observations' target variables (y) will be assigned a value of one. In Figure 6.10 the opposite is true, and Equation 6.13 is above the value of 0 in 90% of the cases, resulting in an event rate of 90%.

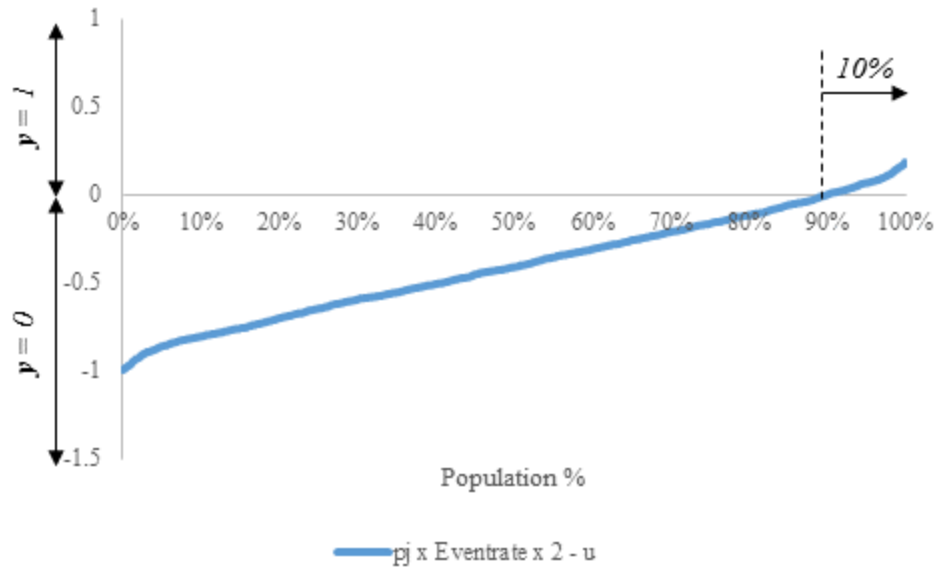


FIGURE 6.9: Graphical presentation of the cumulative distribution of Equation 6.12

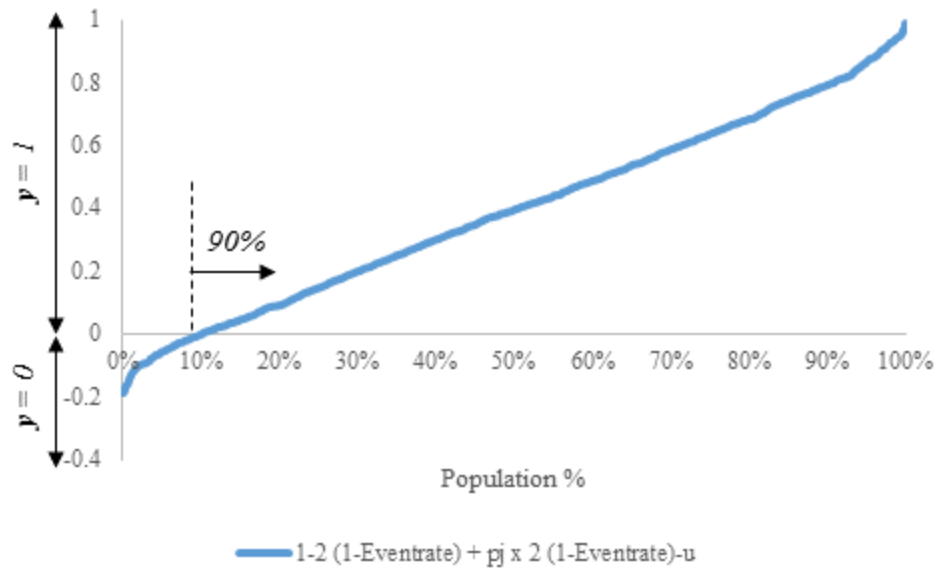


FIGURE 6.10: Graphical presentation of the cumulative distribution of Equation 6.13

It can therefore be seen that Equations 6.8 and 6.9 are able to successfully achieve the desired event rate as requested by the user.

6.3.5 Calculate actual data set statistics

Once all data sets have been simulated, it is now possible to calculate the actual data set statistics and compare it with the statistics that were requested by each scenario.

Note that we have been referring to "true" CH and IVs as well as "true" Ginis in Section 6.2. We refer here to "actual" vs. "estimated" values since we are generating data sets

with specified segments as well as specified event rates per segment. Although the event rates and IV or CH values will be exactly equal to the requested values if data sets were infinitely large, the fact that we limit the size of each segment will mean that the final values obtained will be slightly different due to statistical error. We therefore refer here to "actual" and "estimated" values, since we first estimate what the statistics will be (if the data set is infinitely large), but after the data set has been generated, we can measure exactly what the final statistics actually ended up being.

The final CH value for each scenario can be compared to the estimated value to measure how well the explanatory variables were simulated. In a similar way one can compare the IV obtained for each scenario with what was estimated in order to measure how well the targeted event rates per segment were achieved.

Figure 6.11 shows the accuracy achieved for the IV for a typical set of scenarios that was simulated. The average data set size ($\overline{SS}$) was chosen as 1000 here, and the value of O was set to three. It can be seen that the algorithm accurately simulates the required IVs. The r-square value measured between the expected and actual IVs is 0.991. The accuracy of the expected vs. actual IV and CH values is used here to measure how closely the required data set characteristics were simulated. Analysis of the results confirm that the simulated data sets have characteristics that correlate strongly with expectations.

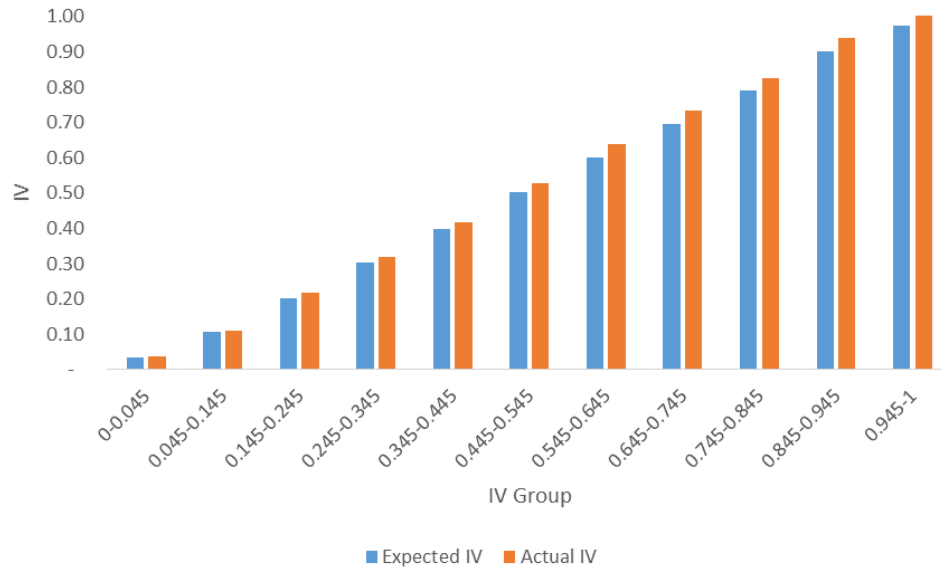


FIGURE 6.11: Accuracy of the simulated IVs

Figure 6.12 shows the accuracy achieved for the CH value. The CH value is grouped into the three groups discussed in Section 6.3.3. The r-square value measured between the raw expected and actual CH values is 0.995.

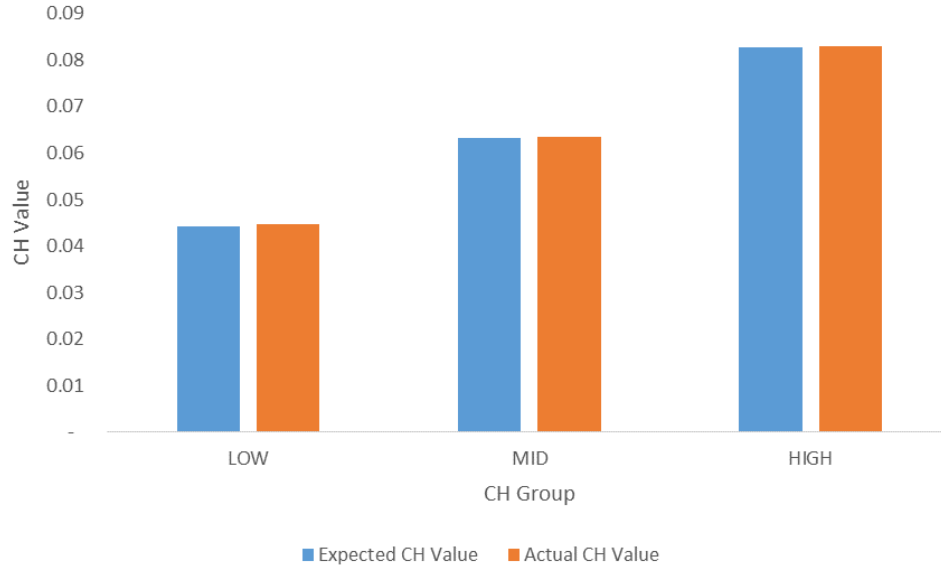


FIGURE 6.12: Accuracy of the simulated CH values

In the following sections we will again refer to the "true" statistics of each segment, which will refer to the final "actual" values as calculated in this step.

6.4 Apply segmentation techniques

As discussed, the approach followed to apply the segmentation techniques to the simulated data sets in this chapter will be similar to the approach followed in Chapter 5. We will, however, make some changes to a few inputs into the algorithm:

1. Because the focus here is on analysing many data sets and less on getting the best answer for a single data set, we reduce the number of values for w that we consider for the SSSKMIV algorithm. In Chapter 5, we considered values for w ranging from 0.1 to 0.9, and performed 20 iterations on each value. For this analysis we will only perform a single iteration for each $w \in \{0.25, 0.5, 0.75\}$. Running the same number of iterations on hundreds of data sets will require a lot of time without adding any notable value.
2. We will select three values for O : $O \in \{5, 10, 15\}$ and test the impact of the segmentation algorithm on each.
3. We will use an average segment size $\overline{SS} = 1000$, since through analysis this has proven to be large enough to provide robust results (this means that on average a data set contains 6000 observations).

The selections made above meant that a total number of 1800 segmentation iterations was performed and 10800 models developed for each value of O . In addition to this, time was required to select and generate the required data sets. Even though the size of the data sets were relatively small, the estimated time required to perform the analyses per value of O was between four and five days.

6.5 Results

We will now discuss the results that was obtained from running segmentation analysis on 360 data sets for each value of $O \in \{5, 10, 15\}$. Similar to the example in Table 6.5, we will therefore select 30 scenarios from each of the strata. We will first discuss the lowest complexity case ($O = 5$) and then contrast it with the highest ($O = 15$). Discussing these first will put the results of the scenario where $O = 10$ into perspective more clearly.

6.5.1 Results on $O = 5$

The first aspect that we investigate is focused on the IV and CH values obtained by each of the segmentation techniques, and how it compares with the true CH and IVs.

The IVs obtained by each segmentation technique are shown in Figure 6.13 and Table C.1. Appendix C will be used in this chapter for presenting data or results that are not practical to include within the main body of the chapter.

What is evident, is that the order of IVs obtained rank orders with unsupervised segmentation obtaining the lowest and decision trees (or supervised segmentation) achieving the highest IVs (similar to what was seen in Chapter 5). It can, however, be seen that for higher IVs the algorithms struggle to obtain IVs that are even close to the true IVs.

Figure 6.14 and Table C.2 summarise the CH values obtained by the segmentation techniques. Again, the achieved values are similar to our results from Chapter 5, with unsupervised techniques achieving the best and decision trees the worst. What is interesting to see here is that the CH values obtained are actually higher than the true CH values. This probably stems from the requirement of the segmentation techniques used here to have clear segment assignments, where segments tend to overlap in reality. The within cluster sum-of-squares will be reduced by clear segment assignments.

We next look at the best Gini values obtained for each of the segmentation techniques, and compare it to the Gini with no segmentation and the true Gini. This is shown by IV

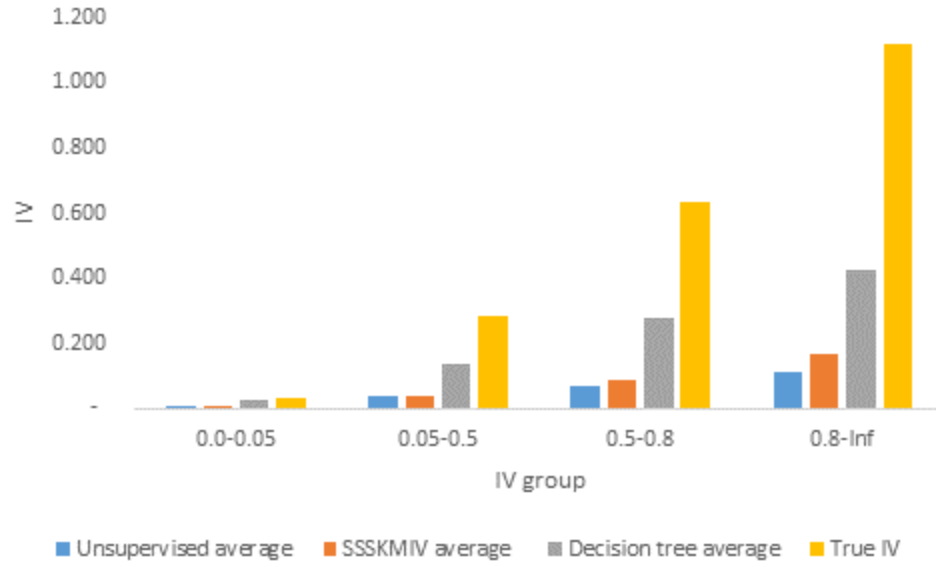
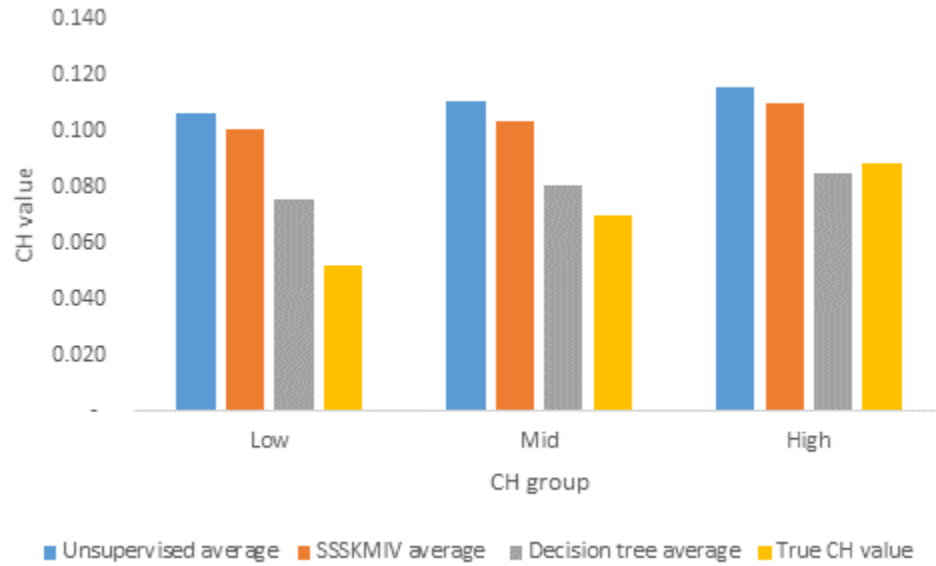


FIGURE 6.13: IVs obtained by segmentation technique and IV group

FIGURE 6.14: CH values obtained by segmentation technique and CH group for $O = 5$

group in Figure 6.15 and Table C.3. Note that the best Gini is found for each scenario here, and the results are averaged for each group. This means that there are no single value of w which can be identified as achieving the best Ginis for SSSKMIV. We will instead discuss some results regarding the value of w in Section 6.5.5.

Grouping the results by IV group show some clear trends. We can see that the decision tree tends to dominate the results for $O = 5$, but it seems that it performs the worst (albeit by a small margin) for the low IVs. This is to be expected since the decision tree

only takes the target variable into account, and when there is little distinguishing the segments from a target variable perspective, the decision tree will struggle.

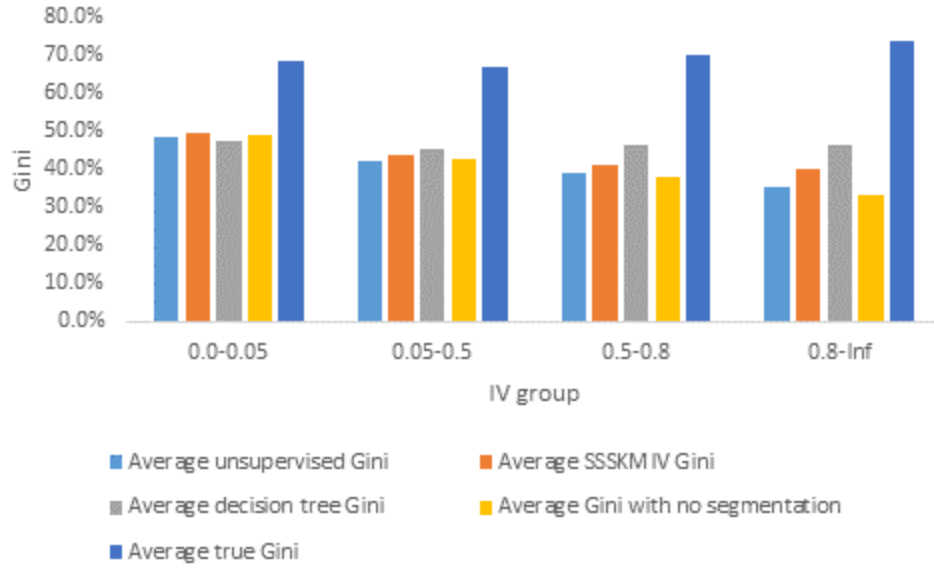


FIGURE 6.15: Ginis obtained by segmentation technique and IV group for $O = 5$

A clearer way to show the difference between the techniques is to focus on the improvement obtained by segmenting with each of the segmentation techniques. Since we also have the true Gini for these simulated sets, we can express that improvement as a percentage of the true improvement. We therefore define the Gini improvement as a percentage of true improvement, namely $Gini_{imp\%} = \frac{Gini - Gini_{NS}}{Gini_{True} - Gini_{NS}}$, where $Gini$ is the Gini obtained by the segmentation technique and $Gini_{NS}$ is the Gini obtained when a model is built on the data set without any segmentation. This is shown for $O = 5$ by IV group in Figure 6.16 and Table C.5.

The dominance of the decision tree is even clearer in this view, with unsupervised segmentation clearly struggling, and even reducing the Gini in some instances. At best, the techniques can only obtain around 30% of the true Gini improvement. This is due to the fact that the number of variables that are varied, O , is low which means that there is not a lot of information available to distinguish one segment from another.

Figure 6.17 and Table C.6 show the Gini improvement percentage by CH group. Here the decision tree clearly dominates, but a clear improvement can be seen in the success of the algorithms as the CH value increases. This is expected since higher CH values in the true segments mean that they are easier to distinguish. The actual Ginis obtained by CH group can be seen in Table C.4.

Our last graph for $O = 5$ splits the results into both IV and CH groups. This can be seen in Figure 6.18 and Table C.7. The best possible Gini improvement percentage is

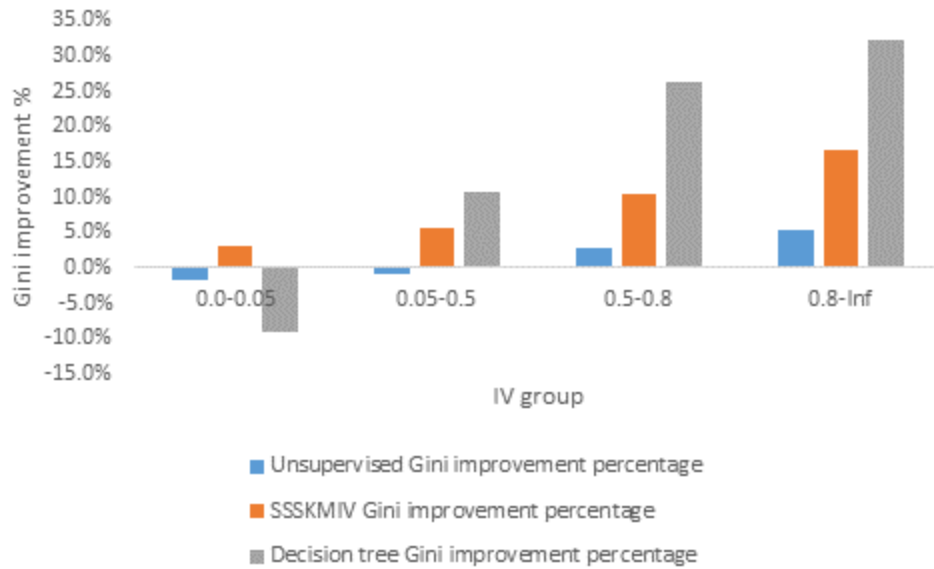


FIGURE 6.16: Gini improvement as a percentage of true improvement obtained by segmentation technique and IV group for $O = 5$

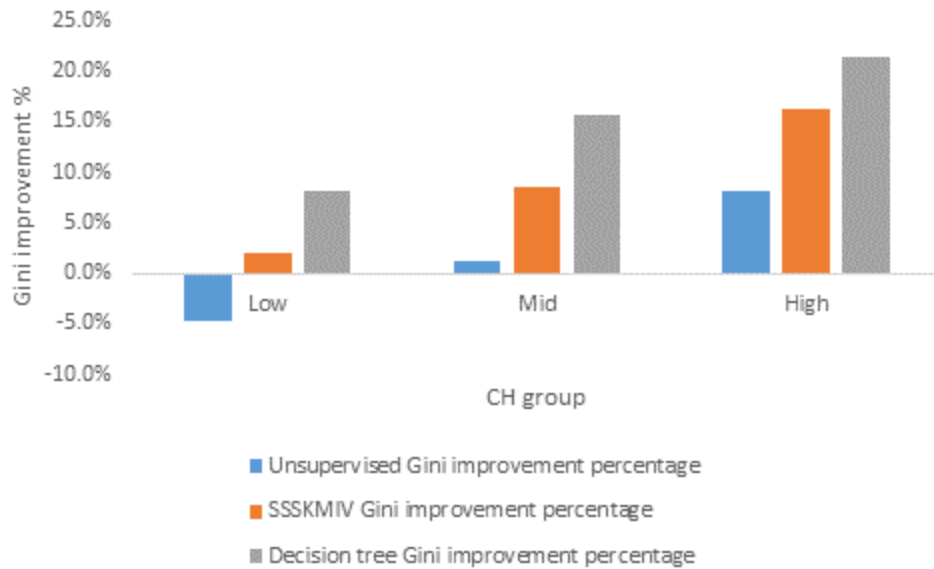


FIGURE 6.17: Gini improvement as a percentage of true improvement obtained by segmentation technique and CH group for $O = 5$

obtained by the decision tree when CH values are high and IVs are greater than 0.8. This group obtains an average of just over 40% of the true Gini improvement. Although the decision tree clearly dominates for the most part, the stable performance of the SSSKMIV algorithm that was observed in Chapter 5 is again clear, since the SSSKMIV algorithm shows improvement over non-segmented models in all but one group. This is not the case with unsupervised segmentation or decision trees.

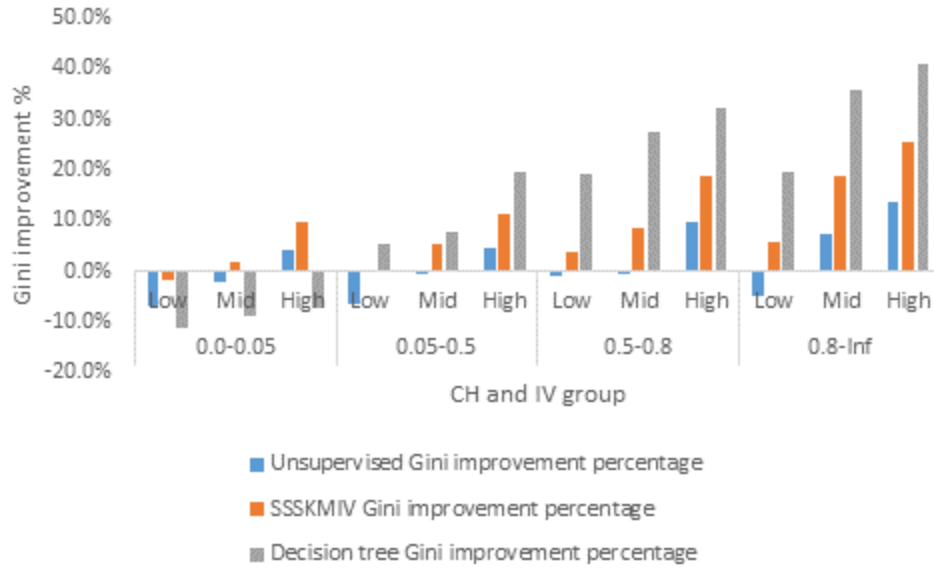


FIGURE 6.18: Gini improvement as a percentage of true improvement obtained by segmentation technique and both CH and IV group for $O = 5$

6.5.2 Results on $O = 15$

We now investigate the impact on the success of the segmentation techniques when we increase the complexity to $O = 15$. This means that the algorithms will have a lot more information available to identify the segments, and we will see that they are therefore more successful.

One of the advantages of the decision tree is that it produces results that are simple to implement and quite transparent (see again Section 2.7.1). However, as we will see, in this case its simplicity is also its downfall. We will see here that, when segments are particularly complex, i.e. requiring many variables and rules to be accurately described, a decision tree is less successful as a segmenter compared to clustering based algorithms. Alternatively put: When used as a segmenter, a decision tree will find simple segments, whilst a clustering based segmenter like k-means or the SSSKMIV algorithm will find more complex segments. Similar observations about decision trees and clustering algorithms (when used for segmentation) are made by many texts (Anderson, 2007, Siddiqi, 2006, Thomas, 2009).

Figures 6.19 and 6.20 as well as Tables C.8 and C.9 show the CH values and IVs in a similar way as was done in the previous section. The true IVs are very similar to those seen in Figure 6.13. As expected, we see slightly higher true CH values for $O = 15$ (see Section 6.3.2), but the differences are small.

A show of things to come is the fact that the SSSKMIV algorithm actually succeeds in finding higher IVs compared to the decision tree algorithm for the higher IV groups. The overall IVs are also closer to true IVs, but the CH values have increased further.

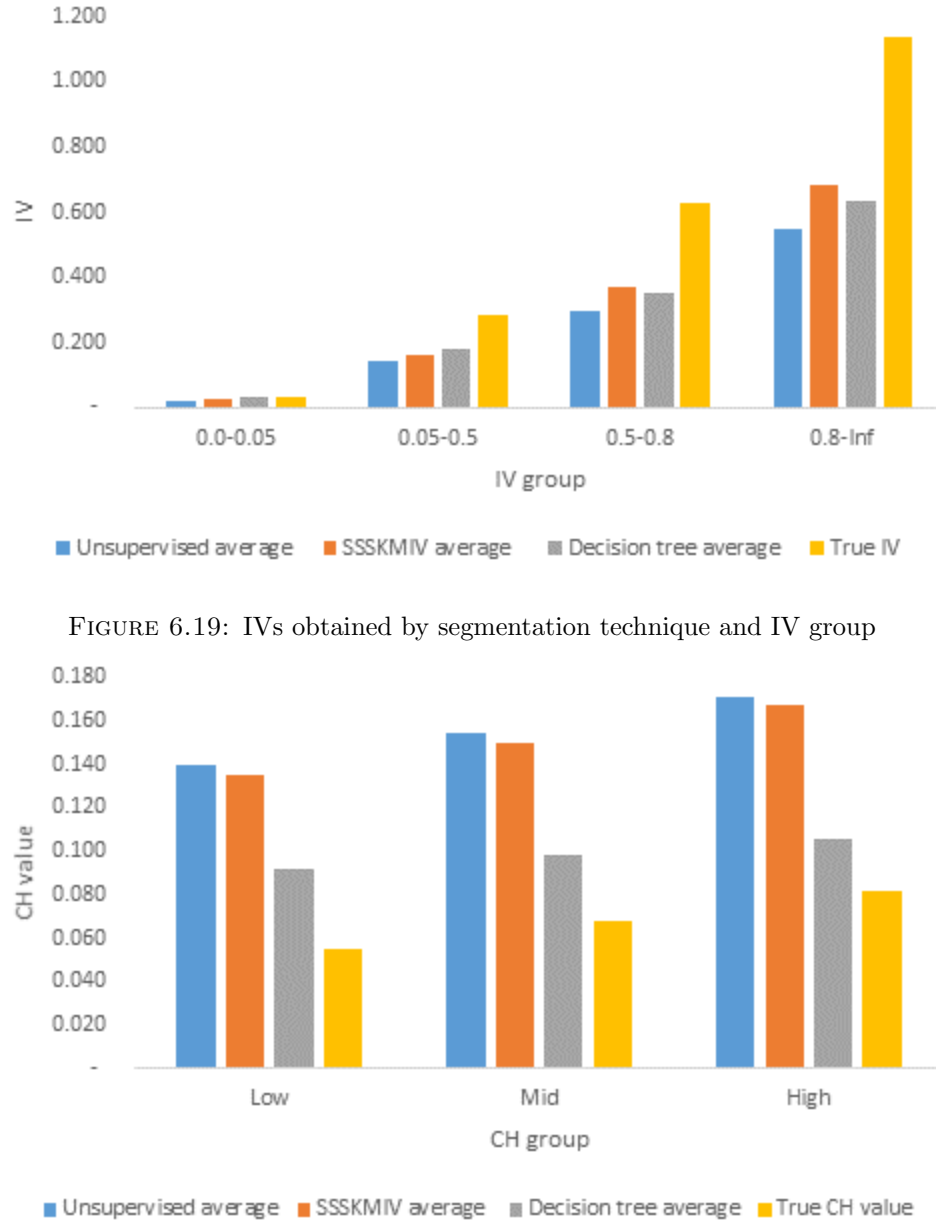


FIGURE 6.20: CH values obtained by segmentation technique and CH group for $O = 15$

A look at Figure 6.21 and Tables C.10 (actual Ginis) and C.12 (Gini improvement percentages) reveals to what degree the SSSKMIV algorithm dominates the other algorithms for $O = 15$. The decision tree algorithm reduces the Gini when IVs are very low.

The SSSKMIV algorithm achieves almost 65% of the true Gini improvement for the highest IV group. We can see here that the unsupervised k-means algorithm comes very

close to the SSSKMIV algorithm for IVs between 0.0 and 0.05. We will explore this a bit more.



FIGURE 6.21: Gini improvement as a percentage of true improvement obtained by segmentation technique and IV group for $O = 15$

Figure 6.22 and Table C.13 show the Gini improvement percentage by CH group. Similar to the results shown by IV, the SSSKMIV algorithm clearly outperforms. The actual Ginis obtained by CH group can be seen in Table C.11.

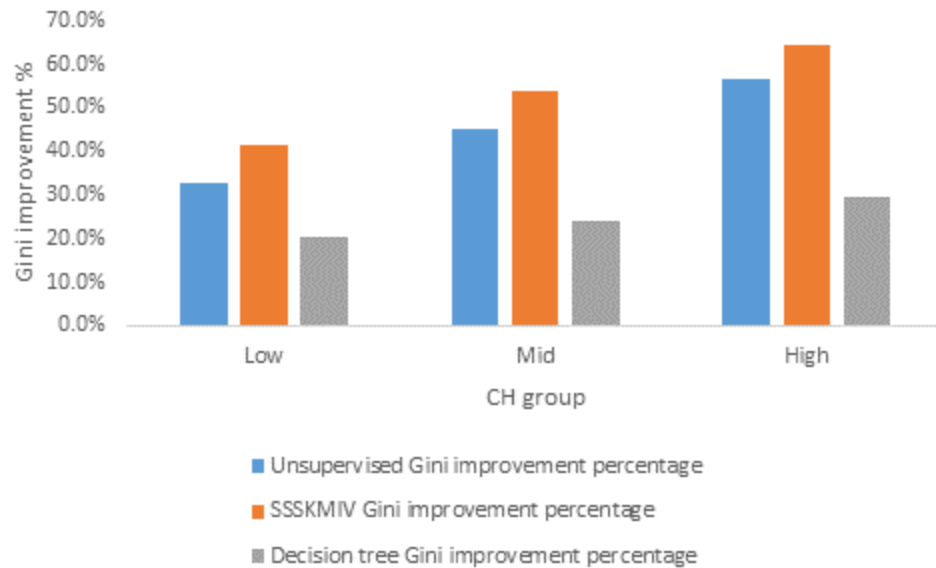


FIGURE 6.22: Gini improvement as a percentage of true improvement obtained by segmentation technique and CH group for $O = 15$

Figure 6.23 and Table C.14 show the results by both CH group and IV group. Here the SSSKMIV algorithm achieves 74% of true Gini improvement on average in the high CH

value, high IV group. The IV group between 0.0 and 0.05, with high CH values, is most closely contested, with the unsupervised k-means algorithm obtaining results very close to the SSSKMIV algorithm.

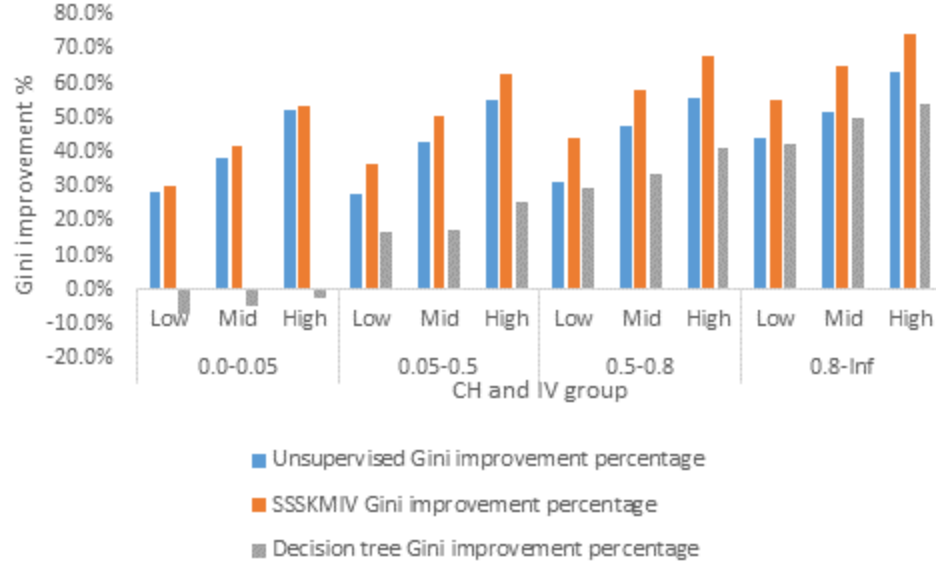


FIGURE 6.23: Gini improvement as a percentage of true improvement obtained by segmentation technique and both CH and IV group for $O = 15$

6.5.3 Results on $O = 10$

Lastly we consider the middle ground, when $O = 10$. This is halfway between the previous two scenarios, where we have clear winners.

Figures 6.24 and 6.25 as well as Tables C.15 and C.16 show the CH values and IVs. Here the decision tree finds higher IV's than SSSKMIV, and the graphs are very similar to those observed with $O = 5$, although both IV and CH values are slightly higher.

Interestingly, Figure 6.26 and Tables C.17 (actual Ginis) and C.19 (Gini improvement percentages) show that the SSSKMIV algorithm obtains the best Gini improvements in all but the last IV group. In the last IV group, the decision tree obtains the best Gini improvement, more than 40% of the true improvement. Also of note is the fact that the unsupervised k-means segmentation technique outperforms the decision tree in both the first two IV groups.

When grouped by CH group, as shown in Figure 6.27 and Table C.20, the SSSKMIV algorithm outperforms the rest for every group. The actual Ginis obtained by CH group can be seen in Table C.18.

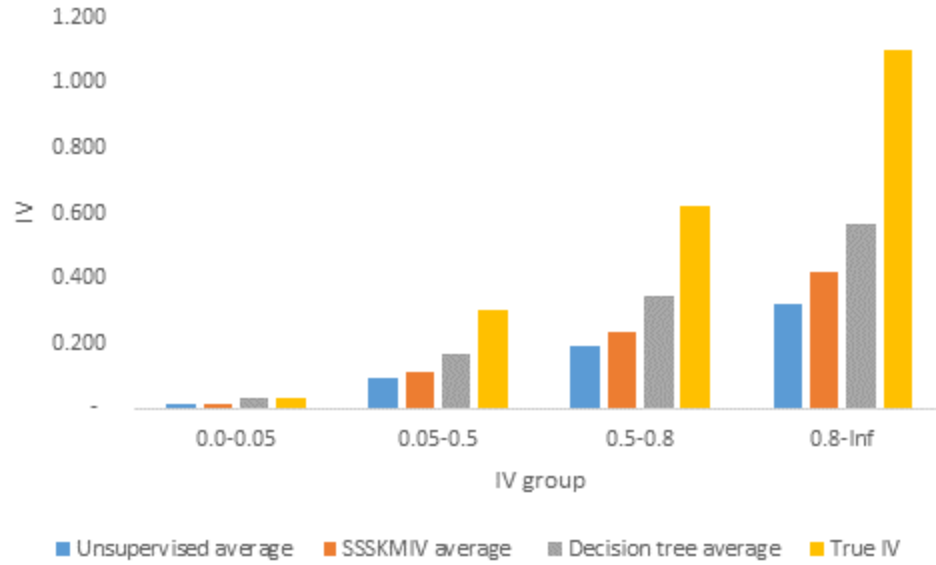
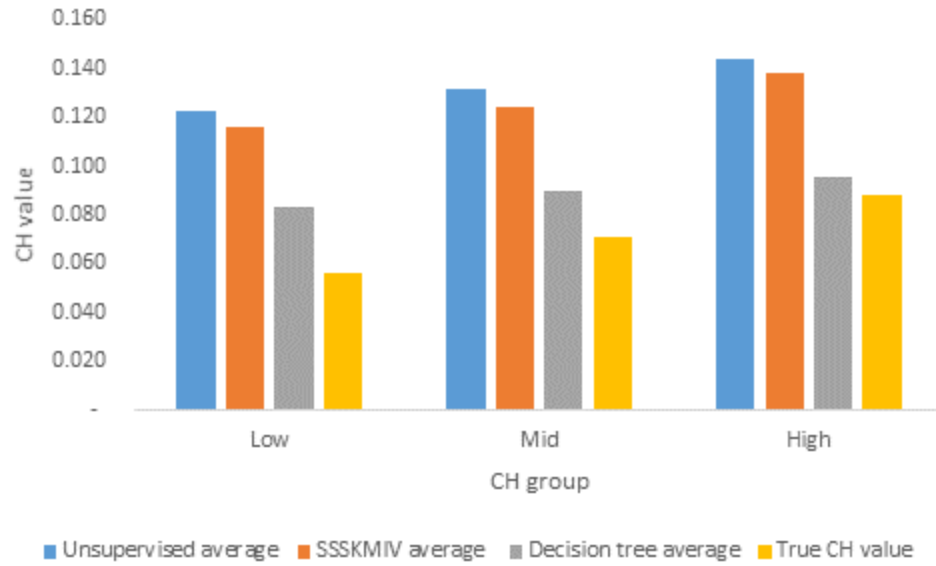


FIGURE 6.24: IVs obtained by segmentation technique and IV group

FIGURE 6.25: CH values obtained by segmentation technique and CH group for $O = 10$

From Figure 6.28 and Table C.21, we see that the SSSKMIV algorithm outperforms the decision tree whenever the CH group is high. This is even true for the highest IV group. When the CH group is low, the decision tree performs better since it uses only target variable information.

6.5.4 Unsupervised segmentation success

In Sections 6.5.1 to 6.5.3, we have found that the complexity of the underlying segments of a data set contributes strongly to whether the decision tree or the SSSKMIV algorithm



FIGURE 6.26: Gini improvement as a percentage of true improvement obtained by segmentation technique and IV group for $O = 10$

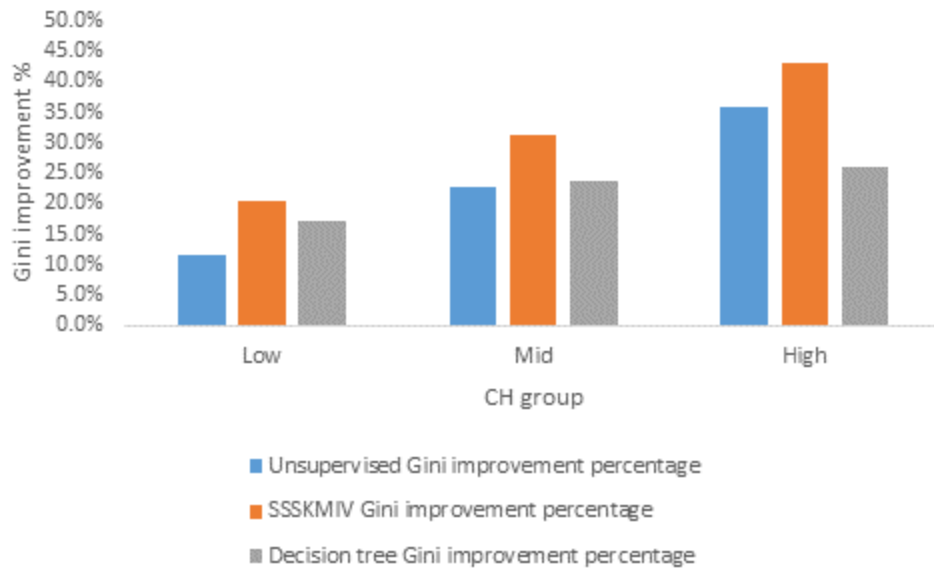


FIGURE 6.27: Gini improvement as a percentage of true improvement obtained by segmentation technique and CH group for $O = 10$

will provide the best result. Because a decision tree finds simple results within data sets, it does not work well when underlying segments are complex. A decision tree also does not function well when segments are not clearly distinguishable by the difference in their event rate.

Throughout the results in this section (in fact, even in Chapter 5) we have not seen the unsupervised k-means algorithm outperforming the other segmentation techniques at any stage. We have seen it comparing well with the decision tree in Section 6.5.2,

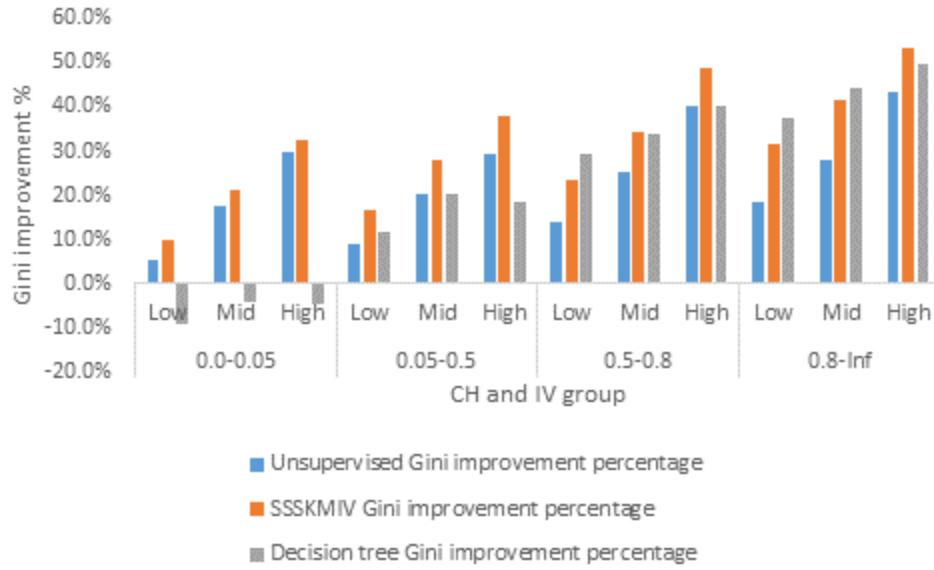


FIGURE 6.28: Gini improvement as a percentage of true improvement obtained by segmentation technique and both CH and IV group for $O = 10$

but never really dominating. One reason may be that the SSSKMIV algorithm retains enough characteristics of the k-means algorithm to ensure it will always perform at least on par with k-means, with the additional benefit of incorporating additional target variable information. However, when IVs are very low, the SSSKMIV's goal of finding segments where event rates differ may mislead it into finding the wrong segments.

To test this theory, we refined our scenario selection criteria by firstly changing the way we simulate the data by sampling event rates from a $U(0.4, 0.6)$ distribution (as opposed to $N(0.5, 0.2)$ as described in Section 6.3.1), and then narrowing our stratification to the IV groups $(0, 0.01]$, $(0.01, 0.02]$, $(0.02, 0.03]$ and $(0.03, 0.05]$. This ensures that a sufficient sample of low IVs are represented. Since we have seen from the preceding analysis that a k-means based methodology performs best when complex clusters are required, we have set $O = 15$.

The resulting Gini improvement percentages are shown in Figure 6.29 and Table C.26. It can be seen that unsupervised segmentation outperforms semi-supervised segmentation for most of the groups. This shows that for low IVs, unsupervised segmentation can be more suited than SSSKMIV. More detail of the results can be seen in Appendix C, Tables C.22 to C.28. Although unsupervised segmentation seems to provide the best results for these scenarios, it does not outperform SSSKMIV by a lot.

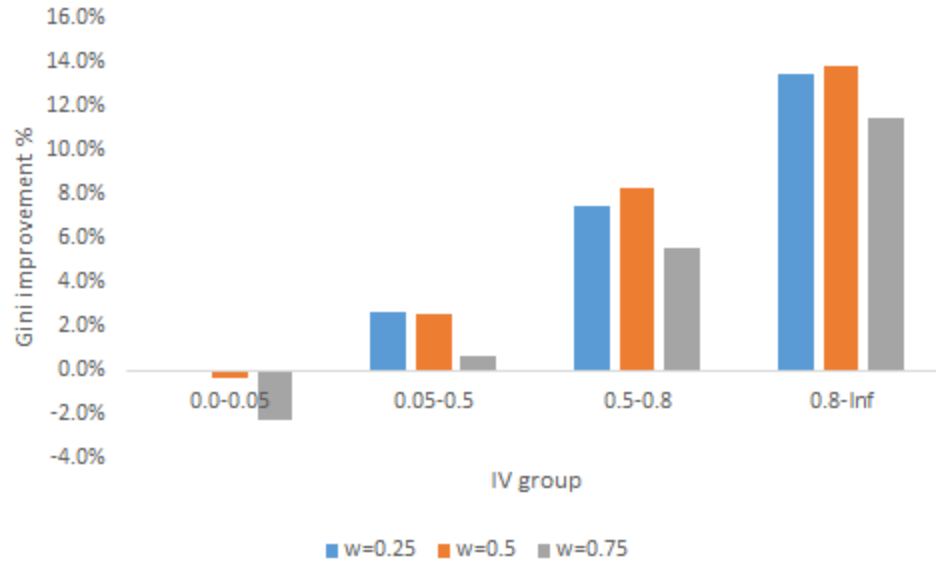


FIGURE 6.29: Gini improvement as a percentage of true improvement obtained by segmentation technique and IV group for low IVs

6.5.5 Value of w

Figure 6.30 is a combined view of all the SSSKMIV iterations performed for $O \in \{5, 10, 15\}$. The Gini improvement percentage is shown by IV group for each distortion weight $w \in \{0.25, 0.5, 0.75\}$. This shows an interesting view, which is not unexpected. The higher the underlying IV, the better higher values of w performs (albeit up to the point where overfit starts to deteriorate performance). We can see that in the first two IV groups, $w = 0.25$ delivers the best results, whilst $w = 0.5$ delivers better results for higher IV groups. The impact of overfit, which was first discussed Section 3.14, is also evident from the results of $w = 0.75$. It can be seen that, on average, $w = 0.75$ underperforms against the lower values for w because of overfit. Analyses have shown that this continues to be true, even for very high IVs greater than two.

From Figure 6.30, it can be seen that there is not a single value of w that always outperforms independent of the data set characteristics. We can therefore not recommend any value of w as the most suited. We rather recommend that the most optimal value of w be determined iteratively on every data set that is segmented using the SSSKMIV algorithm. Figure 6.30 seems to indicate that high values of w will not produce favourable results, but one must note that these are averages across many iterations, and that this does not mean a high value of w will *never* produce the best results.

FIGURE 6.30: Gini improvement percentage by IV group and value of w

	IV group	Low CH group	Mid CH group	High CH group
Complexity (O = 5)	0.0-0.05	SSSKMIV	SSSKMIV	SSSKMIV
	0.05-0.5	SSSKMIV	Tree	Tree
	0.5-0.8	Tree	Tree	Tree
	0.8-Inf	Tree	Tree	Tree
Complexity (O = 10)	0.0-0.05	Tree	SSSKMIV	SSSKMIV
	0.05-0.5	SSSKMIV	SSSKMIV	SSSKMIV
	0.5-0.8	SSSKMIV	Tree	SSSKMIV
	0.8-Inf	SSSKMIV	Tree	Tree
Complexity (O = 15)	0.0-0.05	SSSKMIV	SSSKMIV	SSSKMIV
	0.05-0.5	SSSKMIV	SSSKMIV	SSSKMIV
	0.5-0.8	SSSKMIV	SSSKMIV	SSSKMIV
	0.8-Inf	SSSKMIV	SSSKMIV	SSSKMIV
Complexity (O = 15) Only low IVs	0.0-0.01	SSSKMIV	Clustering	SSSKMIV
	0.01-0.02	Clustering	SSSKMIV	Clustering
	0.02-0.03	Clustering	SSSKMIV	Clustering
	0.03-0.05	SSSKMIV	SSSKMIV	Clustering

TABLE 6.7: Summary of best performing segmentation methodology for all characteristics explored

6.6 Summary of results

A summary of the best performing segmentation methodologies for all the simulations that were explored in Section 6.5 is provided in Table 6.7. The criteria used for best performance is based on the improvement in Gini coefficient achieved. Cases where the SSSKMIV algorithm performed best is highlighted in **bold** to provide some visual contrast in the table.

The results of the analysis performed in this section leads us to a number of conclusions:

- It is possible to simulate data sets for predictive modelling in such a way that known segmentation techniques (like unsupervised clustering, decision trees or semi-supervised segmentation techniques) can be applied to it successfully (i.e. conclusive results can be achieved), and improvement in predictive accuracy can be achieved through segmentation (as measured by improvement in the Gini coefficient).
- The results of segmentation for predictive modelling can be effectively analysed using such simulated data sets. We analysed the impact on CH and IVs as well as the improvement in Gini and found that there are a number of underlying characteristics that will cause one segmentation technique to outperform another. We have also seen that obtaining the best IV or CH values are not necessarily an indicator of which technique will perform best when measured by improvement in Gini.
- Data sets that consist of complex underlying segments, and are described by many variables, are not well suited to decision tree segmentation. The segmentation will be more successful if unsupervised clustering, or SSSKMIV techniques, are used.
- Decision tree segmentation is more successful when segments are simple (i.e. not described by many explanatory variables), and target rates between segments differ a lot (i.e. IVs are high, generally greater than 0.5).
- Unsupervised segmentation, like k-means clustering can outperform both SSSKMIV and decision trees when segments are complex (described by many explanatory variables), but differ very little in event rates (IVs below 0.05).
- The SSSKMIV algorithm consistently performs well within a large range of data set characteristics, and outperforms known techniques in many instances.
- The best value for w for the SSSKMIV algorithm is determined by the underlying characteristics of the data set. Segments that have larger IVs (i.e. differ a lot in event rates) will tend to require larger values for w . We have, however, consistently seen that values of $w > 0.5$ tends to overfit. It is recommended that the best value for w be iteratively determined for each data set.

6.7 Conclusion

In this chapter we have contributed the following:

- We tested the impact of segmentation for predictive modelling on a base-case scenario, where a data set was simulated for the purposes of segmentation for predictive modelling. The data set had 20 explanatory variables and a single, binary variable that was simulated to be dependent on six of the explanatory variables. We used the segmentation techniques applied in Chapter 5 to measure their success compared to the true Gini obtained from the simulation.
- We then expanded on the base-case data set by generating scenarios of data sets. We did so through random sampling of data set characteristics such as segment-event rate, segment-specific explanatory variable distribution and segment complexity. The outcome was a large number of data sets with different characteristics that were investigated to understand what specific characteristics cause one segmentation technique to outperform another. Stratified sampling served to select relevant data sets and to ensure representative samples of data sets, then known characteristics were tested.
- We applied the segmentation techniques discussed in Chapter 5 to the selected simulated data sets (with known characteristics) and analysed what characteristics cause one segmentation technique to outperform another.
- We considered how underlying characteristics (like the true IV) influence what value of w performs well during SSSKMIV segmentation.

Chapter 7

Alternative semi-supervised Segmentation Methodologies

7.1 Introduction

Chapter 5 compares the success of using the SSSKMIV and the SSSKMCSQ algorithms for segmentation to the results obtained by both supervised and unsupervised techniques currently used in industry applications. We found that there is no single segmentation technique that always outperforms, but that results differ depending on the data set characteristics.

In Chapter 6 we explored this phenomenon further and found that segment complexity, differences in segment event rates as well as differences in explanatory variable distributions are a few of the driving characteristics that cause one segmentation technique to outperform another.

In this chapter we will explore two additional semi-supervised segmentation techniques that are developed to address some of the weaknesses of the SSSKM algorithms we have observed in Chapters 5 and 6. We also augment the LK-Means methodology discussed in Sections 2.9 and 2.10, to compare its performance to the results in this study. We apply these techniques to the same data sets that were analysed in Chapter 5 and compare the results with those obtained by the other segmentation techniques.

7.1.1 Research contributions

In this chapter we contribute the following:

- We explain the rationale for developing a semi-supervised algorithm with segment-size equality (SSE).
- We describe the objective function of the $\text{SSSKMIV}_{\text{SSE}}$ algorithm (semi-supervised segmentation applied to the k-means algorithm using IV as distortion measure, with the addition of segment size equality) as well as how it is applied.
- We apply the $\text{SSSKMIV}_{\text{SSE}}$ algorithm to the data sets analysed in Chapter 5, and compare the results with those obtained by the other segmentation techniques where possible.
- We explain the rationale for developing a density based semi-supervised algorithm using Wong's density clustering algorithm.
- We describe the objective function of the SSSWong algorithm (Semi-supervised segmentation applied to Wong's density clustering methodology) and indicate how it connects to the objective function used in the SSSKM algorithms.
- We explain the design and operation of the SSSWong algorithm in five steps and provide more detail on each of the steps.
- We apply the SSSWong algorithm to the data sets analysed in Chapter 5 and compare the results with those obtained by the other segmentation techniques.
- We augment the LK-Means methodology, proposed by Peralta et al. (2013) to make its results comparable to other methodologies used in this study.
- We apply the LK-Means algorithm to the data sets analysed in Chapter 5 and compare the results with those obtained by the other segmentation techniques.

7.2 Adding segment size equality as objective to SSSKMIV

The first alternative methodology that we will consider is the addition of segment size equality to the SSSKMIV algorithm. The primary reason for adding segment size equality to the SSSKMIV objective function is to discourage the algorithm from forming small segments, and rather encourage it to form segments that are similar in size. In Chapter 4 we did point out that the k-means algorithm tends to favour segments that are of similar size (see also (Yan, 2005)), but in some cases the k-means algorithm is prone to identify outlier segments that can be very small (Bradley and Fayyad, 1998, Yoon et al., 2007). Challenges due to small volumes in segments may manifest in a number of ways:

- A number of outlier segments may be identified that are too small to facilitate predictive modelling, which will mean no additional value can be added through predictive models on these segments. This defies the purpose of segmenting for predictive modelling.
- The segmentation algorithm could identify one very large segment (or a very small number of very large segments) and a large number of smaller segments. In such scenarios, even if predictive modelling is possible on the smaller segments (in contrast to the previous point), it will mean that many small models will need to be maintained that are potentially less robust (due to the small volumes), and prone to overfit. In practice it may therefore be more practical to have fewer segments that are more equal in size.
- A developer that prefer many segments (i.e. require a high value for K) may not be able to successfully obtain the number of segments required for predictive modelling, since segment sizes may already be too small for low values of K (and therefore suffer from the challenges discussed above). If segment sizes remain close to equal in size, the developer may be able to find more segments before the two challenges discussed above manifests.

The consequences discussed above are even more applicable when the data set on which segmentation for predictive modelling is performed, is small.

In Chapter 3 we discussed the SSSKM algorithm in some detail, and introduced Equation 3.2 as the objective function of the algorithm. For the addition of segment size equality, we will make only minor adjustments to the objective function of the SSSKM algorithm. For ease of reference, we will refer to the semi-supervised segmentation algorithm using k-means and IV with segment size equality as the SSSKMIV_{SSE} algorithm.

Recall from Equation 3.2 that we defined w as the distortion weight, which is a measure of how much emphasis should be placed on the supervised element of Equation 3.2. For SSSKMIV_{SSE} we introduce ν as the segment size equality weight and ϱ as the segment size equality function. In our application of SSSKMIV_{SSE}, we define ϱ as

$$\varrho(c) = \prod_{j=1}^K \frac{|S_{cj}|}{n_c}. \quad (7.1)$$

In Equation 7.1, n_c is the total number of assigned observations for $c \in \mathbb{C}$ and the terminology $|S|$ is again used to represent the number of observations in segment S . The function is at its maximum when all segments sizes ($|S_{c1}|, \dots, |S_{cK}|$) are equal.

An example of how the value of $\varrho(c)$ is maximised when segment sizes are equal, is given for two segments (i.e. $K = 2$) in Figure 7.1. Notice how the equation reaches its maximum when both segments contain 50% of the population. The x-axis shows simulation number 1 to 11, with each simulation distributing the population as indicated.



FIGURE 7.1: Two segment example of how the segment size equality function maximises when segment sizes are equal

Incorporating ν and $\varrho(c)$ into Equation 3.2 results in a new objective function for $\text{SSSKMIV}_{\text{SSE}}$,

$$\min_{c \in \mathcal{C}} \left[w \varphi(c) + \nu \varrho(c) + (1 - w - \nu) \sum_{j=1}^K \sum_{i \in S_{c_j}} d(S_{c_j}, \mathbf{x}_i) \right], w + \nu \leq 1. \quad (7.2)$$

As discussed, the detail of the algorithm discussed in Sections 3.3 and 3.6 through to 3.15 remains exactly similar for $\text{SSSKMIV}_{\text{SSE}}$. We will therefore not repeat it here, but will rather focus on how it changes the results of SSSKMIV observed in Chapter 5.

We have seen in Chapter 5 that the best Gini results are usually obtained for values of w below 0.5, and that overfit was frequently evident for values of w greater than 0.5 (see again Figures 5.3, 5.7, 5.11, 5.19 and 5.23). With $\text{SSSKMIV}_{\text{SSE}}$ we also have the additional requirement of $w + \nu \leq 1$. We will therefore reduce the number of weights that are considered to 0.1, 0.2 and 0.4, with the same weights being considered for both w and ν . This will therefore mean that the unsupervised element of Equation 7.2 ($1 - w - \nu$) will be weighted at a value of anything between 0.2 and 0.8, providing a well-represented range to analyse. Note that we have observed overfit in most cases in both Chapters 5 and 6 when the unsupervised weight were reduced below 0.5, so the

selected ranges (i.e. 0.2 to 0.8) should provide us with sufficient information to analyse the results thoroughly. The analysis will be repeated 20 times per weight again, and for $K \in \{2, \dots, 10\}$, as was done in Chapter 5.

Recall from the process described in Sections 5.3 to 5.7 that each iteration of segmentation analysis that is performed, segments the data set based on the required number of segments; then fits logistic regression models on each of the segments, and finally compares the Gini impact (on the validation set) to a model fitted with no segmentation. For each value of K , 180 iterations are done ($3 \times 3 \times 20$), since every combination of ν and w need to be tested, and each combination is analysed 20 times. This means that in total 1,620 (180×9) iterations are performed for each data set, and close to 10,000 logistic regression models are fitted. This exercise is time consuming, taking as much as five days to complete for a single data set. The results are provided in a summarised form in the following sections to ensure it can be effectively analysed, and to demonstrate observed trends.

7.3 SSSKMIV_{SSE} results

We will now apply the SSSKMIV_{SSE} algorithm to each of the six data sets used in Chapter 5, and will discuss the results of each data set separately. Because of the addition of ν to the objective function, along with the requirement of $w + \nu \leq 1$, we will not be able to replicate all the graphs discussed in Sections 5.7.1 to 5.7.6 (specifically, the parts of the graphs illustrating information of $w \in \{0.3, 0.5, 0.6, 0.7, 0.8, 0.9\}$), but we will show comparisons where they are relevant and possible. We will again use the direct marketing data set to explain the graphs for all other data sets.

7.3.1 Direct marketing data set results for SSSKMIV_{SSE}

Because the goal of the SSSKMIV_{SSE} algorithm is to find segments that are more similar in size, we first explore how successful the algorithm is in achieving this. Simply adding segment size equality to the objective function as in Equation 7.2, does not guarantee that the algorithm will work and we therefore measure the algorithm's success in obtaining the objective first.

Figure 7.2 and Table D.1 provide the result. On the x-axis, the values for ν that were tested is shown. This includes the results for $\nu = 0$, which was sourced from the results in Chapter 5 where ν was effectively set to 0. The bar graphs are read off the left hand axis and shows the average size (as a percentage of total number of observations) of both

the largest and smallest segments for each value of ν . The impact of an increase in ν is visible from the graph. As the value of ν increases, the average smallest segment size increases, whilst the largest segment size decreases. Note that these are the averages across all values of $K \in \{2, \dots, 10\}$.

The line graph in Figure 7.2 is read of the right-hand axis and is a normalised average value of ϱ . Because all values of K are represented together in this graph, and because the optimal value of ϱ differs depending on the number of segments, the obtained value is divided by the optimal value (i.e. the value of ϱ when all segments are equal) to allow results from all values of $K \in \{2, \dots, 10\}$ to be represented here. A similar conclusion can be made from the normalised value of ϱ . Increasing the value of ν tends to improve the overall equality of the segment sizes. From the graph it may seem that increasing ν to higher values may benefit the segment size equality more, but since we need to also consider the additional supervised weight, w , and since we have seen from Chapter 5 that overfit occurs at values of w greater than 0.5, we have limited ν to values below 0.5 in this analysis. We will also see that this tends to be high enough from the results of the other data sets.

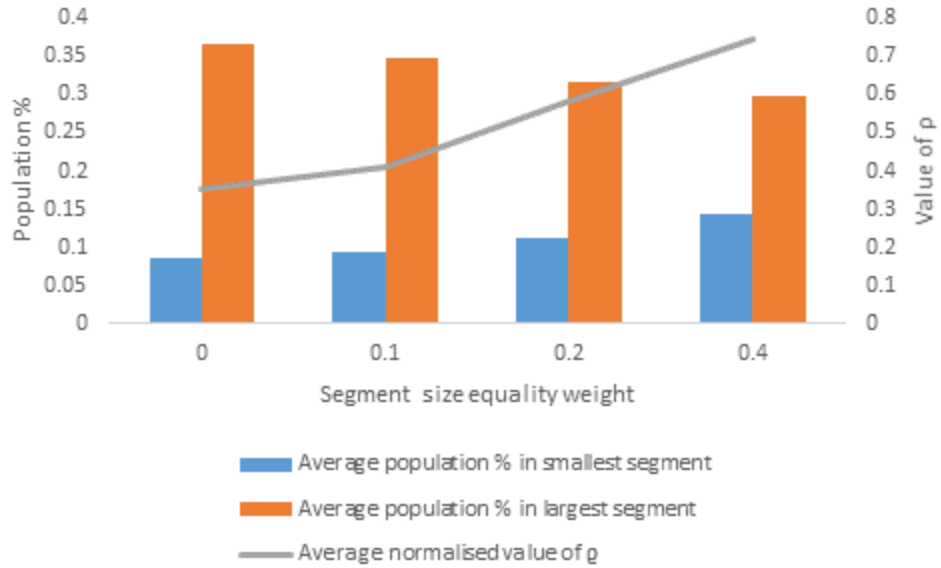


FIGURE 7.2: Impact on population percentages by increasing ν

The next aspect we consider is how the introduction of segment size equality impacts the IVs obtained by the algorithm. Figure 7.3 and Table D.2 show a comparison of the average IVs obtained by the SSSKMIV_{SSE} algorithm for each value of ν and w considered and compare that to the IV results for similar values of w obtained by the algorithms in Chapter 5. It can be seen that the SSSKMIV_{SSE} algorithm succeeds in finding higher IVs at low values of w compared to the other k-means based algorithms. This could be due to the fact that more equal segments will assist with segment assignments that

are statistically more robust (and less likely to overfit), which assists in finding slightly higher IVs. The IVs obtained are, however, still smaller than those obtained by the decision tree.

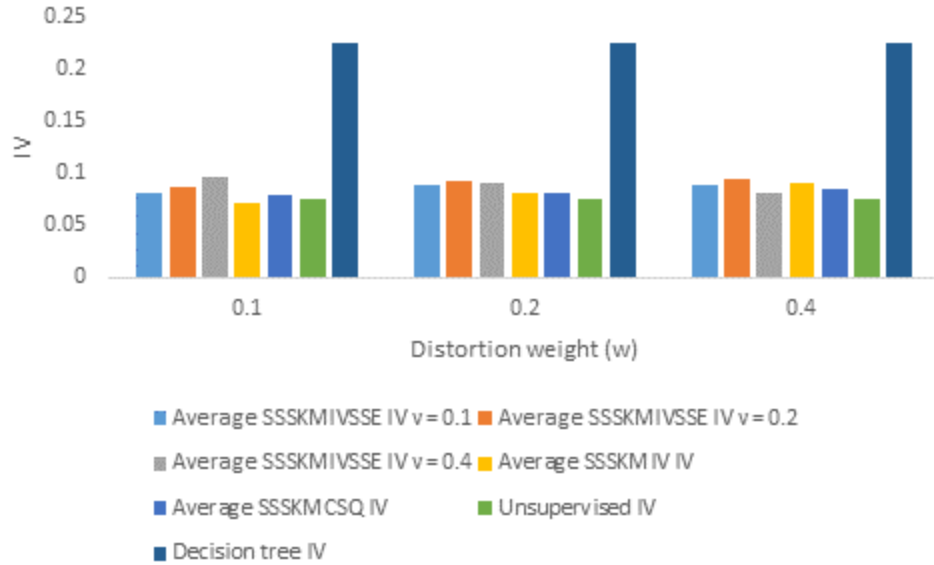


FIGURE 7.3: IVs obtained compared to previous results

In Figure 7.4 and Table D.3 the CH values are shown similarly to how the IV's were shown in Figure 7.3 and Table D.2. Here it is obvious that the CH values are lower when segment size equality is introduced to the SSSKM algorithm.

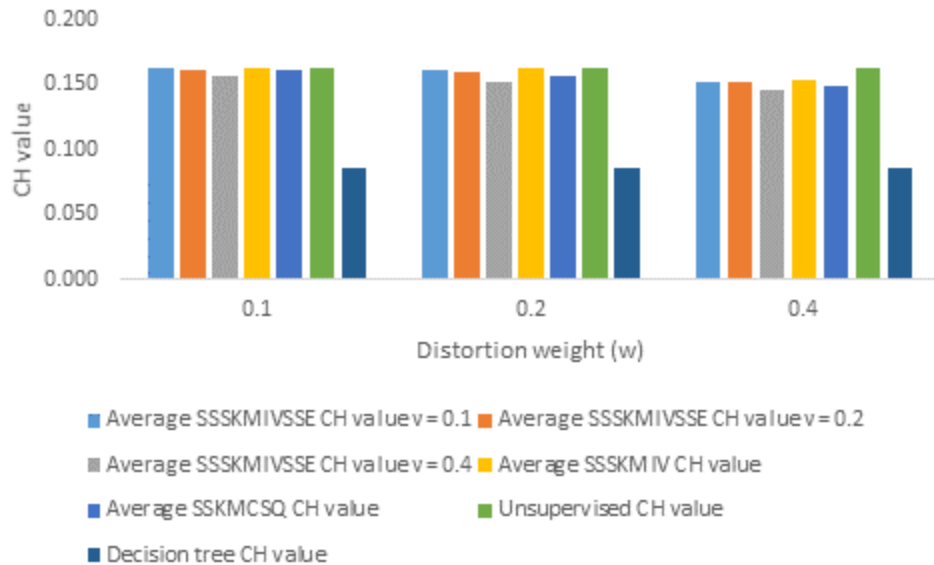


FIGURE 7.4: CH values obtained compared to previous results

We have seen that the $\text{SSSKMIV}_{\text{SSE}}$ algorithm is effective at forming segments that are more equal in size and that this can even cause higher IVs in some cases, but the

main concern for this analysis is whether the algorithm is able to form segments that will result in more accurate models. To gauge this for the direct marketing data set, we present Figure 7.5 and Table D.4. These show the maximum Gini improvement by number of segments for SSSKMIV_{SSE} and compare that to the Ginis obtained for the algorithms investigated in Chapter 5 (see again Figure 5.4 and Table B.12).

The addition of SSE did not improve the accuracy of the segmented predictive models on this data set, but we will see that it does improve accuracy in certain cases on some of the data sets explored in the next sections.

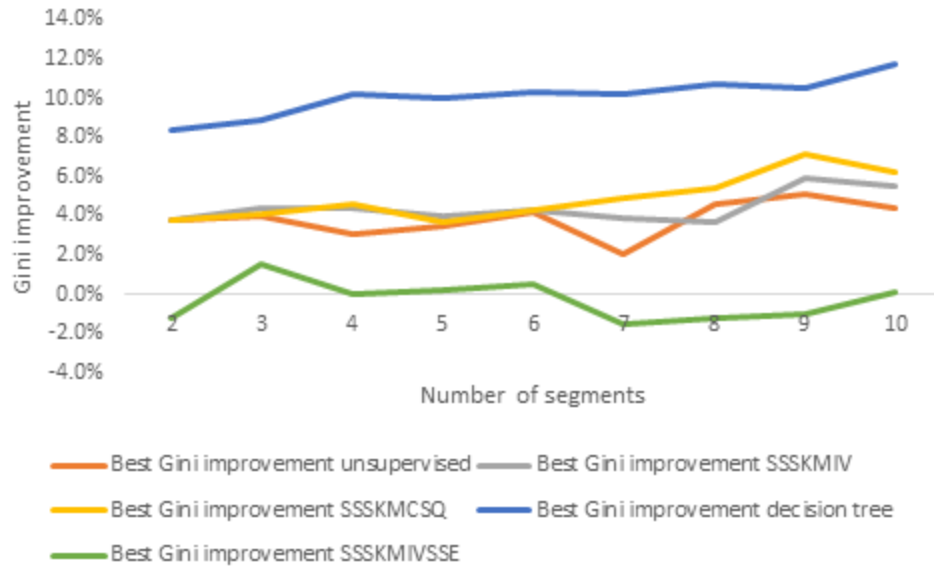


FIGURE 7.5: Gini improvement over unsegmented Gini on the validation set shown by number of segments

It can be seen from Figure 7.5 that the SSSKMIV_{SSE} algorithm fails to impress when measured based on Gini improvement. The algorithm is mostly unsuccessful in even obtaining an increase in Gini, let alone challenge the other algorithms. We have observed in Chapter 5 that, although correlated, improved IVs do not necessarily result in better Ginis, and this is observed here for the SSSKMIV_{SSE} algorithm as well.

In Figure 7.6 and Tables D.5 (Gini improvement) and D.6 (actual Ginis), we see that an increase in ν has an overall positive effect on the Gini improvement for the SSSKMIV_{SSE} algorithm. Surprisingly, as the number of segments increase, the Gini improvement deteriorates even further. This is especially true where values of ν are low.

7.3.2 Protein tertiary structures data set results for SSSKMIV_{SSE}

Figure 7.7 and Table D.7 illustrate the success of the SSSKMIV_{SSE} algorithm in improving segment size equality for the data set on protein tertiary structures. What is

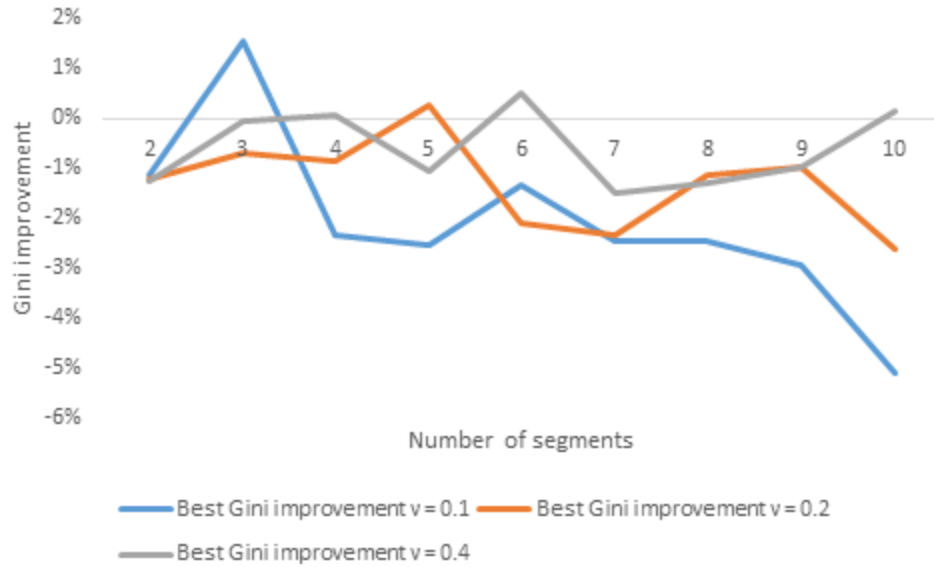


FIGURE 7.6: Gini improvement over unsegmented Gini on the validation set split by value of ν

interesting to see from the graph, is the fact that the average normalised value of ϱ decreases between $\nu = 0.2$ to $\nu = 0.4$. Closer inspection revealed that, similar to what was seen for high values of w in Chapter 5, the SSSKMIV_{SSE} algorithm tends to overfit when the combined values of $w + \nu$ becomes too high. In the case where $\nu = 0.4$, the additional elements in the objective function (i.e. $\varphi(c)$ and $\varrho(c)$) control between 50% and 80% of the objective function, and therefore causes overfit for ϱ . This aligns with what was seen in Chapter 5, where overfit became evident from $w > 0.5$.

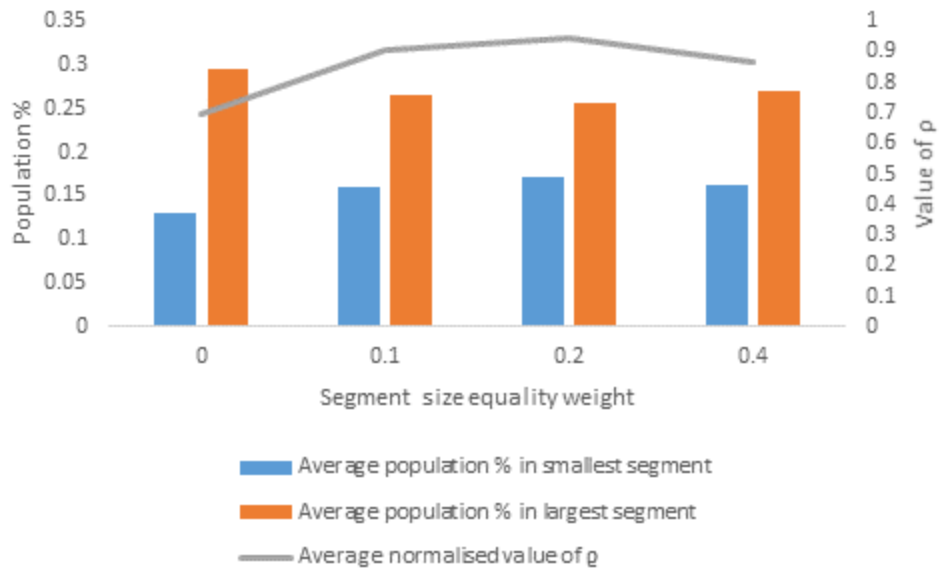


FIGURE 7.7: Impact on population percentages by increasing ν

From Figure 7.8 and Table D.8, we see again that the SSSKMIV_{SSE} algorithm is able to find IVs that are higher than achieved by the SSSKMIV and SSSKMCSQ algorithms for the same values of w . However, it is not able to obtain IVs that are close to the decision tree.

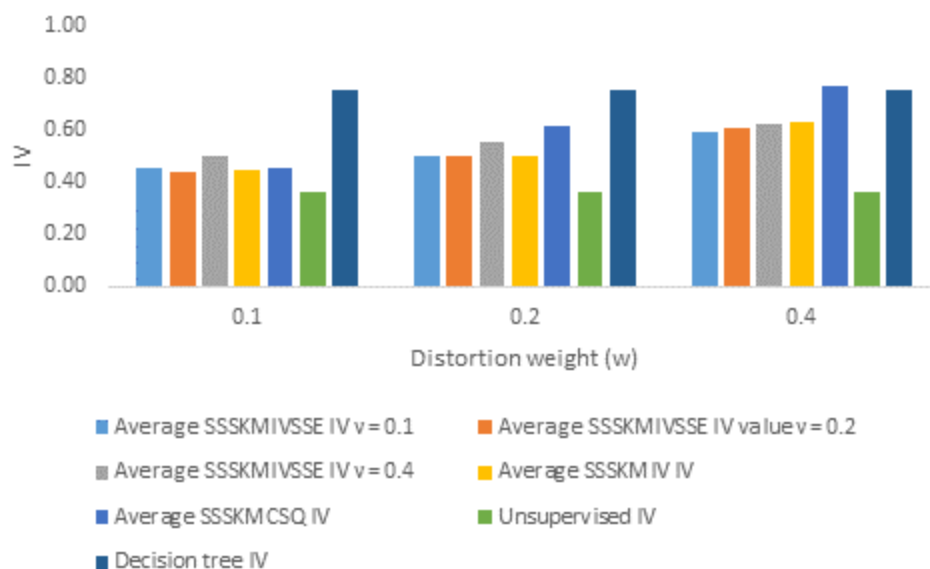


FIGURE 7.8: IVs obtained compared to previous results

In Figure 7.9 and Table D.9, we can observe that the higher IVs do come at the cost of lower CH values.



FIGURE 7.9: CH values obtained compared to previous results

Given the relatively underwhelming impact of the SSSKMIV_{SSE} algorithm observed thus far on the protein tertiary structure data set, Figure 7.10 and Table D.10 are somewhat

surprising. It can be deduced from Figure 7.10 that the $\text{SSSKMIV}_{\text{SSE}}$ algorithm generally produces better Gini's compared to the other segmentation techniques applied to this data set thus far. For four segments, the improvement in Gini is almost double compared to the closest rival, and it continues to outperform barring two values of K ($K = 2, 8$), where SSSKMIV performs best. This shows that the addition of segment size equality can improve the results obtained when segmenting for predictive modelling (in certain data sets).

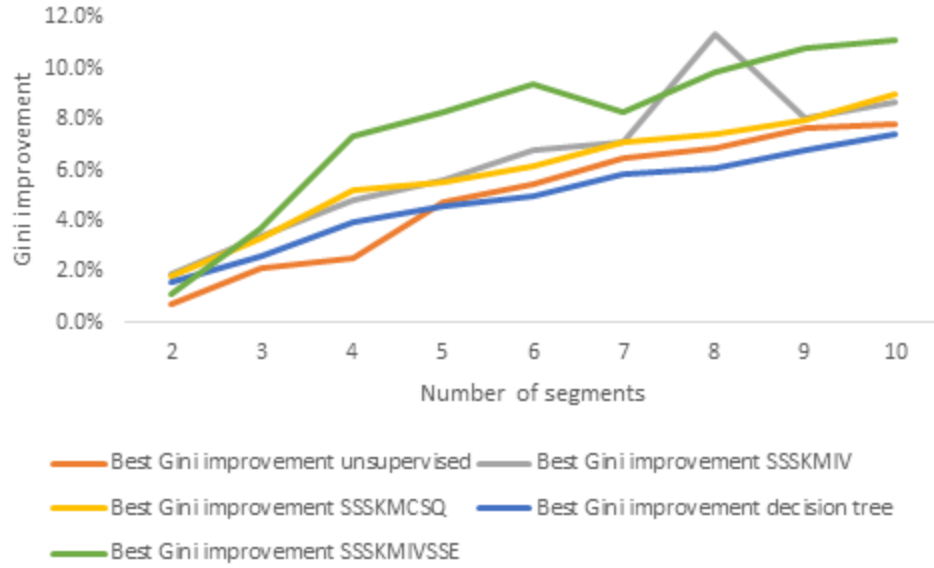


FIGURE 7.10: Gini improvement over unsegmented Gini on the validation set shown by number of segments

Figure 7.11 and Tables D.11 (Gini improvement) and D.12 (actual Ginis) are very different from the results shown in Figure 7.6 in that the Gini improvements are clear and increase as the number of segments increases. It can be seen here that $\nu = 0.2$ obtains the best results for most values of K ($K \in \{5, 6, 7, 9, 10\}$) and that $\nu = 0.4$ outperforms for Segments two to four. $\nu = 0.1$ outperforms marginally when two segments are selected.

7.3.3 Credit application data set results for $\text{SSSKMIV}_{\text{SSE}}$

From Figure 7.12 and Table D.13, it can be seen that the addition of segment size equality has a large impact on the sizes of segments formed for the credit application data set. The average minimum segment increases by more than 7%, whilst the largest average segment size decreases by more than 11% between $\nu = 0$ and $\nu = 0.4$. Clearly overfit is not yet observed on ϱ for $\nu = 0.4$ since the segment size equality continues to increase and does not decrease while ν is increasing as observed in Figure 7.7.



FIGURE 7.11: Gini improvement over unsegmented Gini on the validation set split by value of ν

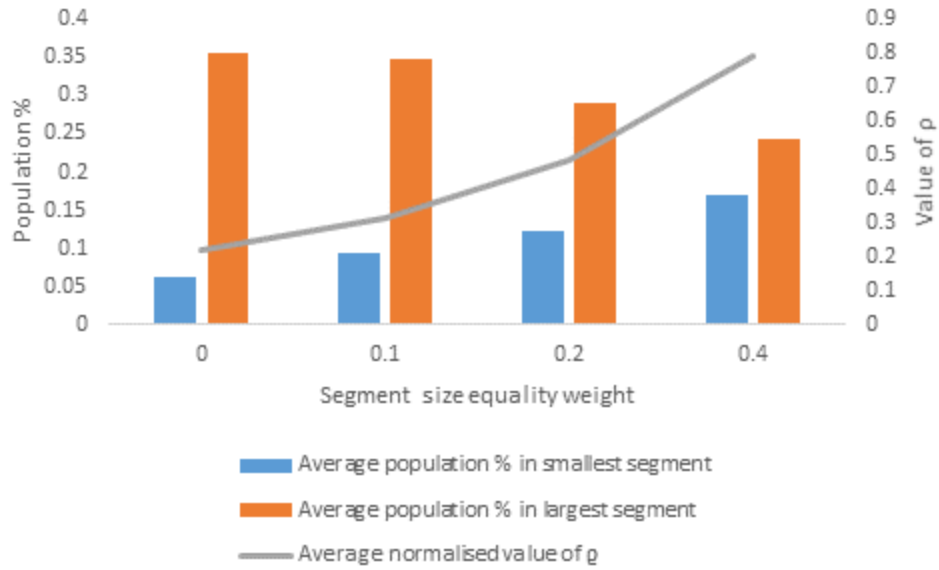


FIGURE 7.12: Impact on population percentages by increasing ν

Once again we observe from Figure 7.13 and Table D.14 that the $\text{SSSKMIV}_{\text{SSE}}$ algorithm obtains better IVs compared to the other SSSKM algorithms (at least for lower values of w), and upon inspecting Figure 7.14 and Table D.15 it achieves worse CH values. They are still, however very far behind obtaining IVs even close to that of the decision tree.

Figure 7.15 and Table D.16 reveal that the $\text{SSSKMIV}_{\text{SSE}}$ algorithm competes well with the other SSSKM algorithms in improving model accuracy, however, the decision tree still outperforms convincingly.

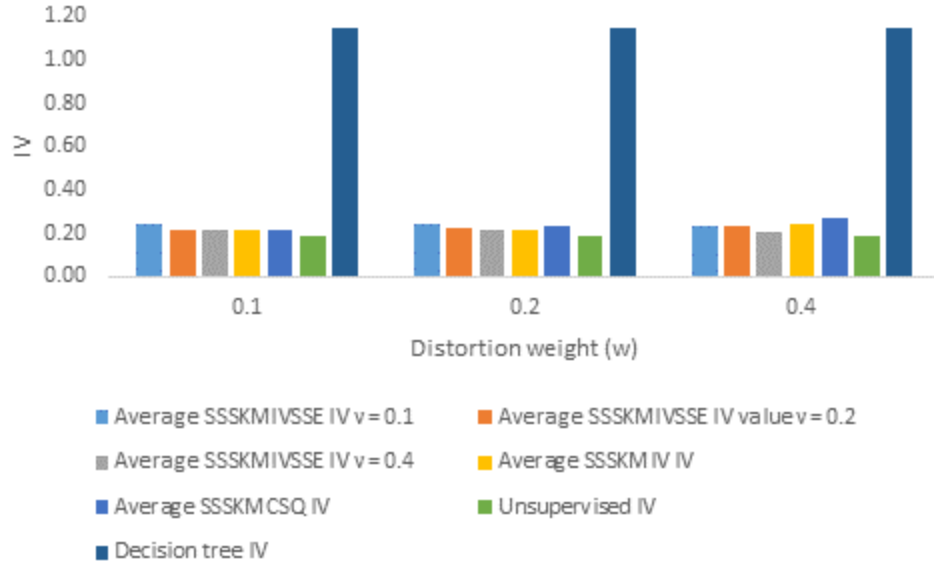


FIGURE 7.13: IVs obtained compared to previous results

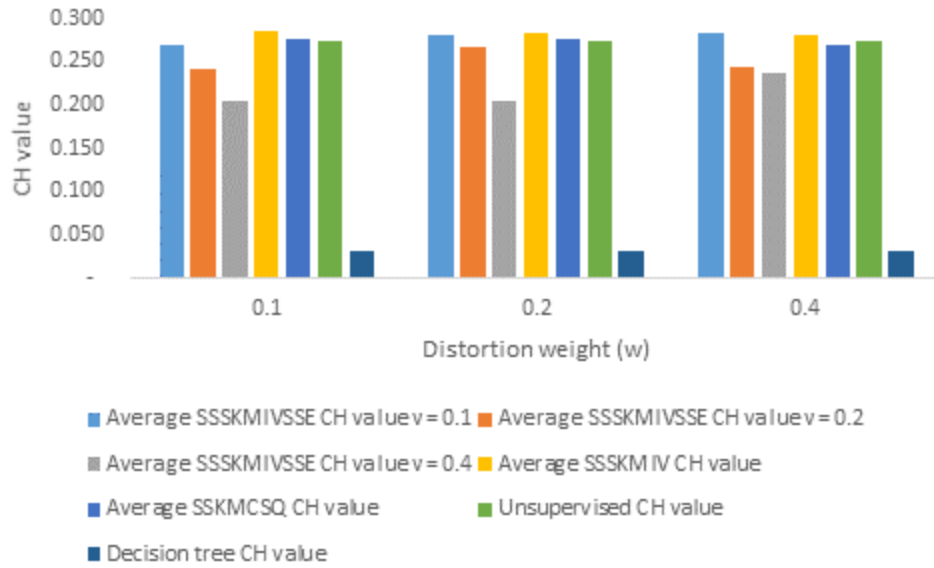


FIGURE 7.14: CH values obtained compared to previous results

Viewing the results by value of ν from Figure 7.16 and Tables D.17 (Gini improvement) and D.18 (actual Ginis) show that low values of ν are more effective when only a few (less than 5) segments are required, but that $\nu = 0.2$ performs better for $K > 5$. Slightly erratic performance can be seen here for $\nu = 0.4$ which can be attributed to overfit.

7.3.4 Wine quality data set results for $\text{SSSKMIV}_{\text{SSE}}$

Similar to the data set on protein tertiary structures discussed in Section 7.3.4, we observe from Figure 7.17 and Table D.19 that the value of ϱ is not guaranteed to increase

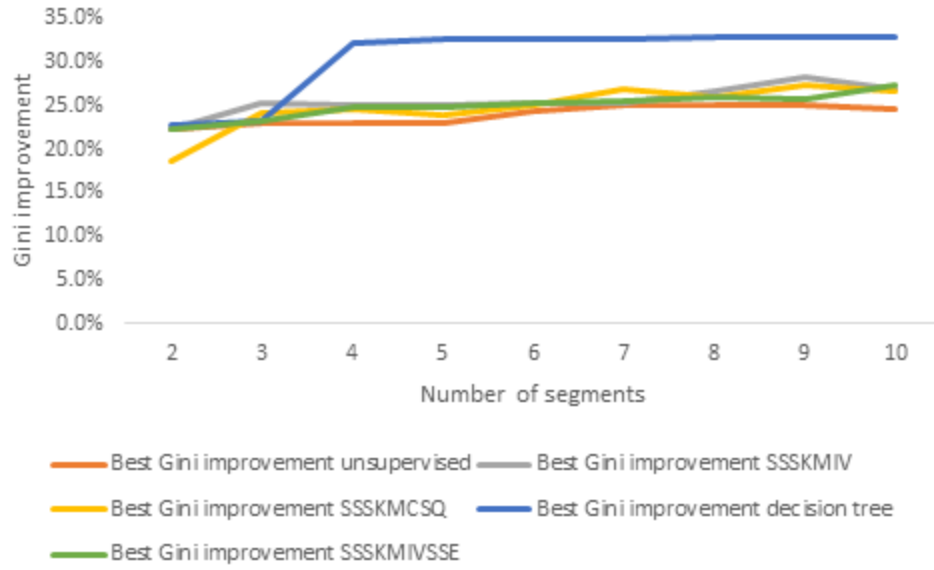


FIGURE 7.15: Gini improvement over unsegmented Gini on the validation set shown by number of segments

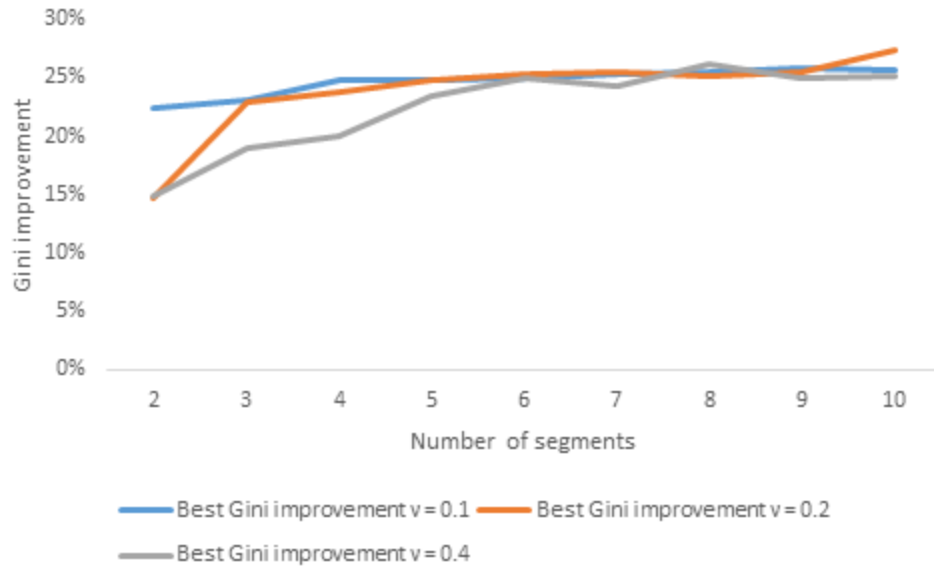


FIGURE 7.16: Gini improvement over unsegmented Gini on the validation set split by value of ν

with increased values of ν . Again we see that, on the wine quality data set, the value of ϱ peaks at $\nu = 0.2$. Although we observe a slight increase in ϱ between $\nu = 0$ and $\nu = 0.2$ this seems to be driven more by a decrease in average maximum segment size as opposed to an increase in the average minimum segment size.

From Figure 7.18 and Table D.20 as well as Figure 7.19 and Table D.21, we see that the SSSKMIV_{SSE} algorithm obtains better IVs than the SSSKMIV algorithm, but not better than SSSKMCSQ. In addition to this, it succeeds in finding CH values that are

FIGURE 7.17: Impact on population percentages by increasing ν

better than SSSKMCSQ, but roughly equal to SSSKMIV. The decision tree finds the highest IVs for this data set.

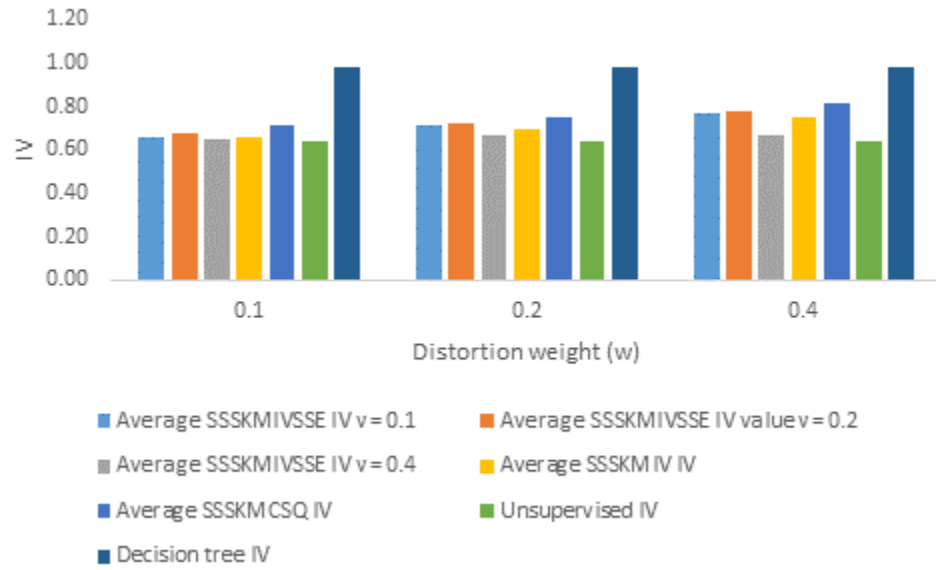


FIGURE 7.18: IVs obtained compared to previous results

The SSSKMIV_{SSE} algorithm performs on par when compared to other segmentation algorithms as seen in Figure 7.20 and Table D.22. This is especially true from four segments and higher, with the algorithm performing best on this data set when $K = 6$. The SSSKMIV algorithm performs best for many values of K , and obtains the best result for $K \in \{3, 4, 7, 8\}$.

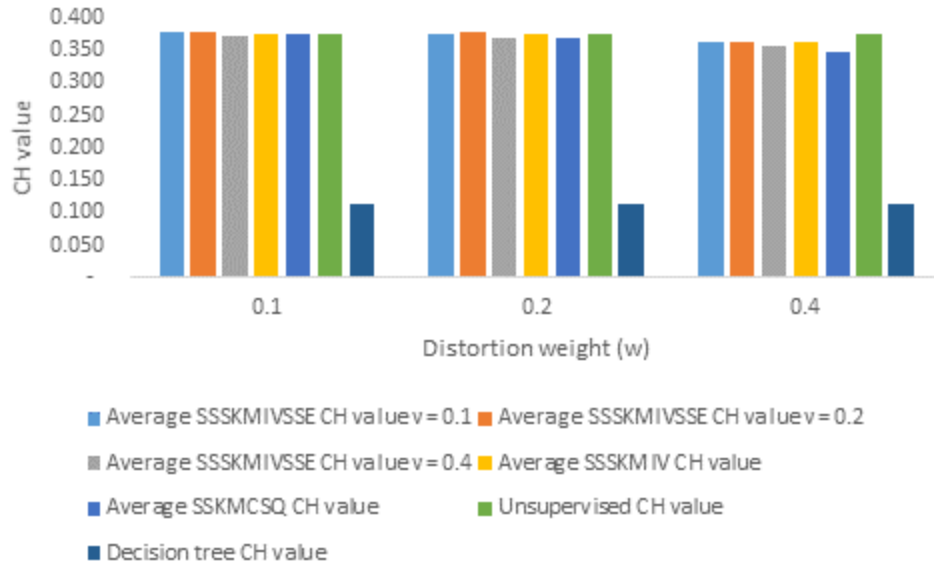


FIGURE 7.19: CH values obtained compared to previous results

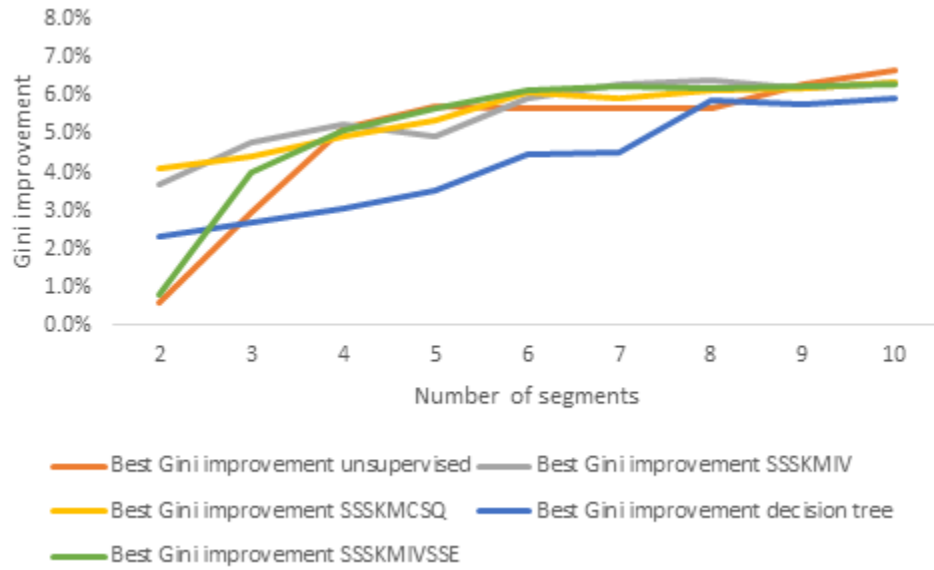


FIGURE 7.20: Gini improvement over unsegmented Gini on the validation set shown by number of segments

Interestingly, from Figure 7.21 and Tables D.23 (Gini improvement) and D.24 (actual Ginis), we observe that $\nu = 0.4$ seems to perform quite well across most values of K , even though we observed some overfit for ϱ in Figure 7.17.

7.3.5 Chess (king-rook vs. king) data set results for SSSKMIV_{SSE}

For the Chess data set we observe from Figure 7.22 and Table D.25 that an increase in ν has no distinguishable impact on the size of the smallest and largest segments.



FIGURE 7.21: Gini improvement over unsegmented Gini on the validation set split by value of ν

In addition to this, the normalised value of ϱ seems to be almost unaffected as well. One reason for this could be the fact that, at $\nu = 0$, the average normalised value of ϱ is already above 90%, the average minimum segment size is just over 18%, and the maximum segment size is just over 24%. Clearly segment sizes are already very similar, and for that reason, increasing the value of ν has little additional impact.

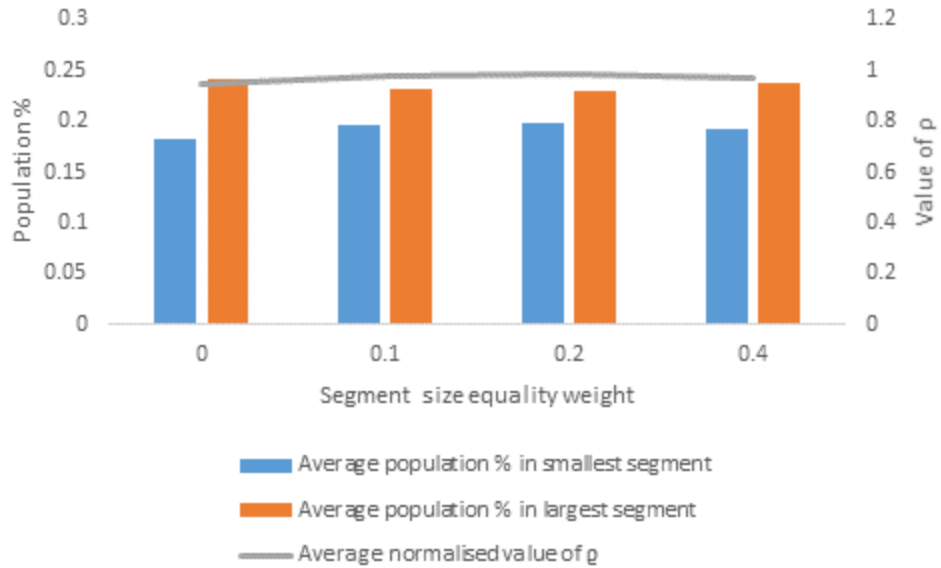


FIGURE 7.22: Impact on population percentages by increasing ν

From Figure 7.23 and Table D.26, we can see that the $\text{SSSKMIV}_{\text{SSE}}$ algorithm obtains better IVs than the other SSSKM algorithms for $w \in \{0.1, 0.2\}$, but that SSSKMCSQ

obtains higher IVs for $w = 0.4$. The $\text{SSSKMIV}_{\text{SSE}}$ algorithm outperforms SSSKMIV for all three scenarios observed when measured by target rate separation.

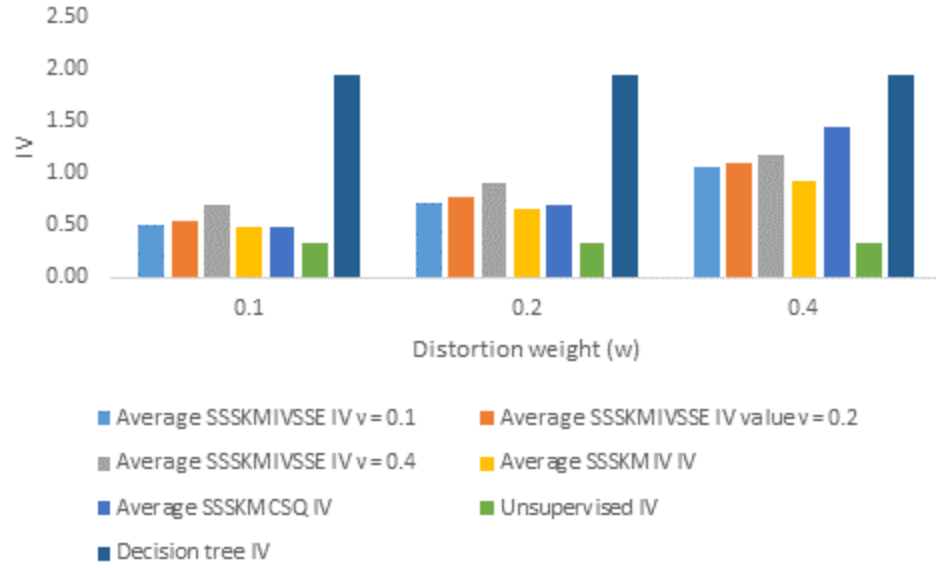


FIGURE 7.23: IVs obtained compared to previous results

It can be seen from Figure 7.24 and Table D.27 that the $\text{SSSKMIV}_{\text{SSE}}$ algorithm again obtains similar CH values when compared to the other SSSKM algorithms, and even when compared to unsupervised segmentation for low values of ν . It performs worse when $\nu = 0.4$, if measured on CH value.

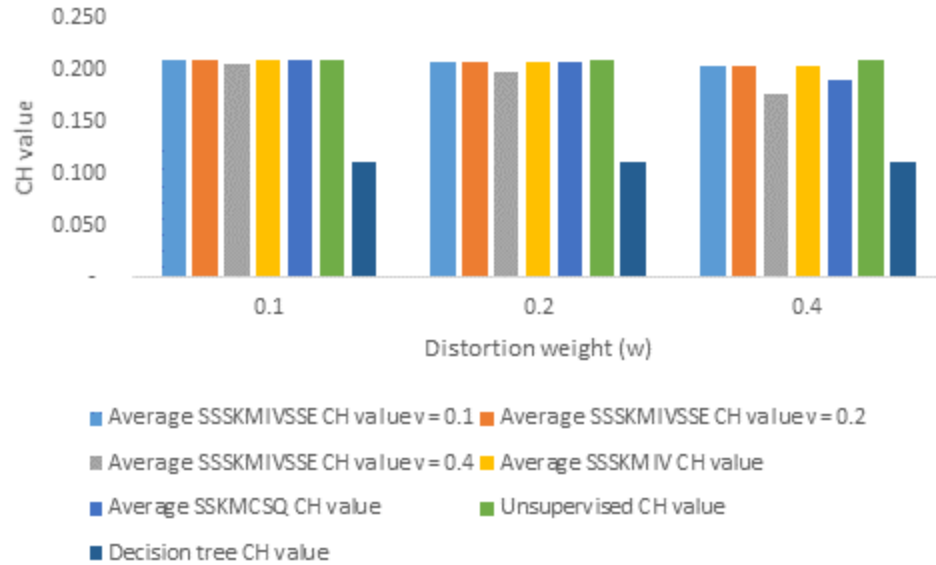


FIGURE 7.24: CH values obtained compared to previous results

We observe from Figure 7.25 and Table D.28 that the $\text{SSSKMIV}_{\text{SSE}}$ algorithm performs in line with all the other k-means based algorithms when measured on Gini improvement.

From $K = 4$, the decision tree still outperforms convincingly though.

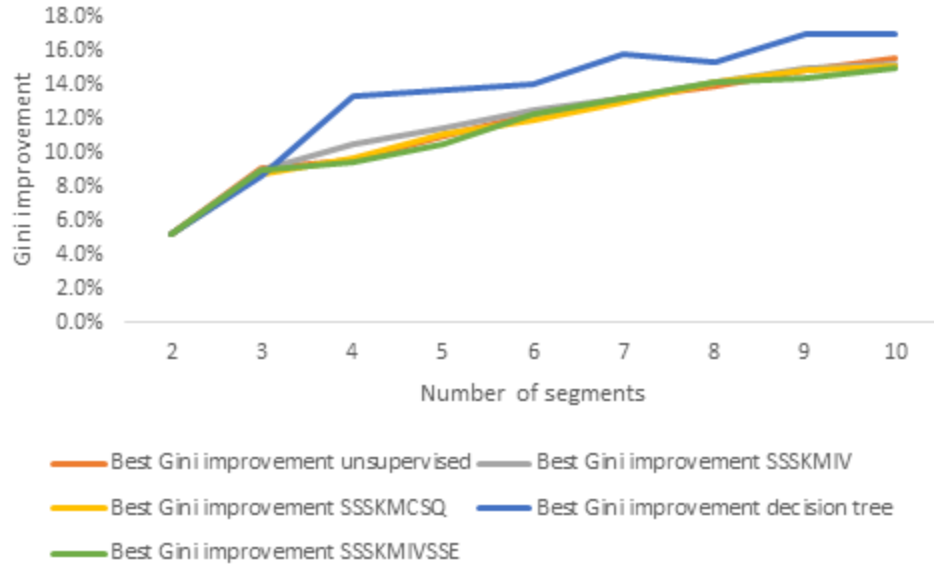


FIGURE 7.25: Gini improvement over unsegmented Gini on the validation set shown by number of segments

In Figure 7.26 and Tables D.29 (Gini improvement) and D.30 (actual Ginis), we see that, as with many of the previous data sets, the $\text{SSSKMIV}_{\text{SSE}}$ performs best when $\nu = 0.2$ for most values of K .

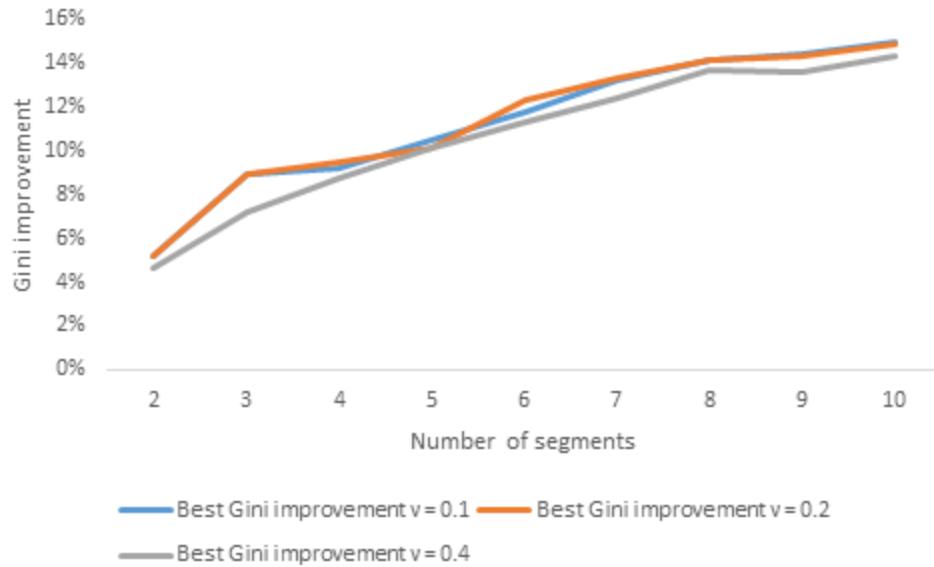


FIGURE 7.26: Gini improvement over unsegmented Gini on the validation set split by value of ν

7.3.6 Insurance claim prediction data set results for SSSKMIV_{SSE}

The impact of ρ on segment size equality for the last data set, which is the claim prediction data set, can be seen in Figure 7.27 and Table D.31. Again we see overfit when $\nu = 0.4$, since the segment size equality decreases even though ν increases from 0.2 to 0.4. This is clearly a common occurrence as it is the third data set where this is clearly observed. In fact, close scrutiny of Figure 7.22 and Table D.25 will reveal such a trend for the chess data set as well, which means that four out of the six data sets analysed in this study show overfit for values of $\nu > 0.2$.

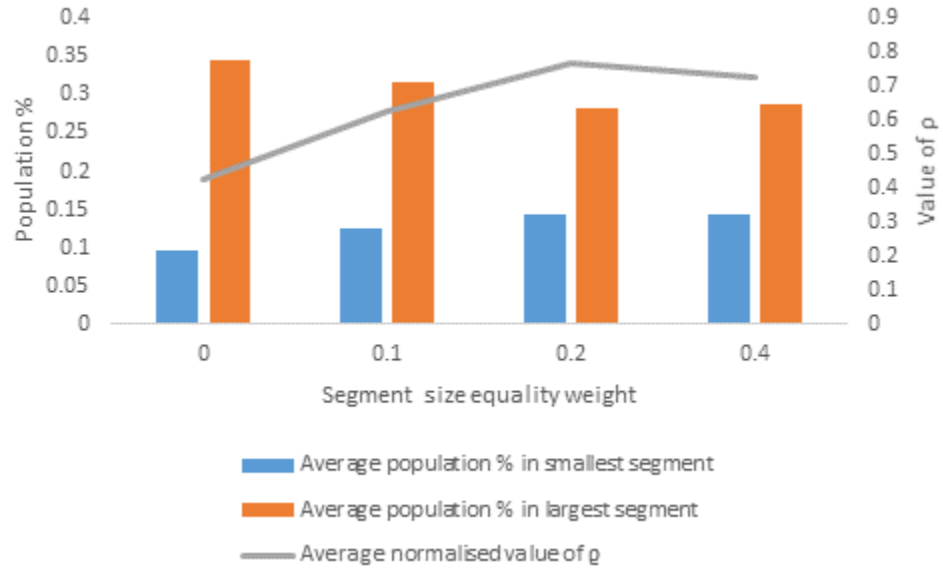


FIGURE 7.27: Impact on population percentages by increasing ν

Again we see from Figure 7.28 and Table D.32, that the SSSKMIV_{SSE} algorithm obtains better IVs compared to the other SSSKM algorithms, especially when w is low. This is also quite a common trend that we have observed for most of the data sets and, as discussed, is most likely caused by an improved statistical robustness when segment sizes are more equally sized.

Similar to previous data sets, we also observe that the SSSKMIV_{SSE} algorithm compares well when measured based on the CH values of the segments, as can be seen from Figure 7.29 and Table D.33.

In Figure 7.30 and Table D.34, we can see that the SSSKMIV_{SSE} algorithm actually performs well when measured on Gini improvement on this data set. It performs best for $K \in \{5, 6, 10\}$, and therefore obtains the best Gini improvement of all methodologies at $K = 10$, albeit with a very small margin.

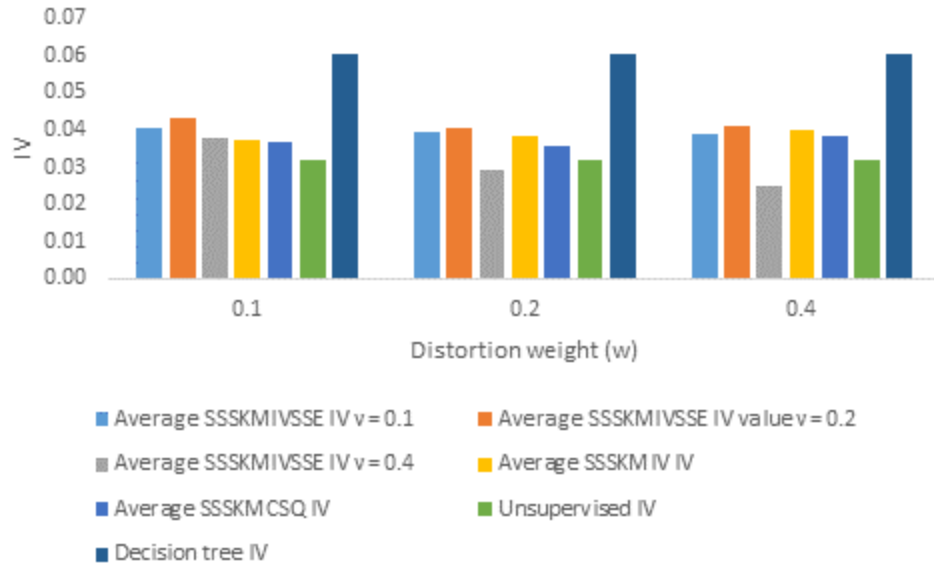


FIGURE 7.28: IVs obtained compared to previous results

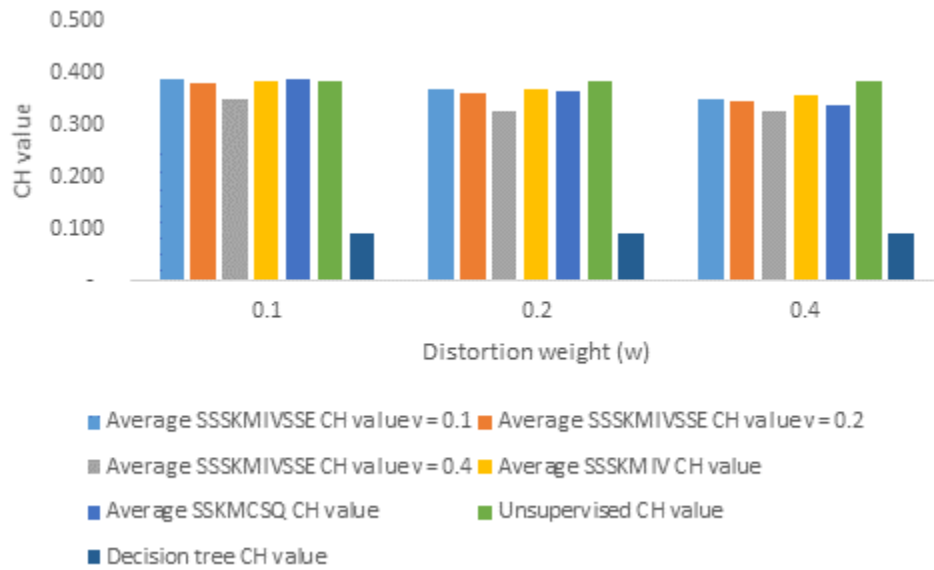


FIGURE 7.29: CH values obtained compared to previous results

We have seen in a number of occasions that the addition of segment size equality to the observation function improves the accuracy of segmentation techniques for certain values of K .

In Figure 7.31 and Tables D.35 (Gini improvement) and D.36 (actual Ginis), we see that the SSSKMIV_{SSE} algorithm obtains the best values with $\nu = 0.1$ and that mixed results are obtained for higher values of ν .



FIGURE 7.30: Gini improvement over unsegmented Gini on the validation set shown by number of segments

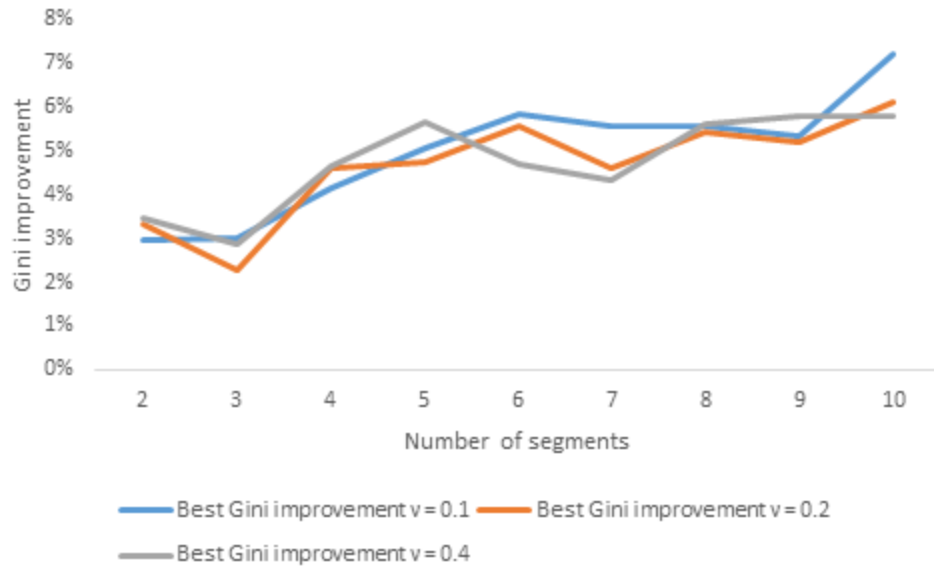


FIGURE 7.31: Gini improvement over unsegmented Gini on the validation set split by value of ν

7.4 Summary of SSSKMIV_{SSE} results

As discussed in the introduction of the SSSKMIV_{SSE} algorithm, the overarching reason for adding segment size equality to the SSSKMIV algorithm is to encourage segment sizes to be more equal in size, which may translate into more robust predictive models, especially where the size of data sets are small or where many segments are required.

We have applied the $\text{SSSKMIV}_{\text{SSE}}$ algorithm to all the same data sets that were analysed in Chapter 5, and we have compared the results where possible. We noted in Section 7.3.1 that simply adding SSE to the objective function does not guarantee its success, and success in obtaining SSE should therefore be measured. From the results, we have learned that the $\text{SSSKMIV}_{\text{SSE}}$ algorithm is able to improve segment size equality for most data sets (as measured by Equation 7.1). We have, however, seen that values of ν that are too high can also result in overfit (similar to high values of w), which causes segment size equality to decrease as compared to lower ν values.

We have learned from the results that the $\text{SSSKMIV}_{\text{SSE}}$ algorithm often finds higher IVs for low values of w when compared to the other SSSKM algorithms. We suggested that this may be due to the increased statistical robustness of the segments, which reduces overfit for the target variable.

Although higher IVs were often found, it sometimes came at the expense of lower CH values. There were, however, some cases where IVs improved without a noticeable decrease in CH values. Note that further work can be done to test this conclusion through statistical confidence intervals, which will show whether the differences are statistically significant.

When comparing the success of the $\text{SSSKMIV}_{\text{SSE}}$ algorithm as measured by the improvement in validation set Gini, we found that there are some instances where the algorithm outperforms all other segmentation algorithms considered. This was especially true for the data set on protein tertiary structures, where the algorithm clearly outperformed for most segments greater than three, with the exception of $K = 8$. For most data sets, the algorithm performed similarly to other SSSKM algorithms, which suggests that the addition of ϱ to the objective function seldom results in worse results, as long as the value of ν is not too high.

We have seen that, for $\nu > 0.2$, the algorithm often fails to further improve segment size equality, and that this often also means that it is less successful in improving validation set Ginis. Lower values of ν were most frequently more successful in obtaining higher Ginis compared to $\nu = 0.4$.

7.5 Semi-supervised segmentation through density clustering

In Sections 2.6.2 and 2.6.3, we discussed the advantages of density based clustering algorithms. Probably the most obvious advantage of density based clustering over k-means is the fact that clusters can take on any form (Hartigan, 1975). This can be

particularly useful if the underlying segments that are investigated are not spherical, since k-means finds globular clusters in m dimensions (Riesen and Bunke, 2008, Wu, 2008).

A clear disadvantage of the SSSKM algorithms investigated in this study is its tendency to overfit for high values of w (or $w + \nu$ in the case of SSSKMIV_{SSE}) (Breed et al., 2013). In this section, we introduce another semi-supervised segmentation technique that is not based on the k-means clustering algorithm, but on the more complex density clustering, specifically Wong's density based clustering algorithm (discussed in Section 2.6.3). Because Wong's algorithm is conglomerative, but starts with clusters already formed, we will see that it reduces the impact of overfit as observations are not considered in isolation. We will call the semi-supervised segmentation algorithm using Wong's method SSSWong in this study.

7.6 Objective function

As described in Section 2.6.3, at a high level, Wong's method consists of two steps. The first step is using k-means clustering to form a large number of preliminary segments (say q segments), whilst the second step combine these segments using Wong's density measure (provided in Equation 2.13).

Our main focus will be to adjust the second step of Wong's method to incorporate the target variable. This means that the main goal of the algorithm is to combine the correct segments in order to optimise both cluster density and differences in target rates.

Similar to Section 3.2 for SSSKM, let $c \in \mathbb{C}$ denote an index of an assignment of all the preliminary segments $s_{c1}, s_{c2}, \dots, s_{cq}$ to the final segments $S_{c1}, S_{c2}, \dots, S_{cK}$, with $K < q$ and with $\mathbb{C}$ the set of all combinations of possible assignments. In this case, q is used to denote the number of preliminary segments formed. Note that each preliminary segment will contain at least one observation, but is likely to contain a number of observations which allows the algorithm to decrease computational complexity on large data sets.

The conglomeration of the preliminary segments into the final set of segments is done in a binary fashion as illustrated by the tree in Figure 7.32.

The final segments for this example of the tree are $S_1 = \{S'_1, S'_2\} = \{s_1, s_2, s_3, s_4\}$ and $S_2 = \{S'_3, S'_4\} = \{s_5, s_6, s_7, s_8\}$. This is an example of only one combination of assignments. In order to enumerate all possible combinations, we use the notation S_{cj} to represent the set of segments assigned to it for a given combination $c \in \mathbb{C}$. In order to evaluate the density dissimilarity between two segments or nodes that will potentially

be joined to form a new node in the tree for a given combination $c \in \mathbb{C}$, we make use of the notation $d(S_{cj})$. For example, to calculate the dissimilarity between nodes S'_1 and S'_2 by using Equation 2.13, we calculate $d(S'_{c1}) = d(s_1, s_2)$.

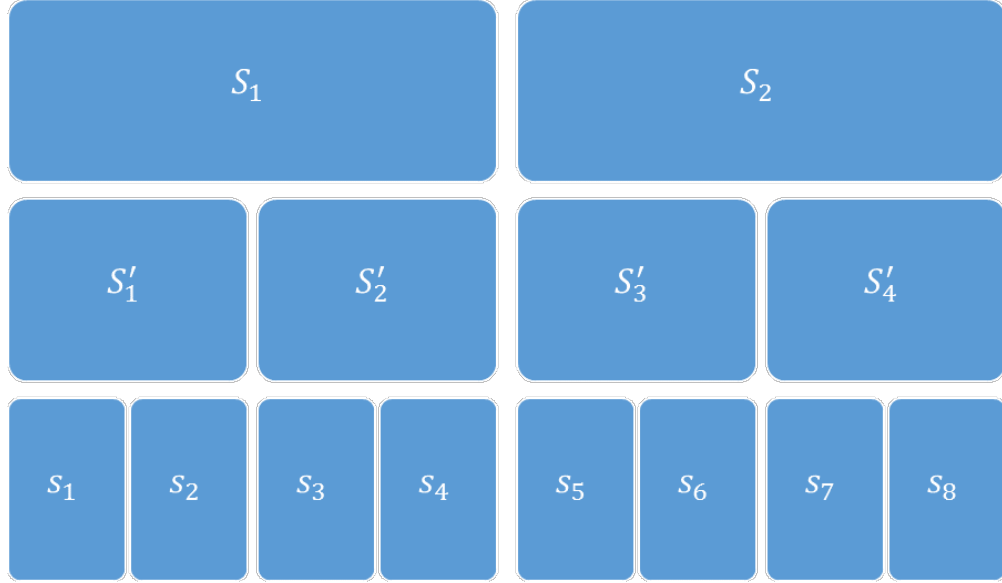


FIGURE 7.32: Example for notational purposes: Wong's method

The proposed optimisation problem from the SSSWong algorithm is

$$\min_{c \in \mathbb{C}} \left[w \varphi(c) + (1 - w) \sum_{j=1}^K d(S_{cj}) \right]. \quad (7.3)$$

Note that in the algorithm, the values of $\varphi(c)$ and $d(S_{cj})$ are also standardised for the same reasoning provided for the SSSKM algorithm in Section 3.9.4.

7.7 Design of the algorithm

Figure 7.33 illustrates the steps followed for a single segmentation analysis using SSS-Wong.

As can be seen, there are five steps:

1. Preliminary segmentation.
 - Similar to Wong's method, the first step creates the preliminary segments that will be iteratively combined, using Equation 7.3.
2. Preliminary segment inspection.

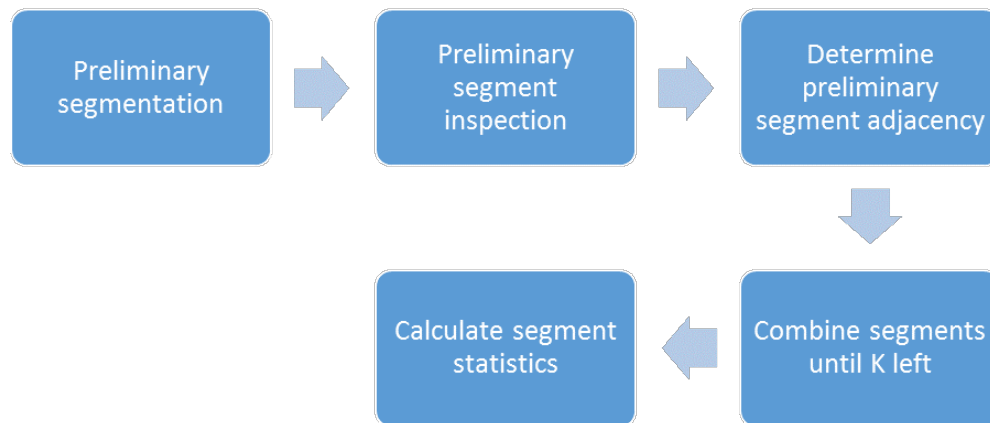


FIGURE 7.33: Process flow diagram for SSSWong

- Due to target rate separation metrics like IV that are unable to deal with segments where there are no events, or non-events, the preliminary segments are investigated for the existence of such segments, and these are combined using Wong's standard density measure.
3. Determine preliminary segment adjacency.
 - All adjacent segments to each preliminary segment are identified using a k-nearest neighbour type approach.
 4. Combine segments until K left.
 - Segments are iteratively combined until the required number of segments are left.
 5. Calculate data set statistics.
 - Statistics like the IV and CH value obtained per segment can now be calculated.

These will be discussed in more detail in the sections to follow.

7.7.1 Step 1: Preliminary segmentation

This step consumes as input the initial number of segments to form as well as the minimum number of observations that can be in a preliminary segment. The algorithm

then performs standard k-means cluster analysis on the data set to form segments with at least the indicated number of segments.

For the purpose of forming preliminary segments, SAS[®] software's Fastclus procedure is used, with the "delete" option set to the required minimum number of observations. The SAS[®] software explains in their documentation that if the "delete" option is set to a value greater than 0, the algorithm would delete cluster seeds that end up with less than the specified number of observations. Once deleted, the clustering is re-performed without the deleted cluster seeds (which means fewer initial segments will be formed). This iterative option is repeated until all preliminary segments contain more than the specified number of observations, or the maximum number of iterations (in this case 999) has been completed. For our purposes the value of the initial number of segments (with which Proc Fastclus starts) is set to 500 as experiments have shown that this is enough for the data sets used in this study.

In Wong's methodology, the input is only the number of initial segments that should be formed (and not also the minimum size of the segments), but because the focus of the SSSWong algorithm is to reduce overfit, avoiding very small preliminary segments is preferable and the algorithm is therefore adjusted as described above.

The output of this step is a data set with segment assignments of all the data points to preliminary segments $(s_{c1}, s_{c2}, \dots, s_{cq})$.

7.7.2 Step 2: Preliminary segment inspection

As was the case with the SSSKMIV algorithm, the requirement of IV to have at least one event and one non-event in each segment requires this additional step. Although this rarely happens, preliminary segments are combined when it is found that they contain only events or only non-events. The combination is done with the segment that is closest based on Wong's standard density measure provided in Equation 2.13.

7.7.3 Step 3: Determine preliminary segment adjacency

As discussed in Section 2.6.3, only one segment (the closest one as measured by Euclidean distance) can be adjacent to another according to Wong's method. This is simple to execute and works well when the objective function is only based on the explanatory variables, but when additional target variable information is incorporated, Euclidean distance no longer represents the full objective function of the semi-supervised algorithm.

For this reason, it was necessary to design a different method for determining adjacent segments. The algorithm that was developed consumes a value k_{nn} as input, and considers the k_{nn} closest neighbours of each of the observations that are assigned to a segment. Note that the subscript "nn" is used here to refer to the term "nearest neighbour", and should not be confused with the number of observations, n . The value k_{nn} is a single integer value that is defined by the user as input.

As an example, refer to Figure 7.34, if preliminary segment s_1 consists of 400 observations, and the value of k_{nn} is set to five, then the algorithm will gather the segment assignments of the five closest observations to each of the 400 observations assigned to s_1 . Conceptually, one would therefore expect that for most of the 400 observations closer to the middle of s_1 , all five neighbours will be assigned to s_1 , but observations that are situated on the edge of the segment are likely to have observations from one of s_1 's neighbours closer to it. For example, observations at the right hand edge of segment s_1 is likely to have observations from s_5 as one of its five closest neighbours, whilst observations at the bottom edge of s_1 is likely to have observations from s_3 as one of its five closest neighbours.

Conversely, as long as the value of k_{nn} is small enough compared to the segment sizes, it is unlikely that any observation in s_1 will have an observation from segment s_7 as one of its closest neighbours, and therefore s_7 will not be seen as a neighbour of s_1 , which is correct in this example. This process is therefore likely to identify all segments that are close enough to s_1 to be considered adjacent.

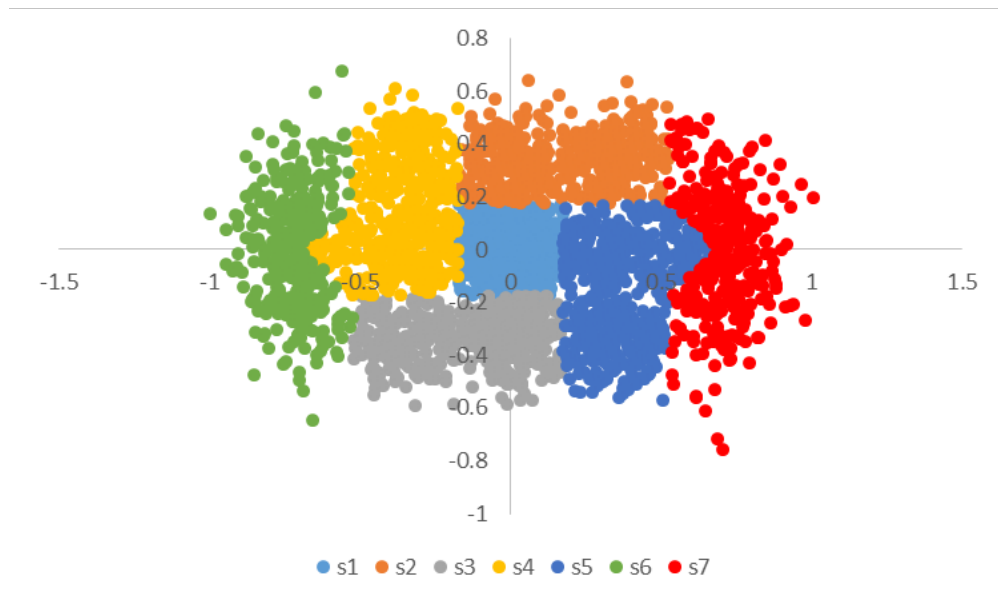


FIGURE 7.34: Example of identifying segment neighbours

This adjacency definition allows the SSSWong algorithm to consider combining more combinations of segments compared to the method prescribed by Wong. The KNN

exercise used to identify segment neighbours is computationally intensive, so is only performed once on the preliminary segments. The assumption is subsequently made that when segments s_1 and s_5 are, for example, combined, then the neighbours of the new, combined segment will be the unique combination of the adjacent segments of both s_1 and s_5 .

To use the example from Figure 7.34, if s_1 is adjacent to s_2 , s_3 , s_4 and s_5 , whilst s_5 is adjacent to s_1 , s_2 , s_3 and s_7 , then the segment formed when s_1 and s_5 are combined, will have neighbours s_2 , s_3 , s_4 and s_7 .

7.7.4 Step 4: Combine segments until K left

After determining the adjacent segments to all the preliminary segments, the algorithm continues to combine adjacent segments, one at a time, that will minimise the objective function as calculated by Equation 7.3. The algorithm does not combine more than one pair for any given cycle, only a single pair is combined. Whenever this happens, the value of the objective function is changed for each adjacent segment to the newly combined segment, so the objective function need to be recalculated.

Segments are combined until the required number of segments remain.

7.7.5 Step5: Calculate data set statistics

Once the required number of segments are reached, the algorithm stops and the values of IV and CH can be calculated as described in Sections 2.11.1 and 2.12. In addition to this, the identified segments can now be used for predictive modelling and the improvement can be measured when compared to a non-segmented data set, as was done in all of the previous analyses performed.

In the next section, we apply the SSSWong algorithm to the (by now well known) data sets which were used in Chapter 5 and Section 7.3.

7.8 SSSWong results

In this section we apply the SSSWong algorithm to the data sets introduced in Chapter 5 and analyse the results.

7.8.1 Direct marketing data set results for SSSWong

For the direct marketing data set we use Figure 7.35 and Table D.37 to analyse the impact that the SSSWong algorithm has on the IVs and the CH values of the final segments. By now the amount of information to show has become extensive. We show IVs in line graphs, which is read from the right hand axis, whilst CH values are read from the left hand side and are represented by bar graphs.

From the graph and table, we observe that the SSSWong algorithm finds higher IVs on average, even higher than the decision tree (albeit at the cost of slightly lower CH values). We also see that there is no evidence of overfit from the IVs as the value of w increases, which indicates that this algorithm is successful in avoiding overfit for this data set, even for very high IVs. The decision tree is not impacted by the value of w used for the SSSWong algorithm, and therefore produces similar CH and IVs for all values of w .

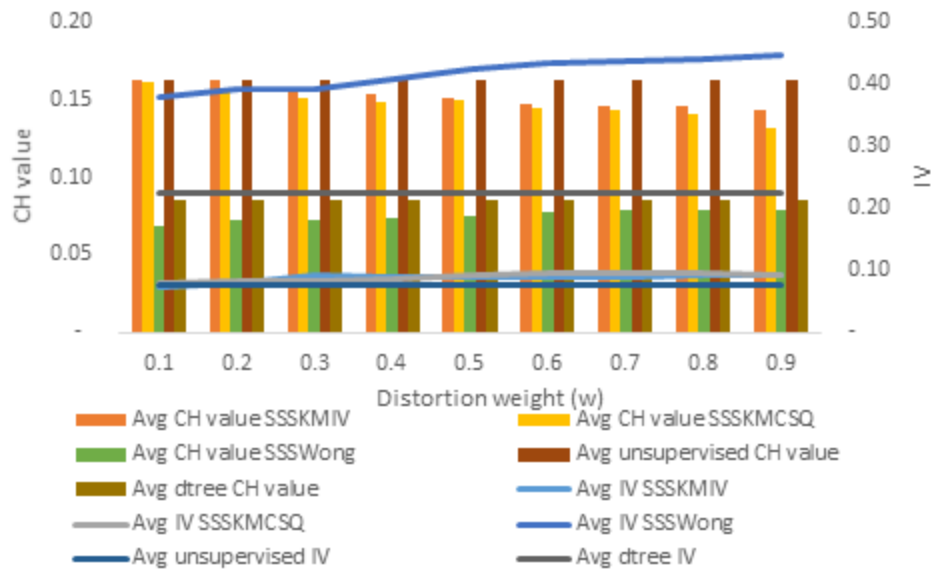


FIGURE 7.35: IV and CH values compared to other segmentation techniques

Figure 7.36 and Table D.38 show the relationship between increasing IVs and the Gini improvement obtained by the SSSWong algorithm. IVs are read off the left hand axis, whilst Gini improvement (over the unsegmented model on the validation set) is read off

the right hand axis. From the figure, it is clear that the SSSWong algorithm is not able to improve unsegmented Ginis at all. However, the segmented Gini continues to improve as IVs improve, and we do not see the same trend of IV overfit as we have seen with the SSSKM algorithms in Chapter 5 (see again Figure 5.3).

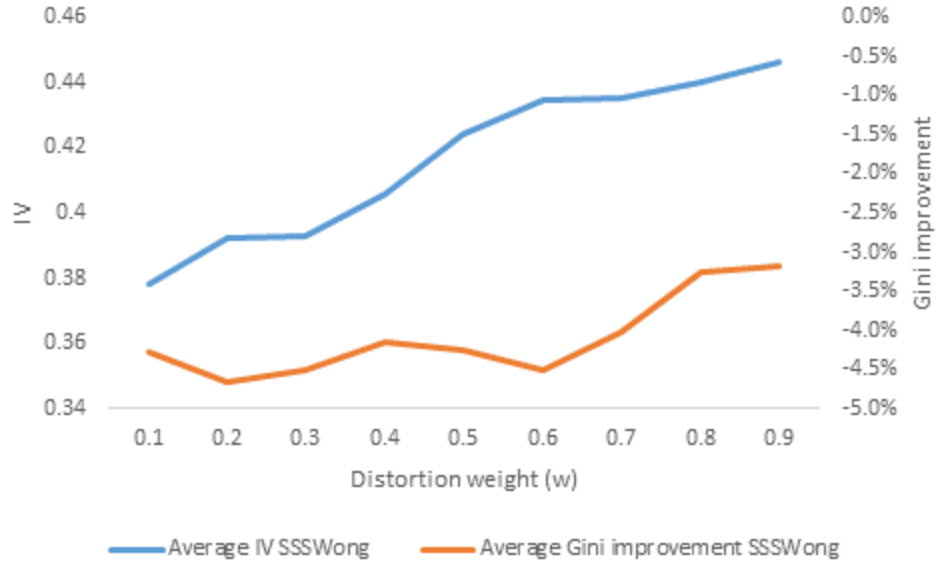


FIGURE 7.36: Comparison of IVs obtained versus Gini improvement

Finally, we compare the success of the SSSWong algorithm to that of all the other segmentation methodologies analysed, as measured by Gini. This can be seen in Figure 7.37 and Table D.39. Actual Ginis can be seen in Table D.40. Similar to the graphs used in Chapter 5 and Section 7.3, we show the best Gini improvement obtained by number of segments, for $K \in \{1, \dots, 10\}$ (or in cases where no improvement is achieved, we show the least Gini worsening). The SSSWong algorithm performs worse than the SSSKMIV_{SSE} algorithm in this case and is clearly not able to contribute to improvement in model accuracy for this data set. The Gini results also do not improve as the number of segments are increased (all values are negative).

Note again that one should not confuse the success that a segmentation algorithm has in finding high IVs between segments, with the final combined Gini improvement that it obtains. For this data set, the SSSWong algorithm finds segments that have very different target rates, but when predictive models are developed on the segments, these fail to add substantial further value.

7.8.2 Protein tertiary structures data set results for SSSWong

Figure 7.38 and Table D.41 compare the IVs and CH values achieved by the SSSWong algorithm to the other segmentation algorithms analysed. Again, it can be seen that the

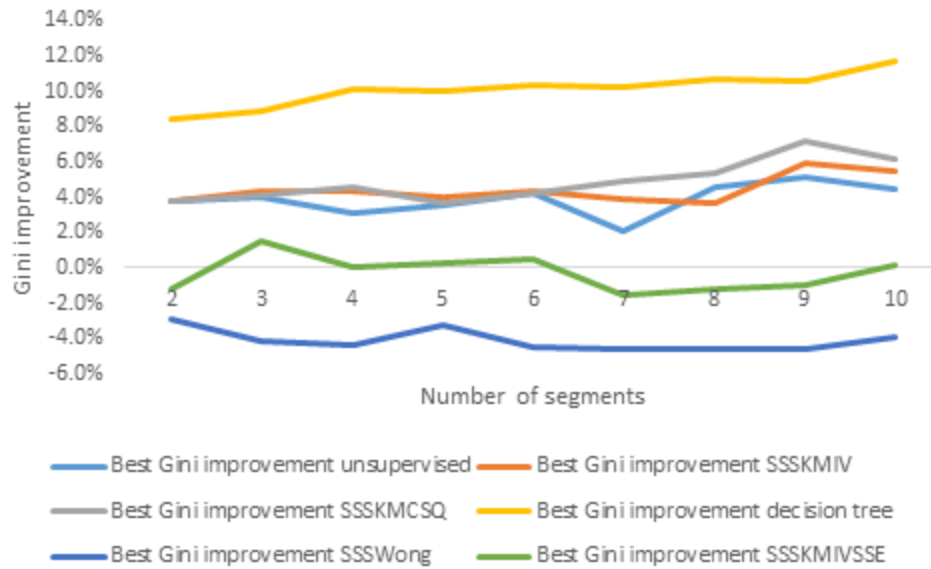


FIGURE 7.37: Gini improvement by number of segments

algorithm achieves the best IVs, even better than the decision tree. No overfit on IVs can be observed as the value of w is increased. This is in clear contrast to the overfit we observe for the SSSKM algorithms. We can see that SSSWong does not achieve high CH values, which also aligns with what was seen on the direct marketing data set.

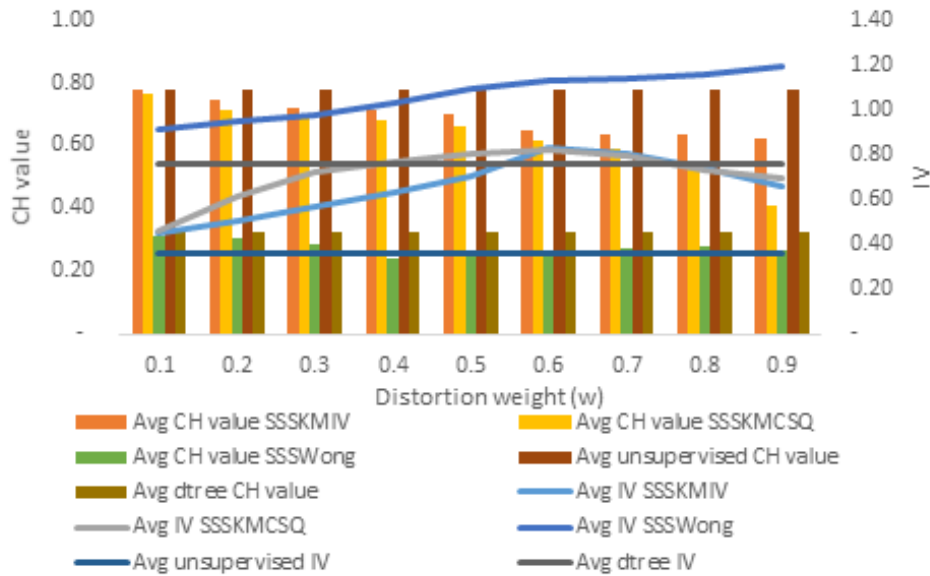


FIGURE 7.38: IV and CH values compared to other segmentation techniques

In Figure 7.39 and Table D.42, we see a strong correlation between the IV obtained and the Gini improvement. As the distortion weight increases, both the achieved IV as well as the Gini improve.



FIGURE 7.39: Comparison of IVs obtained versus Gini improvement

From Figure 7.40 and Table D.43 as well as Table D.44, we see that, although the SSSWong algorithm is very successful at finding segments with high IVs on this data set, it is not successful when comparing Gini improvement to the other segmentation techniques. The algorithm outperforms unsupervised segmentation for values of $K < 5$, but is not very successful for higher values of K .

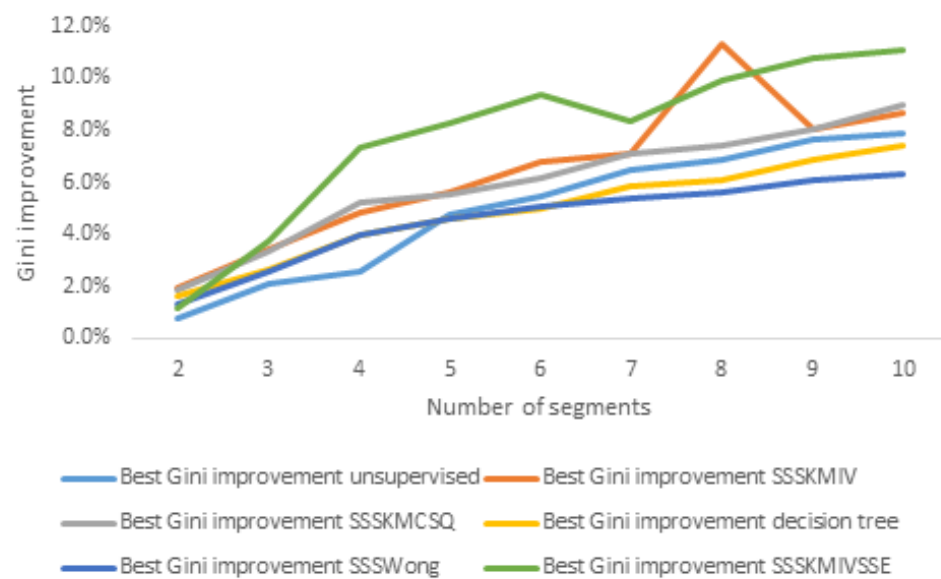


FIGURE 7.40: Gini improvement by number of segments

7.8.3 Credit application data set results for SSSWong

In Figure 7.41 and Table D.45, we see that the decision tree continues to obtain the best IV results for the credit application data set. The IVs obtained by SSSWong are, however, much better than the SSSKM algorithms. Although there is no overfit on IV, we observe that increasing the value of w does not increase the IVs obtained by much. The SSSWong algorithm achieves higher CH values than the decision tree, but lower than the SSSKM algorithms.

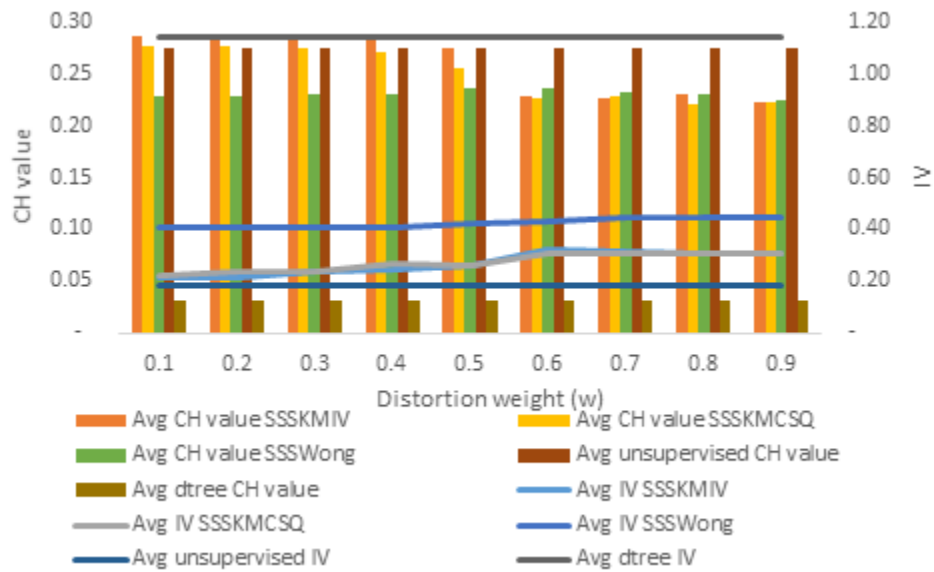


FIGURE 7.41: IV and CH values compared to other segmentation techniques

Figure 7.42 and Table D.46 show a curious trend where an increase in IV does not result in higher Ginis for the credit application data set. That being said, it must be noted that the average Gini improvement only drops slightly from 22.7% to 21.7%.

In Figure 7.43 and Table D.47 as well as Table D.48, we see that the SSSWong algorithm again performs worse as compared to all other analysed segmentation techniques. It is clear that a pattern is emerging where the algorithm is very successful at finding high IVs with no overfit, but this does not translate in improved Ginis.

7.8.4 Wine quality data set results for SSSWong

In Figure 7.44 and Table D.49, it can be seen that the SSSWong algorithm again obtains the best IVs for the wine quality data set. IVs continue to increase as the value of w increases, with no overfit evident. The SSSWong algorithm actually obtains both higher CH values and IVs than the decision tree algorithm, but the CH values are much smaller than those obtained by the SSSKM algorithms or the unsupervised segmentation.

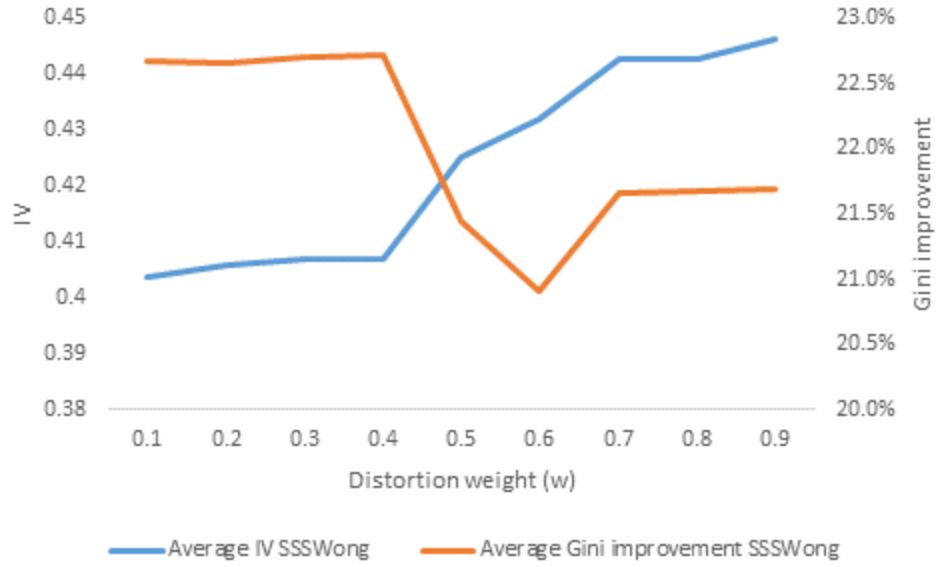


FIGURE 7.42: Comparison of IVs obtained versus Gini improvement

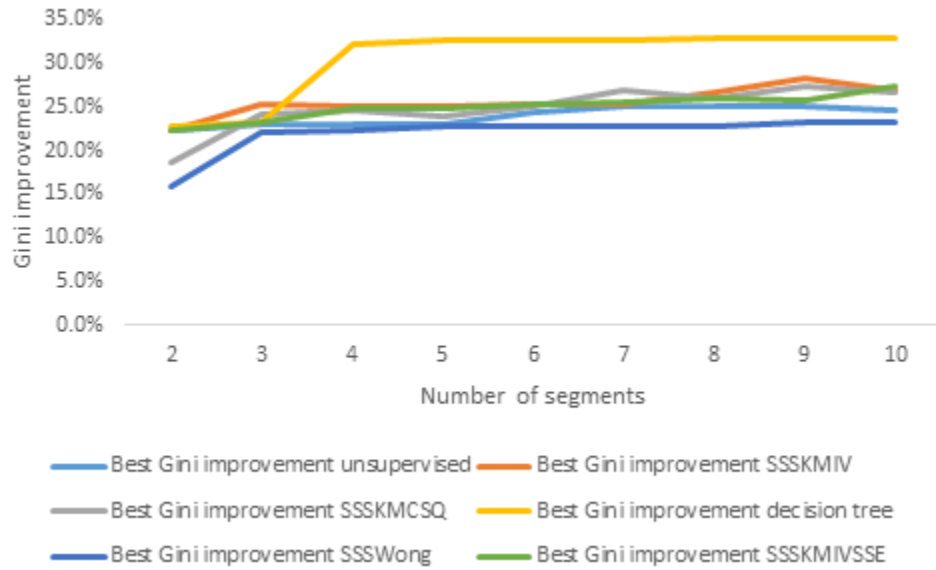


FIGURE 7.43: Gini improvement by number of segments

In Figure 7.45 and Table D.50, we see again that the Gini does not always improve as the IVs improve, with Gini obtained by the SSWong algorithm peaking at $w = 0.5$.

In Figure 7.46 and Table D.51 as well as Table D.52, we see that the SSSWong algorithm outperforms decision trees for $K = 4$, and unsupervised segmentation as well as SSSKMIV_{SSE} for $K = 2$. In all other cases it performs worse than the other segmentation techniques. It is becoming clear that the segments formed by the SSSWong algorithm may not be conducive to accurate model development.

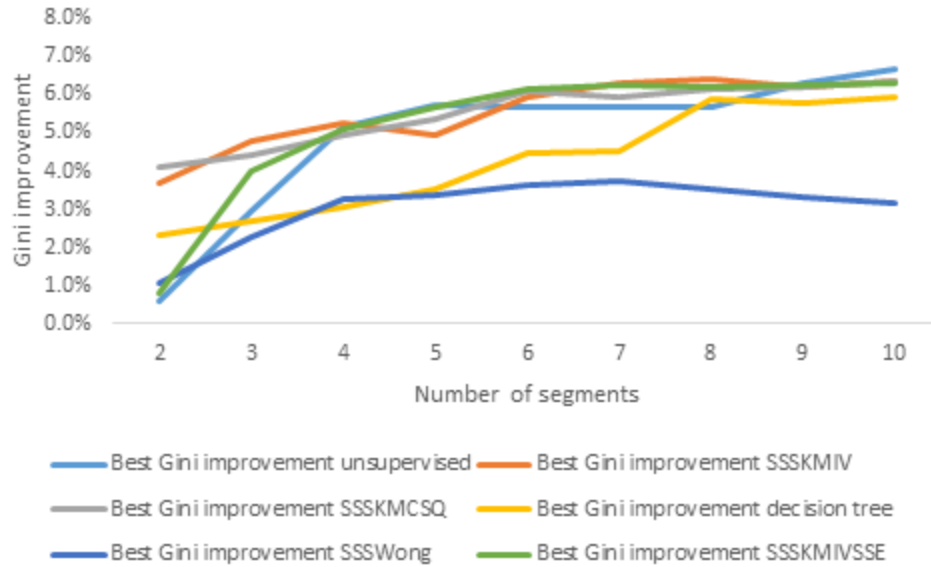


FIGURE 7.46: Gini improvement by number of segments

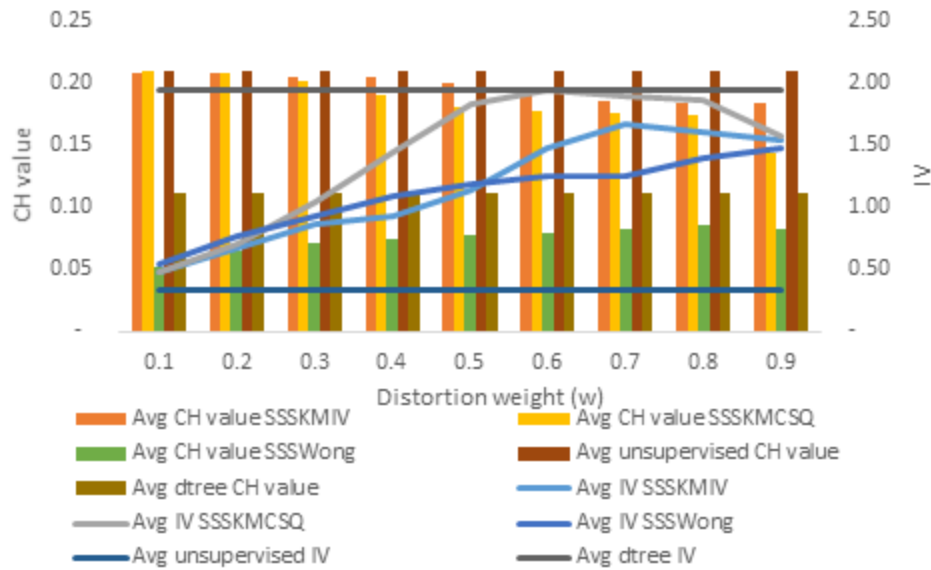


FIGURE 7.47: IV and CH values compared to other segmentation techniques

increasing Gini, as was observed in the results in Chapter 5. However, when we analyse Figure 7.49 and Table D.55 as well as Table D.56, we clearly see that the SSSWong algorithm performs much worse than the other algorithms analysed.

7.8.6 Insurance claim prediction data set results for SSSWong

For the insurance claim prediction data set, we can see from Figure 7.50 and Table D.57 that the SSSWong algorithm is able to find much better IVs compared to all other segmentation methodologies. We also do not see any evidence of overfit for the IVs.

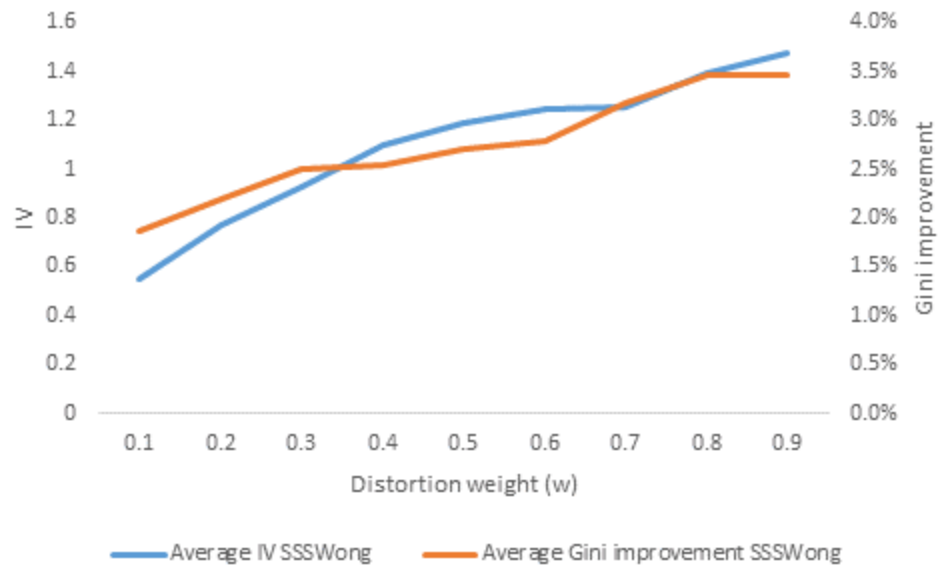


FIGURE 7.48: Comparison of IVs obtained versus Gini improvement

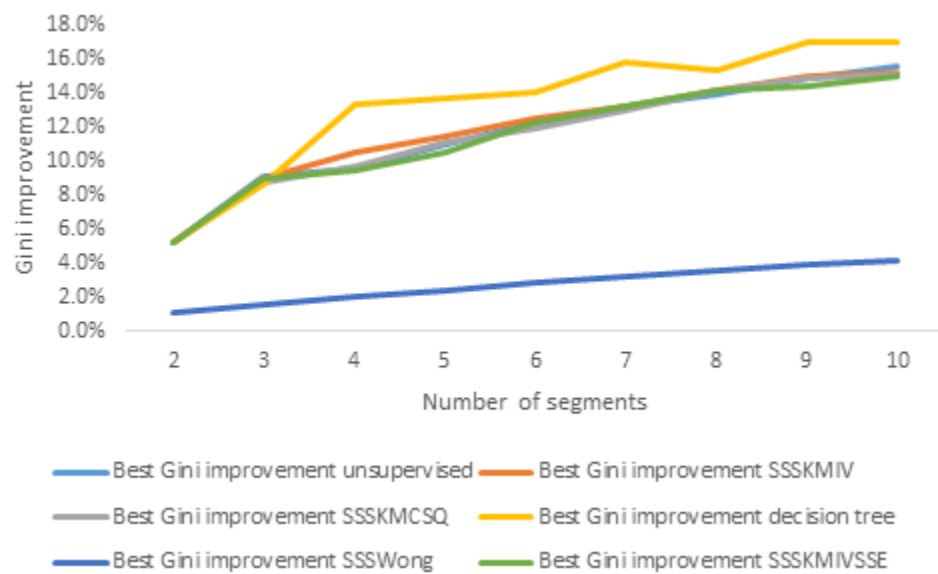


FIGURE 7.49: Gini improvement by number of segments

In Figure 7.51 and Table D.58, we again see an increasing IV and Gini when the value of w is increased.

However, similar to all other data sets analysed in this study, we see in Figure 7.52 and Table D.59 as well as Table D.60, that the SSSWong algorithm does not perform well when predictive models are developed on the segments it forms.

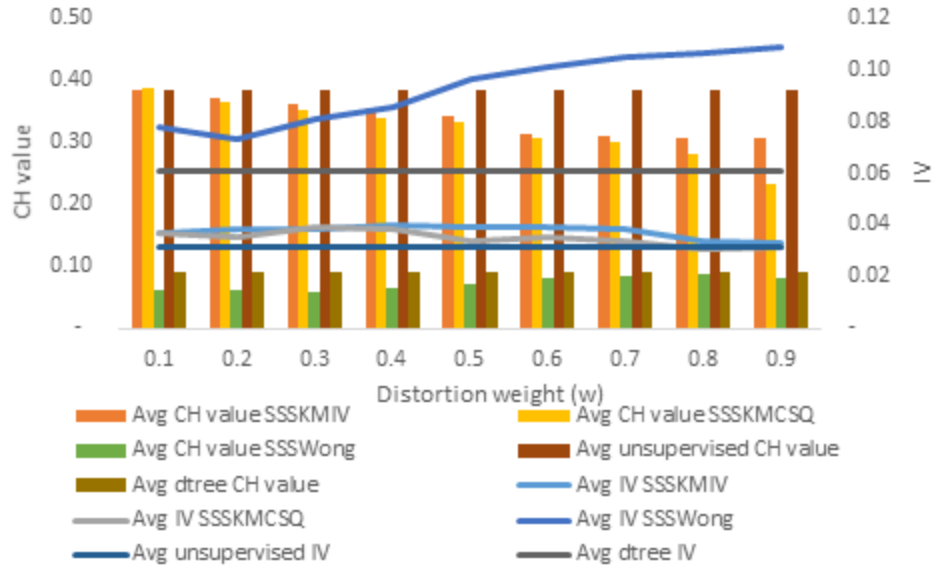


FIGURE 7.50: IV and CH values compared to other segmentation techniques

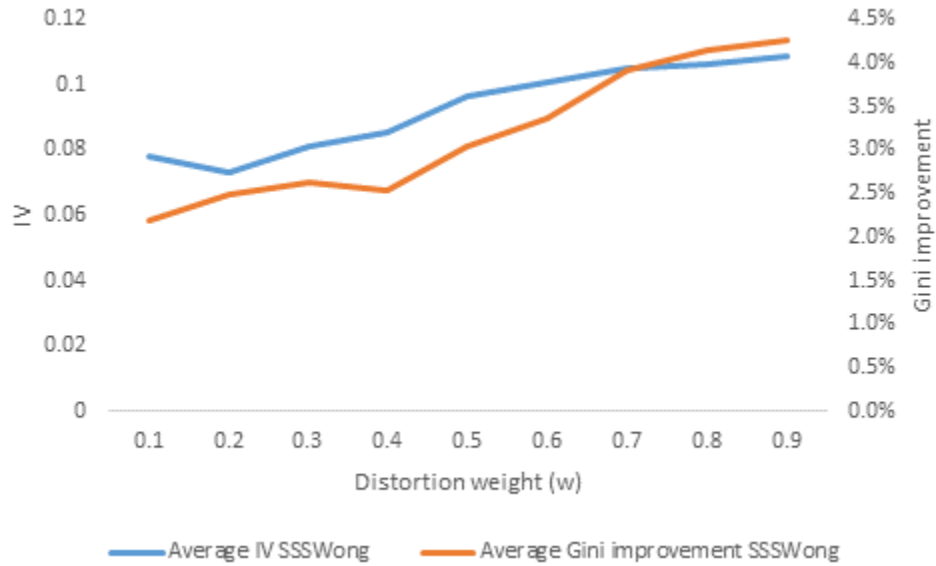


FIGURE 7.51: Comparison of IVs obtained versus Gini improvement

7.9 Summary of SSSWong results

In Section 5.8, we observed that a clear disadvantage of the SSSKM algorithms are that they tend to overfit when the value of w becomes too large. We have also observed that, when these algorithms start to overfit, the resulting Gini improvements that they achieve also reduce in most cases. In Section 7.2, we introduced the SSSKMIV_{SSE} algorithm which adds segment size equality as an additional element to the objective function. Here we have continued to see overfit when the values of $w + \nu$ becomes too large.

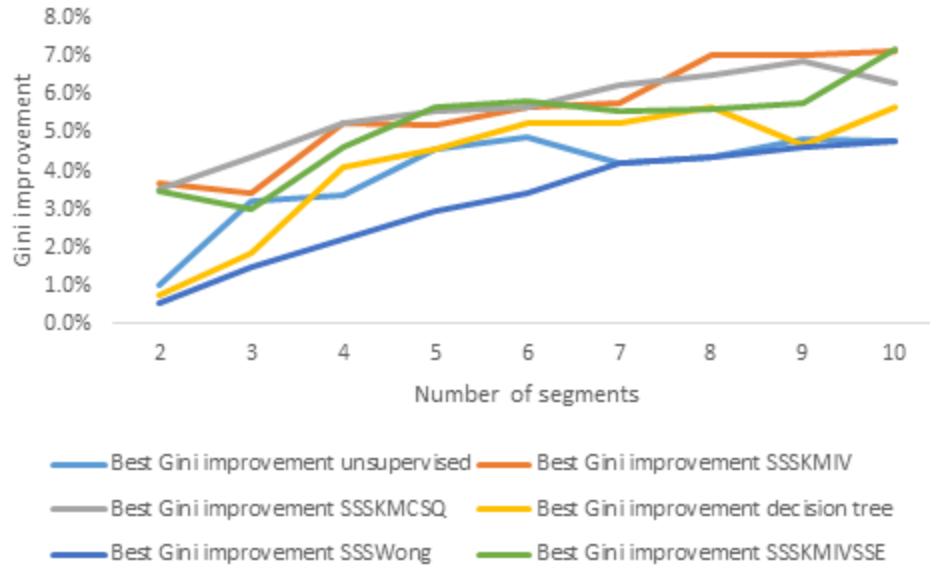


FIGURE 7.52: Gini improvement by number of segments

Overfit was evident in the $\text{SSSKMIV}_{\text{SSE}}$ algorithm for both the IV and segment size equality.

Because of the impact of overfit that was observed with the SSSKM algorithms, we introduced the SSSWong algorithm in Section 7.5. As discussed, density clustering has a number of advantages over k-means clustering and on top of that, Wong's algorithm starts with initial segments already formed, which is likely to reduce the impact of overfit.

When applying the SSSWong algorithm to the same data sets analysed in Chapter 5, we observe in all cases that the SSSWong algorithm is able to avoid IV overfit. Even for high values of w , the IVs obtained by the algorithm on the validation set did not decrease. We also found that, for most data sets, the SSSWong algorithm finds very high IVs - in many cases even higher than the decision tree. This may indicate that the SSSWong algorithm could potentially be applied in areas where the goal of the algorithm is supervised classification, and not segmentation for predictive modelling, as is the case here.

The focus of this study is, however, on segmentation for predictive modelling, and therefore the combined Gini improvement obtained when predictive models were separately developed on the SSSWong segments, were compared to the Gini improvements obtained by the other segmentation techniques. Surprisingly, the algorithm did not perform well on any data set, and it produced the worst overall results in all cases. This could be seen as quite surprising since density clustering is generally accepted to be able to identify more complex segments when compared to k-means clustering.

Something to consider, though, is the results obtained in Chapter 4. In Chapter 4, we have shown that the addition of the distortion factor to the standard k-means objective function allows the k-means algorithm to overcome many of its known weaknesses. It may therefore well be that this perceived advantage of the SSSWong algorithm is therefore not really an advantage as the SSSKM algorithms are already able to form more complex segments.

We can therefore observe that, although we succeeded in developing an algorithm in SSSWong that avoids the overfit that is observed in the SSSKM algorithms, and is actually able to obtain better IV results than the fully supervised decision tree, we have not been able to find a data set on which the SSSWong algorithm performs best. This does not mean that such a data set does not exist though.

7.10 Augmenting the LK-Means algorithm for semi-supervised segmentation

In Sections 2.9 and 2.10 we discussed and compared the LK-Means means supervised clustering methodology (proposed by Peralta et al. (2013)) to the semi-supervised segmentation methodologies proposed in this study. We concluded that the methodology has many similarities to semi-supervised segmentation algorithms, but also that there are a number of clear differences. The major difference between the two methodologies is that the ultimate goals were different, as summarised in Table 2.4.

Even though the authors of the paper that proposed LK-Means had a different goal in mind, there could be value in testing the LK-Means' ability to segment for predictive modelling. The detailed steps of the algorithm for LK-Means is described in their paper (see (Peralta et al., 2013)), and therefore these will not be repeated here. As discussed in Section 2.9, the LK-Means methodology optimises the objective function (see Equation 2.14 for more details),

$$\min_{c \in \mathcal{C}} \left[w \sum_{j=1}^K \sum_{l=1}^L \sum_{i \in S_{cjl}} d(\mathbf{x}_i, \mathbf{u}_{cjl}) \nu_{cjl} + (1 - w) \sum_{j=1}^K \sum_{i \in S_{cj}} d(\mathbf{x}_i, \mathbf{u}_{cj}) \right]. \quad (7.4)$$

One aspect that is not described in detail by the authors, is how the results of a LK-Means algorithm can be applied to new data sets, that do not contain known target values. This is most likely due to the fact that the application of LK-Means that they had in mind was different to semi-supervised segmentation for predictive modelling. Since we require the ability to apply segmentation results, obtained from a development

set, onto a validation set, we will need to augment the LK-Means algorithm so that it possesses that ability.

To ensure the results of LK-Means are as comparable as possible to the other results obtained in this study, we make use of the exact same smoothing methodology that was used for the SSSKMIV and SSSKMCSQ methodologies, to augment LK-Means for this purpose. In Section 3.14 we describe how k nearest neighbour smoothing can be used to apply results of a segmentation exercise on a development set, to a validation set. This methodology can also be used to ensure overfit is clearly visible when it occurs.

The augmentation to the LK-Means methodology, that is done in this study, is minor, and summarised as follows: The data set is separated into a development set and a validation set, and LK-Means is only performed on the development set. The results obtained on the development set, is then applied to the validation set, using KNN smoothing. The development and validation sets used are identical to those used for the other segmentation methodologies.

7.11 LK-Means results

To measure the performance of the LK-Means methodology, we follow a very similar process as was done with the SSSKMIV, SSSKMCSQ and SSSWong methodologies. We also run 20 iterations for each value of $w \in \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$. As with the previous methodologies, these are tested on values of $K \in \{2, \dots, 10\}$. The graphs that we show in the next few sections will be similar to the graphs used to provide the results for the SSSWong algorithm.

7.11.1 Direct marketing data set results for LK-Means

Figure 7.53 and Table D.61 show results in a similar way to what was done for SSSWong in Figure 7.35 and Table D.37. Given the amount of information, the results of SSSWong is not repeated in these graphs, but LK-Means is compared to the segmentation techniques explored in Chapter 5. We do, however include SSSWong in the final Gini comparison shown in Figure 7.55 and Table D.63.

It can be seen from Figure 7.53 and Table D.61 that the LK-Means algorithm is not as successful in finding high IV's as the SSSWong algorithm, but for high values of w it does find better IV's than SSSKMIV, SSSKMCSQ and unsupervised k-means. The algorithm also seem to be more robust against overfit compared to the other k-means based semi-supervised techniques as the IV does not decrease for higher values

of w . It does seem that the CH values produces by LK-Means are, however, lower than SSSKMIV, SSSKMCSQ and unsupervised k-means.

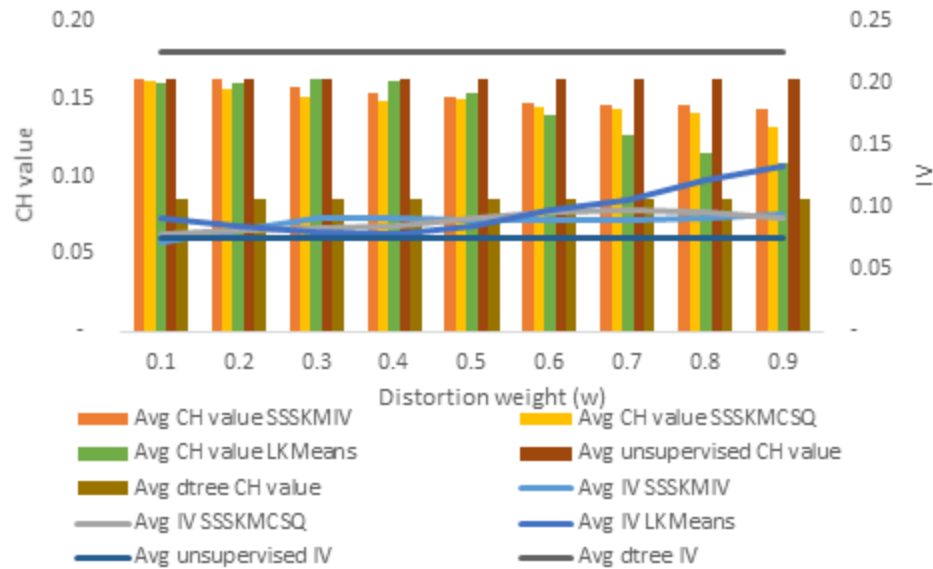


FIGURE 7.53: IV and CH values compared to other segmentation techniques

Figure 7.54 and Table D.62 show the relationship between increasing IVs and the Gini improvement obtained by the LK-Means algorithm (similar to Figure 7.36 and Table D.38). In this graph we do not see a clear correlation as was seen in Chapter 5, but we also observe that, on average, the LK-Means algorithm is not able to improve the Gini for this data set when used for segmentation.

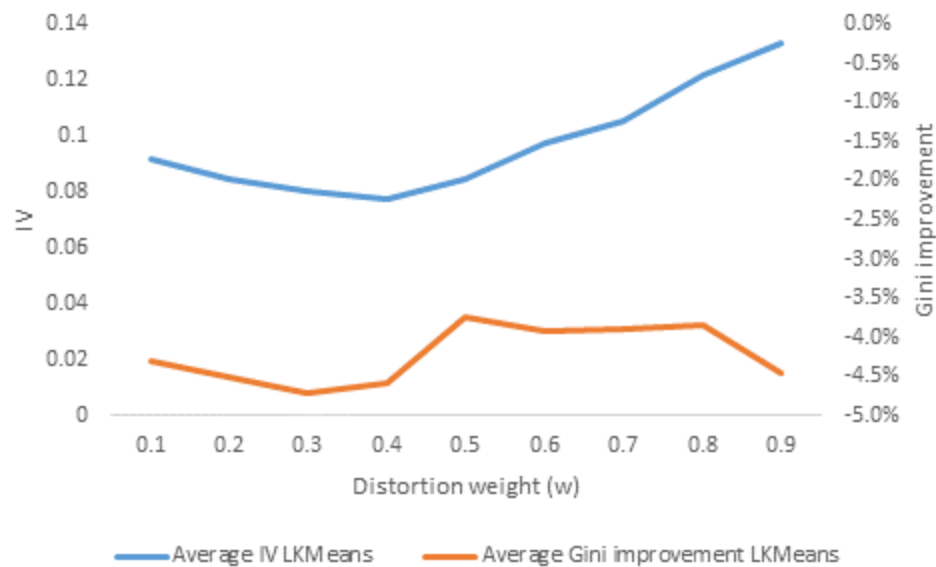


FIGURE 7.54: Comparison of IVs obtained versus Gini improvement

In Figure 7.55 and Table D.63 (similar to Figure 7.37 and Table D.39), we show the best Gini improvement obtained by number of segments, for $K \in \{1, \dots, 10\}$. In this case we include all segmentation methods that were explored previously. LK-Means outperforms $\text{SSSKMIV}_{\text{SSE}}$ and SSSWong, but is not able to obtain similar results to SSSKMIV, SSSKMCSQ, unsupervised clustering or decision trees.



FIGURE 7.55: Gini improvement by number of segments

7.11.2 Protein tertiary structures data set results for LK-Means

LK-Means seem to require high values of w in order to find segments with high IVs, as can be seen from Figure 7.56 and Table D.64. Again the IVs it manages to find is not as high as those produced by the SSSWong algorithm (see Figure 7.38), but at $w = 0.9$ it does better than the decision tree. Again we do not see IVs decreasing as is the case for high values of w in SSSKMIV and SSSKMCSQ.

In Figure 7.57 and Table D.65 we see that the Gini improvement decreases at first for LK-Means up to values of $w = 0.6$, but then Gini improvements increases in line with the increase in IVs observed. We can therefore see a link between the algorithm finding better IVs and better improvements in Gini in this case.

Figure 7.58 and Table D.66 show that LK-Means obtains the best results for $K \in \{2, 3\}$, and contends well for all other values of K with $\text{SSSKMIV}_{\text{SSE}}$ outperforming for $K \in \{4, \dots, 10\}$.

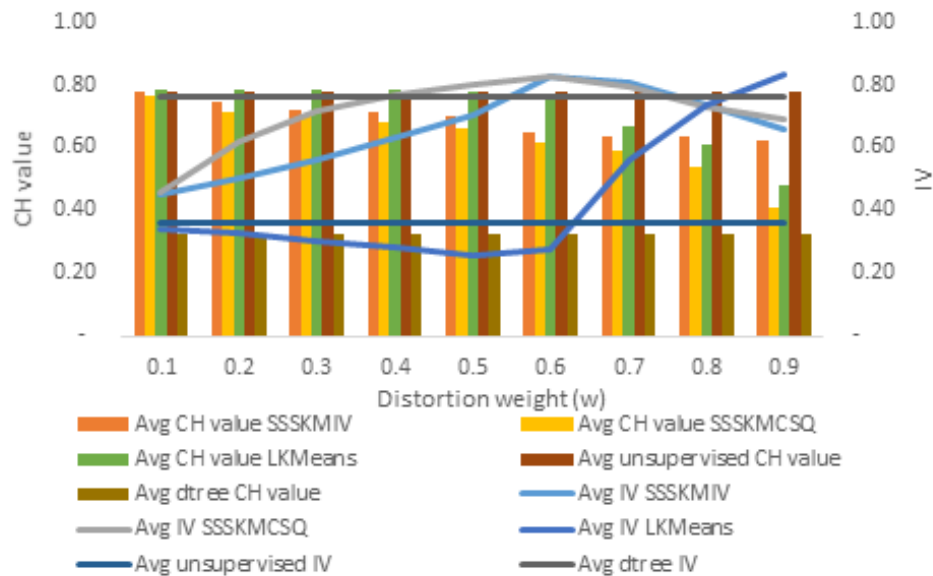


FIGURE 7.56: IV and CH values compared to other segmentation techniques

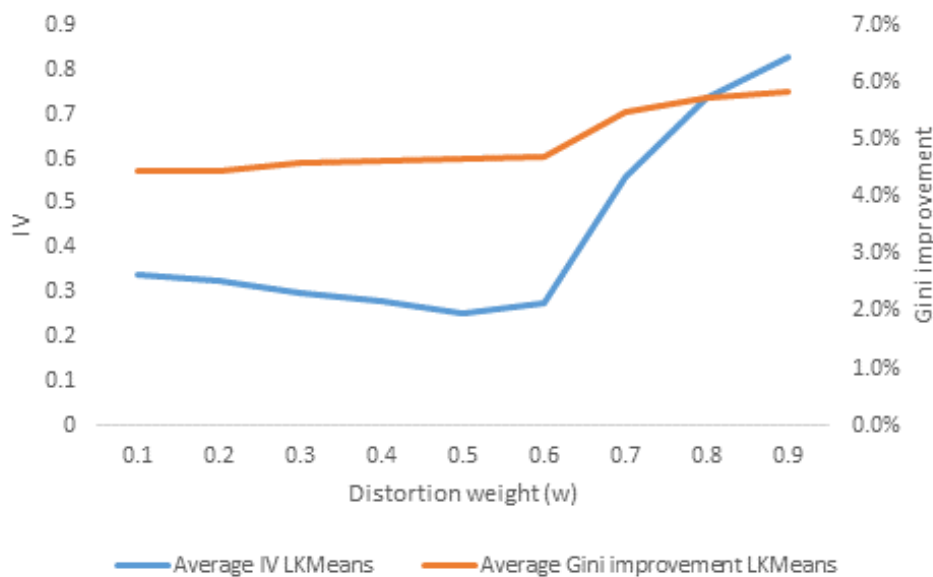


FIGURE 7.57: Comparison of IVs obtained versus Gini improvement

7.11.3 Credit application data set results for LK-Means

For this data set LK-Means seems to struggle to obtain high IVs, as can be seen from Table D.67 and Figure 7.59. The IVs also seem to decrease as the value of w increases. The algorithm also does not do very well when measured by the CH values obtained.

In Figure 7.60 and Table D.68 the IV trend observed in Table D.67 can be seen more clearly. Very little correlation seems to exist between the algorithm's success in obtaining

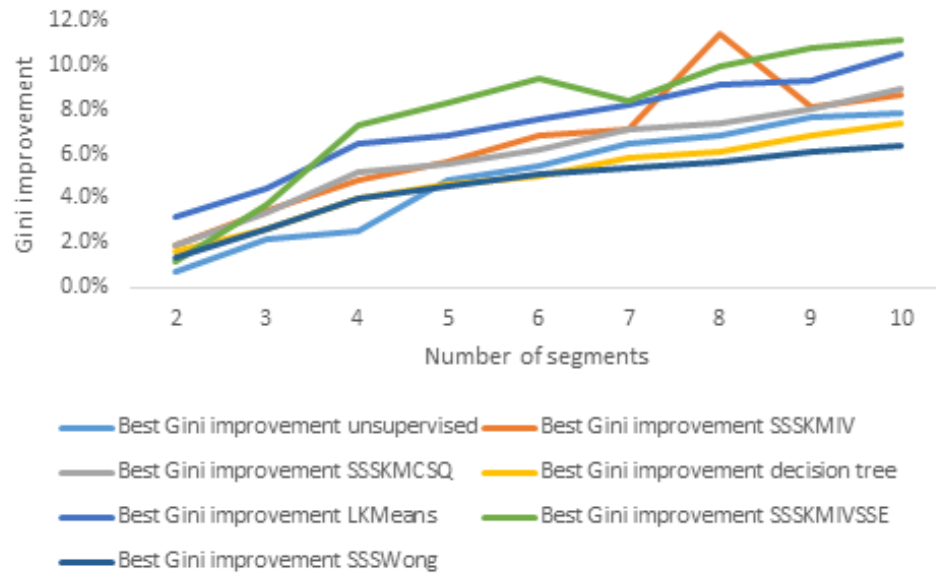


FIGURE 7.58: Gini improvement by number of segments

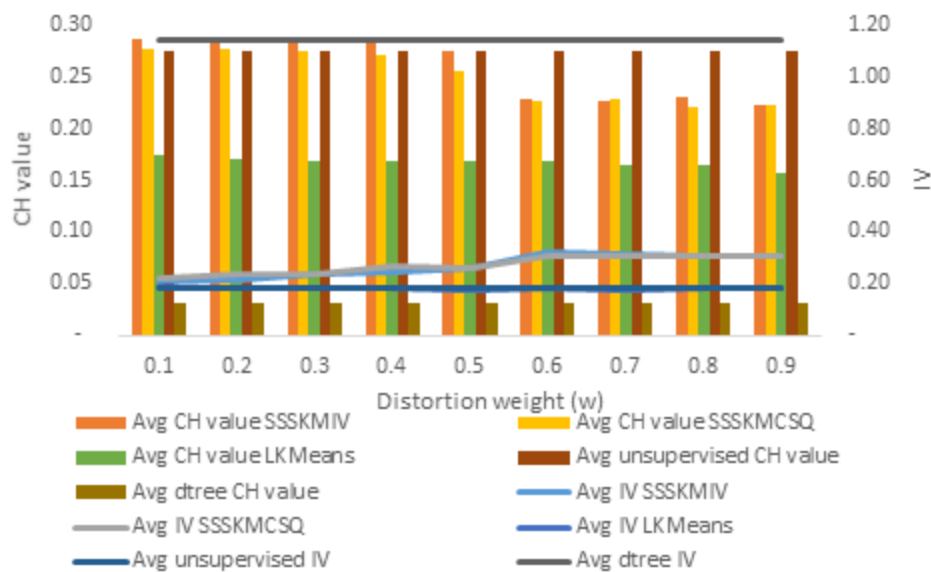


FIGURE 7.59: IV and CH values compared to other segmentation techniques

high IVs and the improvement in Gini obtained. The best average Gini is obtained for $w = 0.9$.

From Figure 7.61 and Table D.69 we see that the LK-Means algorithm does not perform well on this data set when measured by Gini improvement. Decision trees mostly outperform all the other techniques, with LK means only outperforming unsupervised k-means and SSSWong

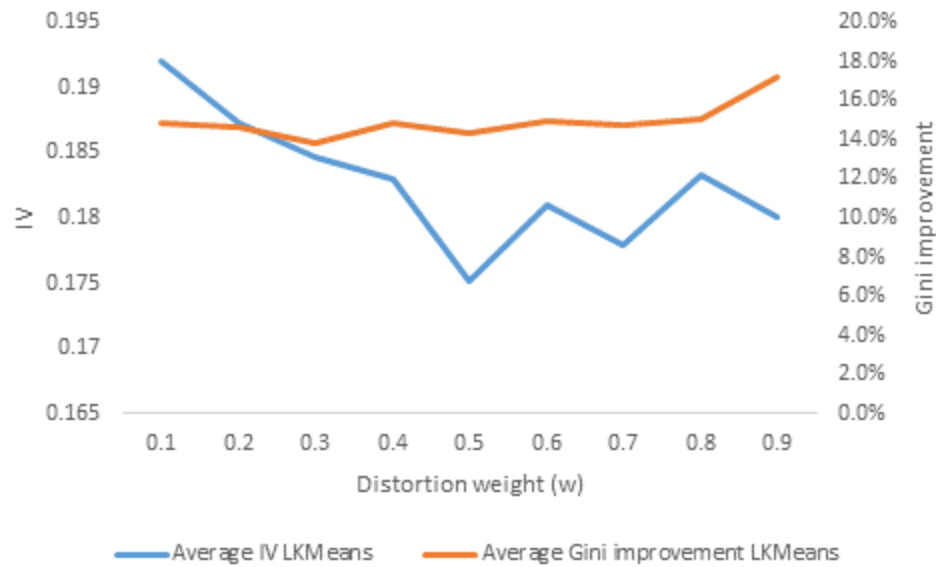


FIGURE 7.60: Comparison of IVs obtained versus Gini improvement

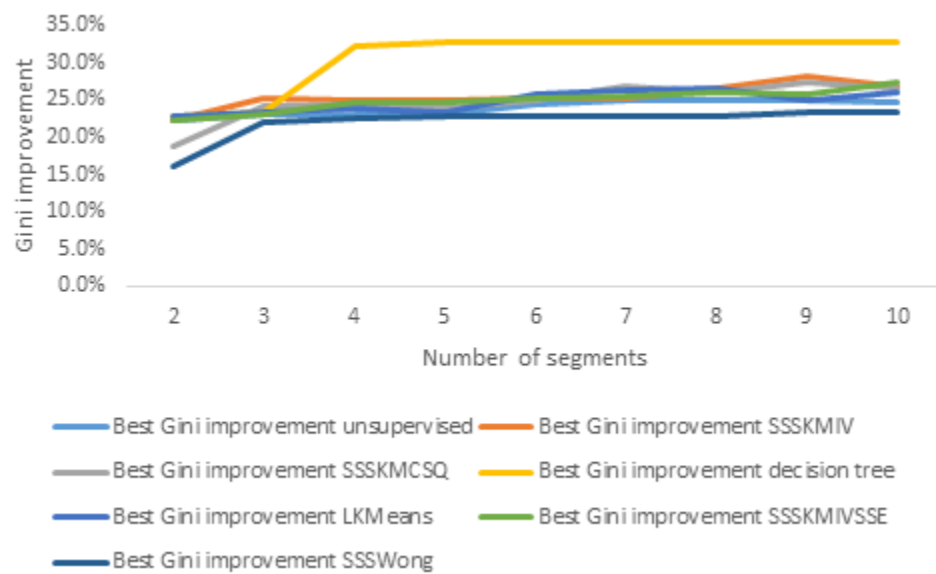


FIGURE 7.61: Gini improvement by number of segments

7.11.4 Wine quality data set results for LK-Means

The results of the IV and CH values for the wine quality data set can be seen in Figure 7.62 and Table D.70. The trends observed are remarkably close to the trends observed for the protein tertiary structures data set (see Figure 7.56 and Table D.64). From $w = 0.6$, the IVs that are obtained increases sharply, with LK-Means outperforming decision tree segmentation for $w = 0.9$. The higher IVs obtained by LK-Means for $w = 0.9$ is obtained without sacrificing as much on CH value compared to the decision tree. A look at Figure 7.44 and Table D.49 does, however, show that LK-Means is not

able to obtain higher IVs than the SSSWong algorithm. Again there is little evidence of overfitting for this data set.

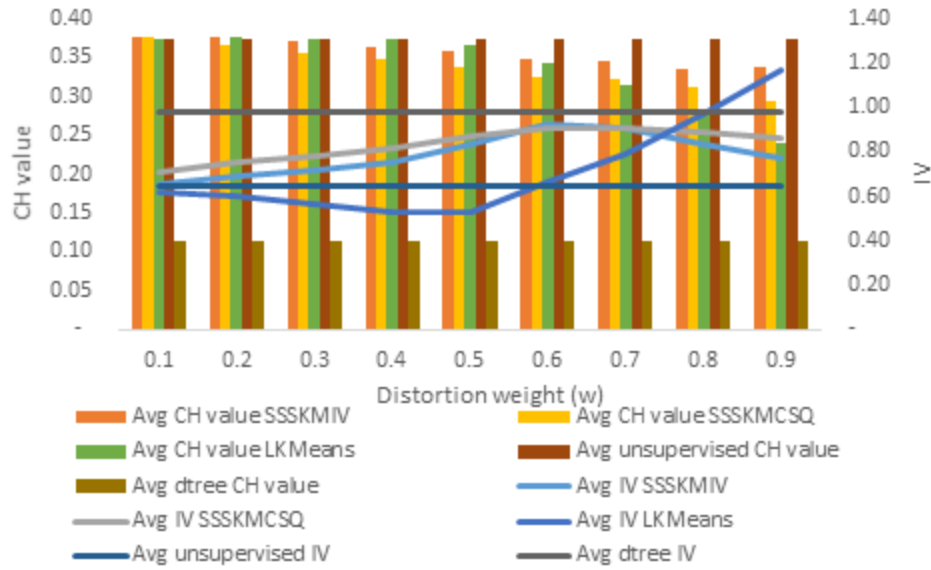


FIGURE 7.62: IV and CH values compared to other segmentation techniques

In Figure 7.63 and Table D.71 we see a clear correlation between the success of obtaining higher IV values and average Gini improvement. From $w = 0.6$, which is the weight where IVs increase sharply, one can also observe an increasing trend in Gini improvement. Gini improvement seem to decrease slightly for $w = 0.9$.

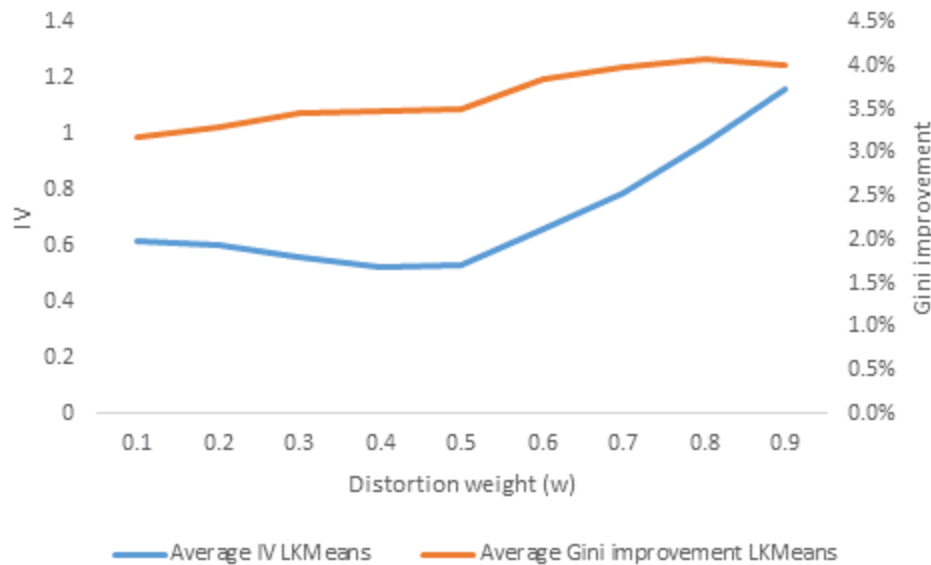


FIGURE 7.63: Comparison of IVs obtained versus Gini improvement

Figure 7.64 and Table D.72 show that, on this data set, LK-Means actually performs the best of all other segmentation methodologies considered. From $K = 3$, the algorithm

outperforms the other methodologies, and obtains its best Gini improvement for $K = 8$, where it improves the Gini by 7.5%. Since unsupervised segmentation outperformed even the non-linear techniques explored in Chapter 5 (see Section 5.9.4), this means that LK-Means outperforms all modelling options explored in this thesis for this specific data set.



FIGURE 7.64: Gini improvement by number of segments

7.11.5 Chess (king-rook vs. king) data set results for LK-Means

From Figure 7.65 and Table D.73, it can be seen that IVs already start increasing for LK-Means from $w = 0.4$, and the IVs continue to increase until $w = 0.9$. Here LK-Means obtain higher IVs than decision trees (clearer from Table D.73 than Figure 7.65), although SSSKMCSQ manages to obtain the same IV at $w = 0.6$ whilst maintaining a higher CH value.

In Figure 7.66 and Table D.74 the correlation between improved IVs and average improvement in Gini is once again clear. The trend is similar to what was observed in Chapter 5. It seems that, as long as the algorithm is effective at improving Ginis, one is likely to observe a correlation between IV and Gini improvement.

From Figure 7.67 and Table D.75 we see that the LK-Means algorithm is not able to outperform decision trees for this data set, but that it performs well compared to other segmentation techniques.

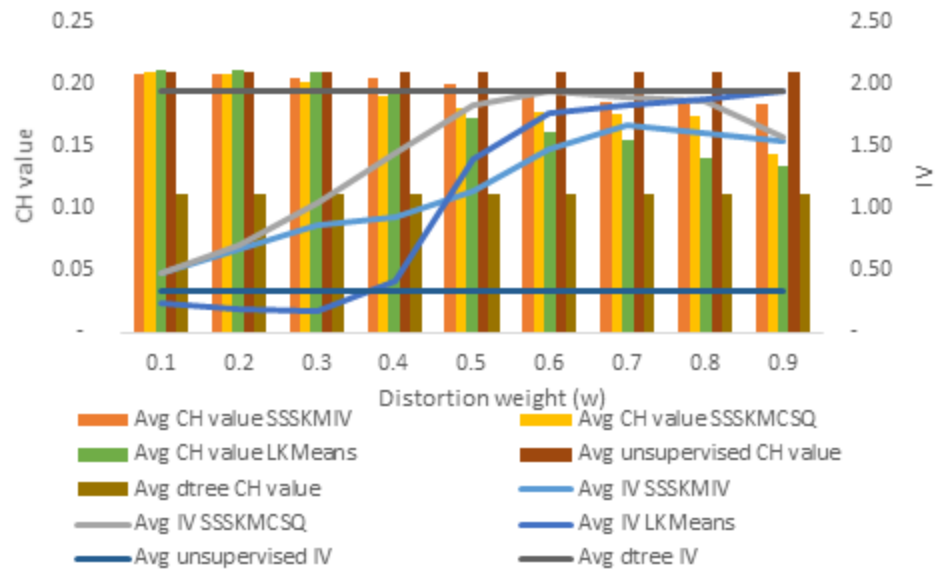


FIGURE 7.65: IV and CH values compared to other segmentation techniques

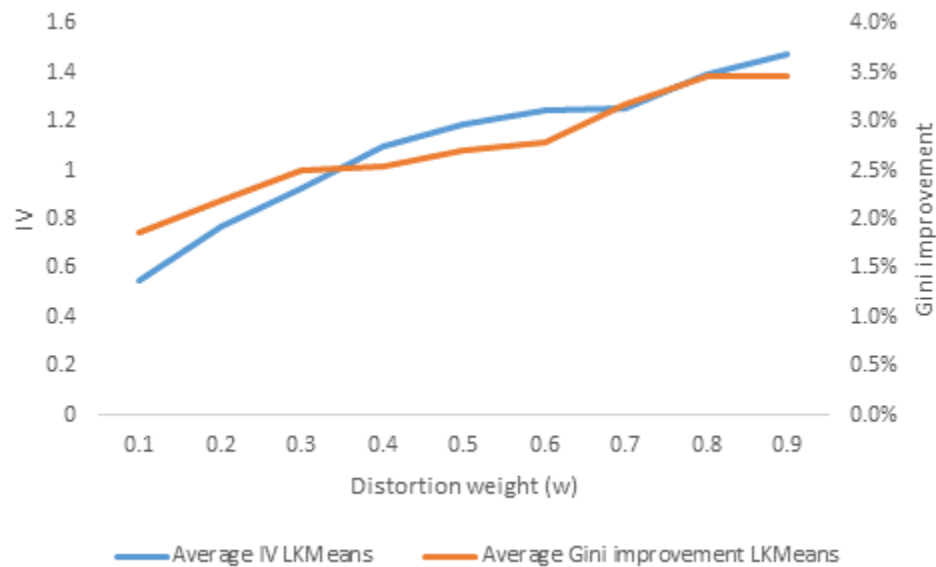


FIGURE 7.66: Comparison of IVs obtained versus Gini improvement

7.11.6 Insurance claim prediction data set results for LK-Means

In Figure 7.68 and Table D.76, it can be seen that LK-Means is able to obtain similar IV values than the other k-means based methodologies, and that it is not able to find IV values that are close to those found by the decision tree for the insurance claim prediction data set. In contrast to the SSSKMIV and SSSKMCSQ methodologies, LK-Means finds the best IV values for values of $w > 0.6$.

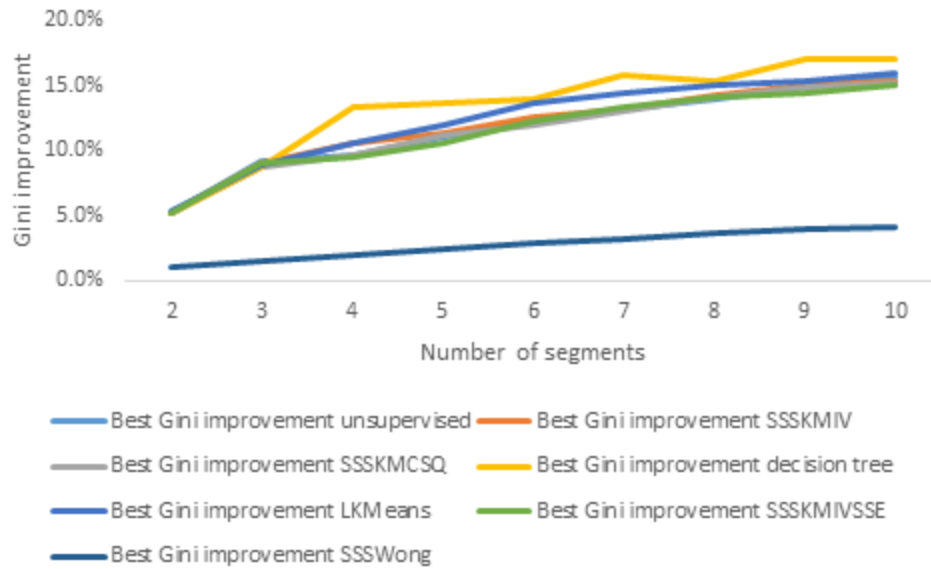


FIGURE 7.67: Gini improvement by number of segments

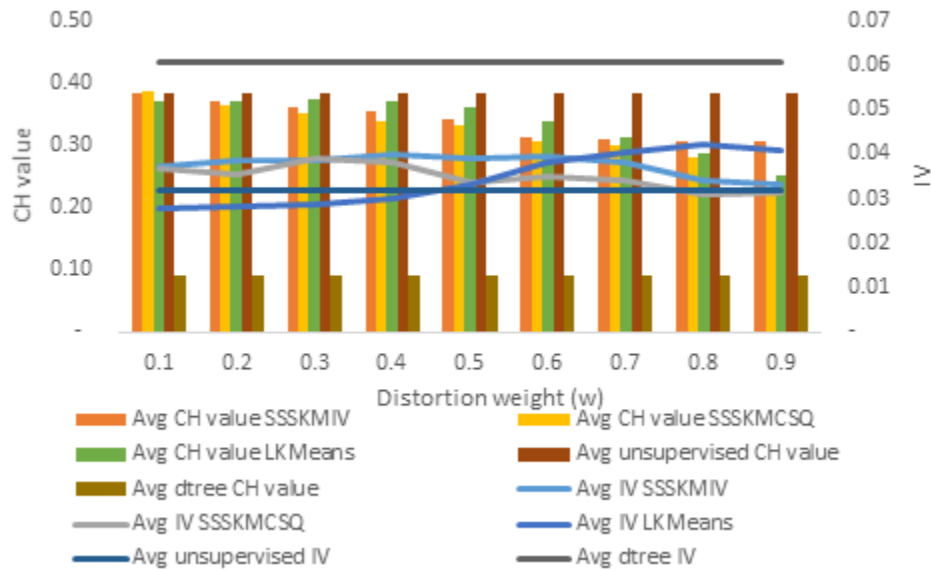


FIGURE 7.68: IV and CH values compared to other segmentation techniques

Correlation between IV and Gini improvement is again confirmed in Figure 7.69 and Table D.77.

From Figure 7.70 and Table D.78 we see that the LK-Means algorithm obtains the best Gini improvement for low values of K , even though SSSKMIV and unsupervised segmentation ultimately obtains better Gini improvement for $K > 7$. It is clear from the results seen that, similar to the other semi-supervised segmentation techniques, LK-Means delivers good results consistently.

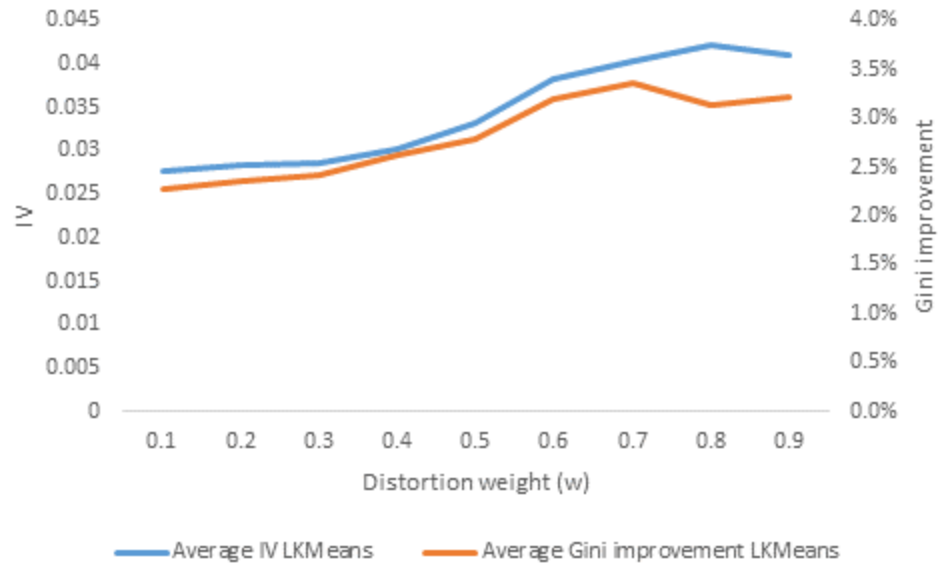


FIGURE 7.69: Comparison of IVs obtained versus Gini improvement

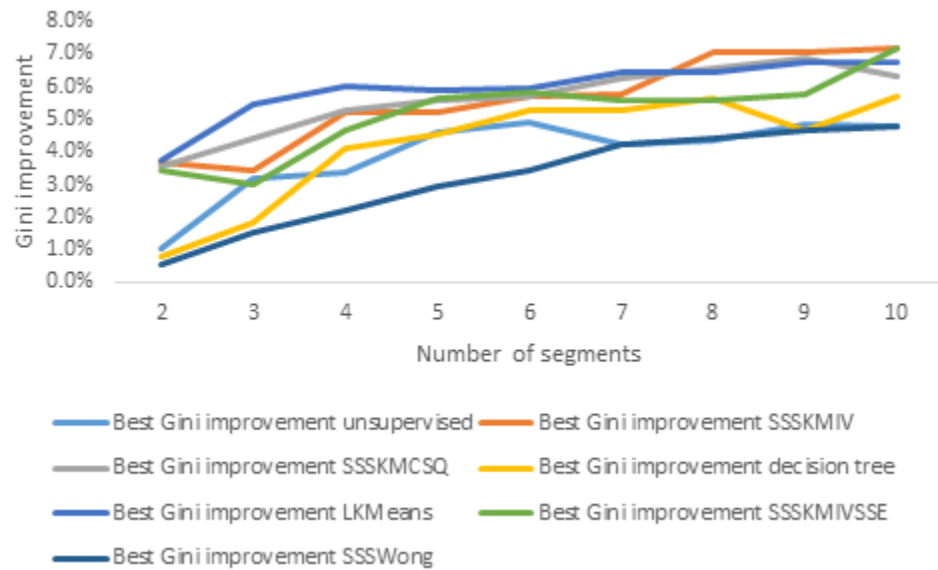


FIGURE 7.70: Gini improvement by number of segments

7.12 Summary of LK-Means results

When used as a method to segment data sets for predictive modelling, LK-Means performed well on the data sets tested in Section 7.11. We have observed many similarities in the LK-Means results when compared to the results from SSSKMIV and SSSKMCSQ. Most notable was the correlation that exist between the IV values obtained and the algorithm's success in improving the segmented models' overall Gini. This further implies that IV could be used as a quick measure to determine the optimal value for w .

LK-Means performed the best of all methodologies as measured by Gini improvement on the wine quality data set. It also performed well on the protein tertiary structure data set for low values of K , and on the claim prediction data set, also for low values of K . Given these results, LK-Means is a viable option to consider when segmenting for predictive modelling. The main disadvantage of the algorithm is, however, that it can only work on data sets with discrete targets, and not on data sets where targets are continuous.

7.13 Conclusion

In this chapter we explored two additional semi-supervised segmentation techniques that were developed to address some of the weaknesses of the SSSKM algorithms we observed in Chapter 5. We also considered the supervised clustering technique, LK-Means and augmented it to be comparable to the techniques investigated in this study. We applied these techniques to the same data sets that were analysed in Chapter 5 and compared the results with those obtained by the other segmentation techniques.

Table 7.1 summarises the performance of all the segmentation methodologies tested. The table contains the ranking of each of the methodologies on each of the data sets. The ranking was determined by comparing the best improvement in Gini for each of the methodologies for every data set analysed. It can be seen that SSSKMIV continues to rank as the most consistent performer on the data sets, with decision trees and LK-Means sharing the second position. SSSWong performed the worst on every data set.

	Direct Mar- keting data set	Protein ter- tiary struc- tures data set	Credit appli- cation data set	Wine quality data set	Chess (king- rook vs. king) data set	Claim pre- diction data set	Average
SSSKMIV	3	1	2	3	4	2	2.5
Tree	1	6	1	6	1	5	3.3
LK-Means	5	3	5	1	2	4	3.3
SSSKMCSQ	2	4	4	4	5	3	3.7
SSSKMIV_{SSE}	6	2	3	5	6	1	3.8
Clustering	4	5	6	2	3	6	4.3
SSSWong	7	7	7	7	7	7	7.0

TABLE 7.1: Performance ranking of all segmentation techniques on the data sets analysed

We contributed the following in this chapter:

- We explained the rationale for developing a semi-supervised algorithm with segment size equality.
- We described the objective function of the $\text{SSSKMIV}_{\text{SSE}}$ algorithm as well as how it is applied.
- We applied the $\text{SSSKMIV}_{\text{SSE}}$ algorithm to the data sets analysed in Chapter 5, and compared the results with those obtained by the other segmentation techniques, where possible. The algorithm obtained the best results in three out of the six data sets for at least one value of K .
- We explained the rationale for developing a density based semi-supervised algorithm using Wong's density clustering algorithm.
- We described the objective function of the SSSWong algorithm and indicated how it connects to the objective function used in the SSSKM algorithms
- We explained the design and operation of the SSSWong algorithm in five steps and provided more detail on each of the steps.
- We applied the SSSWong algorithm to the data sets analysed in Chapter 5 and compared the results with those obtained by the other segmentation techniques. The algorithm obtained the best IVs between segments for four of the six data sets, but did not manage to improve model accuracy as measured by Gini improvement.
- We augmented the LK-Means methodology, proposed by Peralta et al. (2013), by adding the ability to apply results obtained on a development set to a validation set, which ensured the results were comparable to other methodologies used in this study.
- We applied the LK-Means algorithm to the data sets analysed in Chapter 5 and compare the results with those obtained by the other segmentation techniques. The algorithm performed well on a number of data sets and it outperformed all other segmentation techniques on the wine quality data set.

Chapter 8

Conclusion and future research

8.1 Introduction

The problem statement of this thesis is to develop, define and comprehensively analyse a semi-supervised segmentation algorithm as an alternative to current segmentation algorithms that not only considers the explanatory variables or the target variable, but optimises both simultaneously during segmentation for predictive modelling.

This was achieved by addressing the primary objective as well as the five secondary objectives. The primary objective was addressed by defining the newly developed SSSKM algorithm's objective function mathematically and also describing how the algorithm works in Chapter 3.

The first secondary objective was achieved in Chapter 4, by exploring how the addition of a supervised element to the objective function of the k-means clustering algorithm (the SSSKM algorithm) alters the behaviour of the algorithm.

The second and third secondary objectives were achieved in Chapter 5, by applying the SSSKM algorithm to six data sets from different disciplines in the industry on which predictive models can be developed. The characteristics of the resulting segments were analysed and the improvement in model accuracy assessed. These improvements were compared to the improvements (on the same data sets) achieved by models developed on segments informed by other segmentation techniques that are commonly used (i.e. supervised and unsupervised techniques).

Finally, the accuracy of the segmented models was compared to the accuracy achieved by a number of popular non-linear modelling techniques on the same data sets and the results were analysed (addressing the third secondary objective).

Addressing the fourth secondary objective in Chapter 6, enabled us to answer some of the questions raised regarding the characteristics that cause one segmentation technique to outperform another, depending on the data used. Data sets were generated with known segment assignments, segment and variable characteristics as well as known target-variable dependencies. The impact of these characteristics on the success (or failure) of the various segmentation techniques was subsequently measured and analysed.

The last secondary objective was achieved in Chapter 7, by exploring two variations of the semi-supervised segmentation methodologies that are aimed at addressing some of the weaknesses of the SSSKM algorithm that were observed. Both these methodologies were applied to the same data sets used in Chapter 5 and their performance was compared to the methodologies already assessed.

This remainder of Chapter 8 highlights the key findings and conclusions reached throughout the thesis (Section 8.2) and provides suggestions for further research (Section 8.3).

8.2 Key findings

Chapter 2 provided an overview of the statistical methodologies and the terminology used within this thesis (literature overview). We found that there are some differences and some similarities between supervised and unsupervised clustering and that these in turn differ from semi-supervised segmentation in a number of ways. We found previous studies that have certain similarities to the work performed here, but we also discovered that all of these differ in a variety of ways.

In Chapter 3, we defined the objective function of the SSSKM algorithm. The objective function consists of a supervised and an unsupervised element and the algorithm is based on the standard k-means algorithm. Through describing the algorithm in ten steps, we also brought to light that the algorithm differs from standard k-means in how it functions. This is a newly developed algorithm and our main research contribution. The flow of the algorithm is again depicted in Figure 8.1

We applied the SSSKM algorithm to simple, two dimensional data sets in Chapter 4 and compared the output of the SSSKM algorithm to that of ordinary k-means. We found that the addition of the objective function improved the accuracy with which segments are identified, even if they are difficult to recognise spatially. We also determined that the SSSKM algorithm can overcome some of the known weaknesses of standard k-means, in particular its tendency to form segments that are equally sized as well as its reliance on globular segments. This chapter also offers the first opportunity to observe the effects of overfit for values of w that are too high.

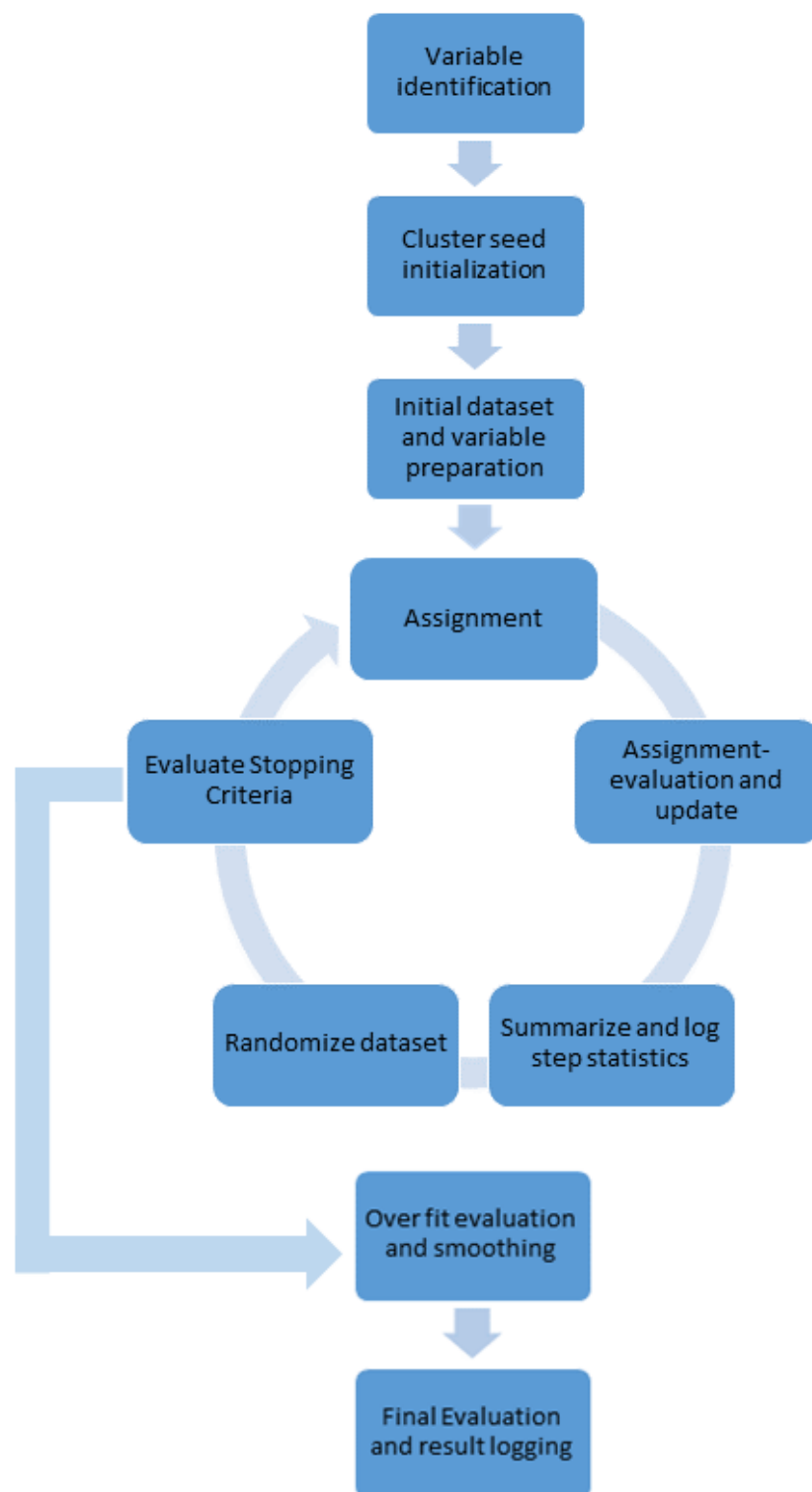


FIGURE 8.1: Functional flow of the single K-means semi supervised segmentation algorithm

In the first part of Chapter 5, we applied the SSSKM algorithm to actual data sets used for predictive model development to compare the accuracy of the models developed on its segments against segmentation methodologies already used in the industry.

We learned that there is not one single segmentation algorithm that always outperforms when segmenting for predictive modelling. Depending on the characteristics of the data set, one method may outperform another. We also realised that a single methodology will not outperform other methodologies on a specific data set each and every time, but that the results may differ depending on the number of segments required. In most cases, an increase in the value of w also coincides with an increase in the separation of the target variable, however, for high values of w , some degree of overfit was observed.

We discovered that in general, the SSSKMCSQ algorithm tends to override more segment assignments for lower values of w , which causes it to achieve, in some cases, higher target separation earlier. It was, however, also clear that the SSSKMCSQ algorithm generally creates segments with lower CH values in comparison to the SSSKMIV algorithm and that it was never able to outperform the SSSKMIV algorithm, when measured based on improvement in Gini. Note again that this conclusion can be strengthened by testing statistical significance.

We observed that the improvement in target rate separation on the validation set is, in many cases, correlated with the Gini improvement achieved, indicating that this measure may be used as an indication of the best value of w without the need to develop models.

We concluded that the SSSKM algorithms, although not always outperforming unsupervised and supervised techniques, can be a valuable tool to improve segmentation for predictive modelling and do in many cases provide better segmentation than the traditional segmentation methodologies. Table 8.1 summarises the results obtained in the first part of Chapter 5 again. It shows that the decision tree performs best for the direct marketing, credit applications and chess (king-rook vs. king) data sets, whilst the SSSKM algorithms performs best on the remaining data sets.

The second part of Chapter 5 compared the results of the SSSKM algorithm to a number of popular non-linear modelling techniques. We determined that no single methodology dominated the top position, but that the four segmentation-based methodologies explored always performed well (and in some cases the best), whilst other methodologies demonstrated inconsistency. The SSSKMIV algorithm proved to perform most consistently on all the data sets (as measured by its average performance ranking). A great many examples were found where segmentation-based methodologies can achieve acceptable results with a relatively low number of segments. In these cases, the reduced

Data set	Unsegmented Gini on validation set	Range of best improvements in Gini	Best suited methodology (in general)
Direct marketing	22%	8%-12%	Decision tree
Protein tertiary structures	59%	2%-11%	SSSKM
Credit applications	38%	23%-33%	Decision tree
Wine quality	61%	4%-6%	SSSKM
Chess (king-rook vs. king)	71%	5%-17%	Decision tree
Insurance claim prediction	8%	4%-7%	SSSKM

TABLE 8.1: Summary of results

complexity of segmented logistic regression models can be considered an advantage. The results are once again set out in Table 8.2.

Data set	Best methodology	Position of best segmentation-based methodology
Direct marketing	Gradient boosting (random forests)	2
Protein tertiary structures	Memory based reasoning (KNN)	2
Credit application data	Neural networks	2
Wine quality	Unsupervised segmentation (k-means) with logistic regression	1
Chess (king-rook vs. king)	Decision trees	3
Insurance claim prediction	SSSKMIV segmentation with logistic regression	1

TABLE 8.2: Summary of results of alternative techniques compared to segmentation with logistic regression

In Chapter 6, we simulated data sets for predictive modelling in such a way that known segmentation techniques can be applied to it successfully. We saw that obtaining the best IV or CH values is not necessarily an indicator of which technique will perform best, when measured by improvement in Gini.

We, furthermore, found that data sets that consist of complex underlying segments and are described by many variables are not well suited to decision tree segmentation. The segmentation will be more successful if unsupervised clustering or SSSKMIV techniques are used. Conversely, decision tree segmentation is more successful when segments are

	IV group	Low CH group	Mid CH group	High CH group
Complexity (O = 5)	0.0-0.05	SSSKMIV	SSSKMIV	SSSKMIV
	0.05-0.5	SSSKMIV	Tree	Tree
	0.5-0.8	Tree	Tree	Tree
	0.8-Inf	Tree	Tree	Tree
Complexity (O = 10)	0.0-0.05	Tree	SSSKMIV	SSSKMIV
	0.05-0.5	SSSKMIV	SSSKMIV	SSSKMIV
	0.5-0.8	SSSKMIV	Tree	SSSKMIV
	0.8-Inf	SSSKMIV	Tree	Tree
Complexity (O = 15)	0.0-0.05	SSSKMIV	SSSKMIV	SSSKMIV
	0.05-0.5	SSSKMIV	SSSKMIV	SSSKMIV
	0.5-0.8	SSSKMIV	SSSKMIV	SSSKMIV
	0.8-Inf	SSSKMIV	SSSKMIV	SSSKMIV
Complexity (O = 15) Only low IVs	0.0-0.01	SSSKMIV	Clustering	SSSKMIV
	0.01-0.02	Clustering	SSSKMIV	Clustering
	0.02-0.03	Clustering	SSSKMIV	Clustering
	0.03-0.05	SSSKMIV	SSSKMIV	Clustering

TABLE 8.3: Summary of best performing segmentation methodology for all characteristics explored

simple (i.e. not described by many explanatory variables) and target rates between segments differ substantially. There were not a great number of instances where unsupervised segmentation performs best, but closer investigation proved that it can outperform both SSSKMIV and decision trees when segments are complex, yet differ very little in event rates. A summary of the performance of the three segmentation techniques that were compared is provided in Table 8.3

Similar to the findings in Chapter 5, we again found that the SSSKMIV algorithm consistently performs well within a large range of data set characteristics and often outperforms known techniques.

We discovered that the best value for w for the SSSKMIV algorithm is determined by the underlying characteristics of the data set. Segments that have larger IVs (i.e. differ significantly in event rates) tend to require larger values for w . We concluded that the best value for w will need to be iteratively determined for each data set, since no optimal value can be determined that will work optimally for all data sets. However, in our experience (analysing the data sets throughout this thesis) values of $w \leq 0.5$ generally work well.

In the first part of Chapter 7, we introduced the SSSKMIV_{SE} algorithm that adds a further element to the objective function, namely to encourage the algorithm to form segments that are equal in size. After comparing the results of the SSSKMIV_{SE} algorithm to other results already obtained, we learned that the SSSKMIV_{SE} algorithm is able to improve segment size equality for most data sets. We did, however, see that

segment size equality weights that are too high can also result in overfit (similar to high values of w), which causes segment size equality to decrease.

We learned from the results that the SSSKMIV_{SSE} algorithm often finds higher IVs for low values of w when compared to the other SSSKM algorithms. We suggested that this may be due to the increased statistical robustness of the segments, which reduces overfit for the target variable.

Although higher IVs were often found, it at times came at the expense of lower CH values. There were, however, some cases where IVs improved without a noticeable decrease in CH values.

When comparing the success of the SSSKMIV_{SSE} algorithm as measured by the improvement in validation set Gini, we found that there are some instances where the algorithm outperforms all other segmentation algorithms considered, and that the algorithm shows the same consistency observed from the other SSSKM algorithms.

The second part of Chapter 7 introduced the SSSWong algorithm that is based on density clustering and aims (amongst other things) to address the tendency of the SSSKM algorithms to overfit. When applying the SSSWong algorithm to the same data sets analysed in Chapter 5, we saw that the SSSWong algorithm is able to avoid IV overfit across the board. We also found that for most data sets the SSSWong algorithm finds very high IVs. In many cases even higher IVs than the decision tree were found, which led us to conclude that the algorithm may be well suited as a predictive model on its own in some applications.

The SSSWong algorithm did not, however, perform well as a “segmentor” for predictive models on any data set and it produced the worst overall results in all cases. We reasoned that this may be due to the fact that some of the weaknesses of k-means were already addressed by SSSKM (as seen in Chapter 3) and that the additional complexity of the SSSWong algorithm, therefore, adds no additional benefit.

The third part of Chapter 7 augmented the LK-Means methodology, proposed by Peralta et al. (2013), so that it can be used as a segmentor for predictive modelling. The LK-Means methodology performed well on the data sets and outperformed all other modelling methodologies on the wine quality data set. Again we observed the tendency that the improvement in target rate separation (as measured by IV) is correlated with the improvement in Gini, strengthening the notion that this measure may be used as an indication of the best value of w without the need to develop models.

Table 8.4 summarises the results of the segmentation methodologies analysed and reflects that the SSSKMIV methodology performs most consistently on the data sets used. The

table does, however, also confirm that no single segmentation algorithm will always outperform another, which reiterates the finding that the underlying characteristics of a data set will determine which segmentation methodology performs best. Our conclusion was, therefore, reaffirmed that having more segmentation methodologies to consider is likely to enhance final model accuracy in many applications. Also note that on average SSSKMIV ranked the best at an average rank of 2.5.

	Direct Mar- keting data set	Protein ter- tiary struc- tures data set	Credit appli- cation data set	Wine quality data set	Chess (king- rook vs. king) data set	Claim pre- diction data set	Average
SSSKMIV	3	1	2	3	4	2	2.5
Tree	1	6	1	6	1	5	3.3
LK-Means	5	3	5	1	2	4	3.3
SSSKMCSQ	2	4	4	4	5	3	3.7
SSSKMIV_{SSE}	6	2	3	5	6	1	3.8
Clustering	4	5	6	2	3	6	4.3
SSSWong	7	7	7	7	7	7	7.0

TABLE 8.4: Performance of all segmentation techniques on the data sets analysed

8.3 Future work

As indicated by the problem statement, the focus of this study is the development, definition and analysis of a semi-supervised segmentation algorithm to broaden the tools available for statistical analysis of data sets for predictive modelling.

Not all avenues of possible research could be explored within this study and some are, therefore, left for future work. One aspect that falls outside the scope of this research is the efficiency and execution time of the semi-supervised segmentation algorithm. Because the optimal value of inputs, like the supervised weight (w), cannot be determined beforehand and differs for every data set, it requires running numerous iterations with different values of w . Only upon investigating a range of different values of w can one make an informed decision on the best value. Furthermore, the algorithm's tendency to converge to local optima means that one also needs to perform more than one iteration for each value of w . If the algorithm can be rewritten in software that allows it to perform more rapidly, this would be less of an issue. At present it means that large data sets may take a number of days to be fully analysed with the algorithm.

Linked to the previous point, is further study into segment initialisation methods for the semi-supervised segmentation algorithms. We mentioned in Section 3.7 that cluster seed initialisation is a large area of research for the k-means algorithm (see again (Kang and Cho, 2009, Khan and Ahmad, 2004, Pena et al., 1999, Redmond and Heneghan, 2007)), but that the methods developed do not take the addition of the target variable into account and could thus not be used in this study. However, further research into segment initialisation, specifically for the SSSKM algorithm, could reduce the number of iterations required for analysis and, therefore, improve the algorithm's utility.

Conducting research into determining a narrower band for the value of w could also be of value and reduce the number of required iterations. We have seen in the study that data sets that have segments with higher IVs tend to require higher values of w (see Section 6.5.5), but we did not explore whether there may be some way to derive an optimal value from data set characteristics. We also observed a correlation between the target separation obtained between segments and the eventual Gini improvement for a number of data sets and segmentation methodologies. This may be worth investigating further and could lead to refined ways of narrowing the band of optimal values of w for a particular data set.

We only analysed data sets with binary target variables in this study, since the use of IV requires target variables to be binary. Further studies could focus on the impact of the algorithm on models with continuous target variables by means of the SSSKMCSQ algorithm, which does not require the target to be binary. This also sets the SSSKM algorithm apart from other methods, like LK-Means.

The SSSWong algorithm explored in Chapter 7 did not provide satisfactory results and can be further enhanced in future studies:

- We observed that the algorithm tends to form very small segments in some instances, which is not ideal when segmenting for predictive modelling. The algorithm may benefit from introducing segment size equality in a similar way to what was done with the SSSKMIV_{SSE} algorithm. This could assist in obtaining segments that are large enough to facilitate increased robustness of the models that are developed.
- The initial segments that are formed for the SSSWong algorithm, by the k-means algorithm, is completely unsupervised, which means that the first step of the SSSWong algorithm has no supervised element to guide initial segment formation. It may be that the algorithm could benefit from rather using one of the SSSKM algorithms to form initial segments, which will mean the supervised element is introduced from the very start.

In some disciplines, analysts deal with sparse data sets where the number of explanatory variables available may far exceed the observations. It may be worthwhile to explore the performance of the developed segmentation techniques on such sets, especially since it was found that the SSSKM algorithms do well when there are a greater number of explanatory variables (see Section 6.6).

Appendix A

Software examples of segmentation

A.1 Case study of unsupervised segmentation in the industry: SAS[®] Enterprise Miner[™]

As an industry example of where clustering is used for segmenting predictive models, we will discuss the SAS[®] Enterprise Miner[™] software. Enterprise Miner[™] 12.1 offers a "Cluster" node within its "Explore" tab which enables the user to perform cluster analysis on a data set. The role of clustering within the model development process can be set to a number of options:

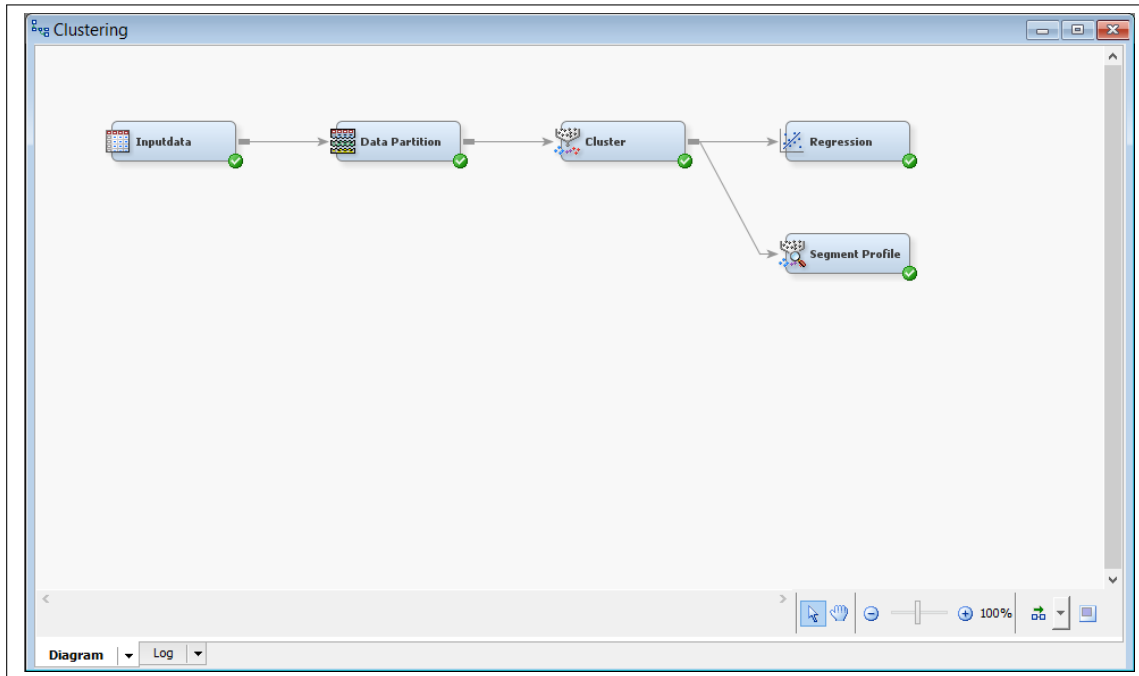
- Segment
 - When set to this, the clustering analysis is performed with the goal of segmenting the observations of the data set into distinct groups with the aim of developing predictive models on each of the groups separately, in line with the definition of segmentation used in this thesis.
- ID
 - The ID setting will divide the input set into clusters so that all observations belonging to a cluster can be identified in future. The goal usually is to be able to compare cluster specific statistics when needed.
- Input
 - When set to input, the result of the cluster analysis can be used as independent characteristic input into a subsequent predictive model development.

- Target
 - Setting the role to target will mean that the output of the clustering analysis will be used as a dependent, or target variable in subsequent predictive model developments.

The default value for the role of the node is segmentation, and this is also what the software intends as the model's main purpose from its documentation. The node provides a number of additional options that can be controlled by the user to allow control over the output that is obtained. Examples are:

- Standardisation.
 - In line with standard practice, the SAS[®] software recommends that input characteristics be standardised. It offers two types of standardisation as well as the option not to standardise.
- Maximum and minimum number of segments allowed.
 - This provides the user with the option to force the result to be within the specified number of segments. Even though the underlying data may be segmented into many possible groups, it is impractical to develop and maintain many predictive models in practice.
- Criteria used to determine optimal number of segments.
 - The node uses the Cubic Clustering Criterion (CCC), which was developed by the SAS institute (Sarle, 1983) to determine the optimal number of segments based on comparing observed R^2 values to estimated expected R^2 values. The user can provide the value for the CCC at which the optimal number of segments are reached.
- Clustering method used.
 - Three options exist, namely Average (Sokal, 1958), Ward (Ward Jr, 1963) or Centroid (Sokal, 1958). These indicate the linkage methods used for clustering.

Once the clustering analysis has been performed, Enterprise MinerTM provides the user with a number of graphical statistics that assist the user to understand the nature of the segments that are proposed. Figure A.1 shows a snapshot of the nodes used and the layout of the diagram in Enterprise MinerTM. In this diagram, an example data set is analysed with the clustering node.

FIGURE A.1: Enterprise MinerTM - diagram layout

The graphs shown in Figures A.2 to A.4 show some of the available graphical statistics offered by the software. Figure A.2 shows the relative size of each segment compared to the overall population. Figure A.3 shows the graph of the CCC value for each number of segments analysed. It can be seen that the value peaks at six segments - this is why the Enterprise MinerTM software selected six segments as the most optimal number. A very useful graphical statistic is shown in the graph in Figure A.4. From this graph, the difference in distribution of all the input characteristics are shown between segments. From the graph it is clear to see that distributions of all characteristics differ substantially between segments.

A.2 Case study: Fico model builder

In its white paper named "Using segmented models for better decisions" (Fico, 2014), Fair Isaac Corporation (FICO) describe the technique they developed to segment for predictive model development. The information is provided here to illustrate how decision trees are used for segmentation in the industry, and is entirely based on their paper.

FICO is a leading analytics company internationally and has a presence in more than 90 countries. They have developed software called "Model Builder", which provides

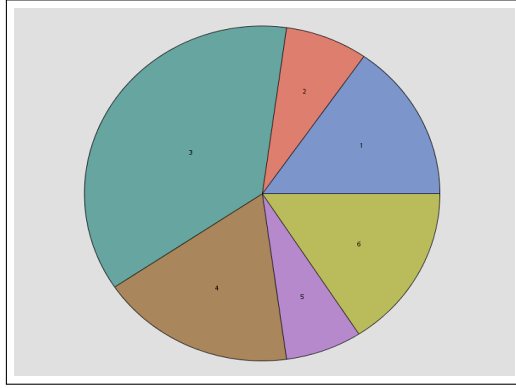


FIGURE A.2: Segment size

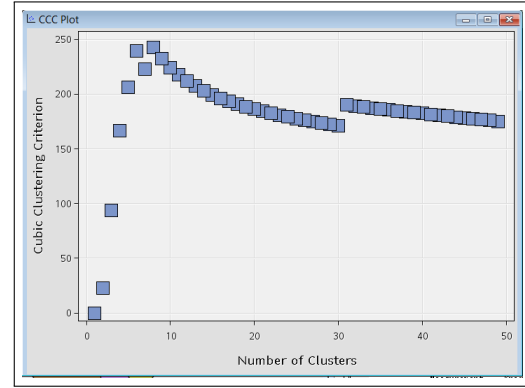


FIGURE A.3: CCC graph

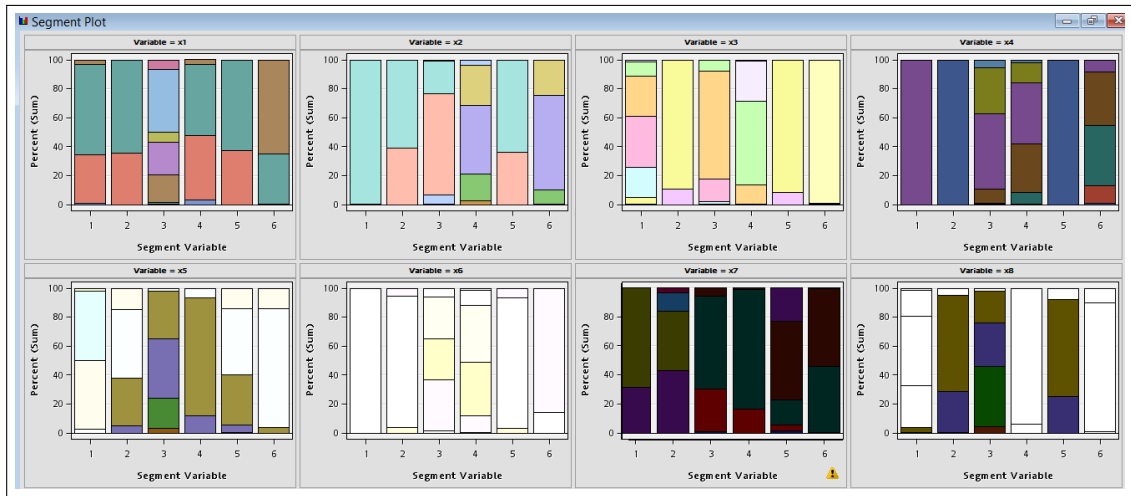


FIGURE A.4: Segment characteristic distributions

statisticians with graphical tools to aid in predictive model development. One of the tools they provide in this software is a segmentation "search algorithm".

The technique they developed makes use of decision trees to find optimal segmentation, but the way in which the decision trees are used provides the model developer with a lot of insight into the power of the segments used and limits the impact of the locally optimised "greedy" algorithm of the decision tree. They use the flow diagram in figure A.5 to describe their process. We will summarise the role of each of the nodes here.

- Input.
 - The development and validation data sets need to be provided along with the required settings and a model that was developed with no segmentation. All subsequent results are compared to this "Parent" model.
- Initialize.
 - As discussed in Section 2.3, there are many reasons why segmentation is applied during model development and not all reasons are based on statistics.

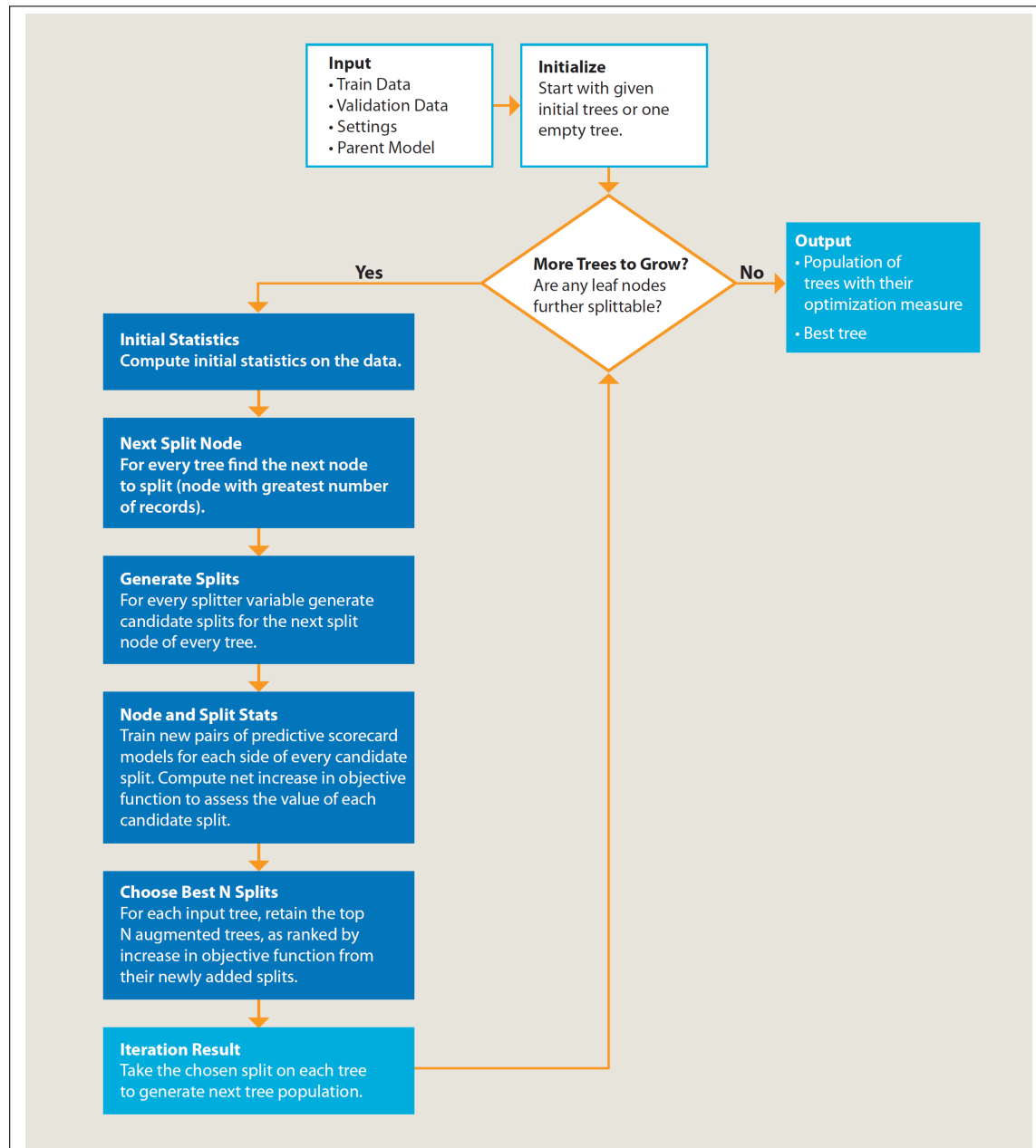


FIGURE A.5: FICO Model Builder - Segmentation process

Therefore, the algorithm allows the user to force segmentation on certain characteristics initially. The algorithm initialises the segmentation process with the specified segmentation already applied. If no initial segmentation is required, the process starts with the total population.

- Assess if more leafs are possible.
 - Since this process is iterative, the segmented population is assessed for the possibility of further splits - this is based on settings such as minimum segment sizes, etc.

- Initial Statistics.
 - Standard statistics are calculated based on the available data.
- Next split node.
 - This step identifies the node with the largest population for further segmentation. This is a slight deviation from most decision tree methodologies which use split criterion, not segment size, to decide whether a node should be further split.
- Generate splits.
 - Because the ultimate goal of the segmentation exercise is not to obtain the best local optimisation with regards to the target variable, but to create the best segmented model, the FICO algorithm assesses *all* identified splitter variables to measure the impact of further segmentation. This limits the impact of the greedy local optimisation of the decision tree. However, splits are still generated based on the target variable alone.
- Node and split statistics.
 - The automated Model Builder software is able to automatically run models on each of the identified segmentations from the previous node. An objective function (like Gini, K statistics, etc.) is identified and the splits are evaluated based on the net lift in the objective function it produces.
- Choose best N splits.
 - The best number of splits as evaluated by the objective function are retained for further segmentation analysis.
- Iteration result.
 - The new populations for which segmentation should be considered are identified for each of the options selected.
- Output.
 - Once the iterative process is complete and no further segmentation possible, the tool produces summary statistics for all possible segmentations and provides the model developer with an overview of the impact on the objective function for each of the segmentation frameworks. The model developer weighs up complication added to the net lift in objective function attained by segmentation. Once the developer decides on a specific framework, the next step is to develop robust industry models on each of the segments.

The segmentation scheme used in Model Builder is a good example of how supervised learning is used in the industry in a practical way to segment populations with the aim of developing predictive models. The iterative process described here will also work for other techniques such as unsupervised and semi-supervised segmentation.

Appendix B

Appendix for additional information in Chapter 5

This appendix contains relevant additional information as referred to in Chapter 5.

Variable name	Variable description	Data type
BALANCE_SCORE	A score between 1 and 5, indicating the size of the customer's total exposure to the bank relative to other customers. 1 being low and 5 being high	Integer
CIS_NIR_AMOUNT	The net interest revenue generated by the customer in the latest month	Real
COMPOSITE_SCORE	A score combining elements such as the customer's exposure to the bank, tenure with the bank, revenue generated and number of products, 5 being good, 1 being bad	Integer
FACTORCOMP	Similar to the composite score, but with real values instead of integer values	Real

TABLE B.1: Data dictionary of the direct marketing data set

Variable name	Variable description	Data type
DaysSinceCNTCT	Number of days since the customer was previously solicited for a new product with the bank	Integer
DaysSinceOPND	Number of days since the customer opened his first product with the bank	Integer
FREQABF	Number of asset based finance products the customer has with the bank	Integer
FREQCA	Number of current accounts the customer has with the bank	Integer
FREQCAMS	Number of credit card products the customer has with the bank	Integer
GENERIC_CSTMRL_SCORE	A generic scorecard developed on the customer to estimate profitability	Integer
GRP_NIR_AMOUNT	Another calculation of the customer's net interest revenue generated	Real
GRPTOTSAL	The customer's net salary (after tax and employer deductions)	Real
HasBusT	A binary variable indicating whether the customer has a business telephone contact number on the system	Binary
MAINTOPNDDIFF	The number of days between the date the customer first opened a product with the bank and the latest date that the customer's details were updated	Integer
NETBALCAMS	Balance of the customer's credit card products	Real
NETBALHL	Balance of the customer's home loan products	Real
NETBALSA	Balance of the customer's savings account products	Real
PRODUCT_SCORE	A score between 1 and 5, indicating the number of products the customer has with the bank relative to other customers, 1 being few products and 5 being many	Integer
REVENUE_SCORE	A score between 1 and 5, indicating the customer's total revenue generated relative to other customers, 1 being low revenue and 5 being high revenue	Integer
RISK_SCORE;	A score indicating the default risk of the customer, higher scores indicating better risk	Integer
SEG_L1_3	Product level segmentation of the customer based on the bank's internal segmentation framework	Integer

TABLE B.2: Data dictionary of the direct marketing data set (continued)

Variable name	Variable description	Data type
TENURE_SCORE	A score between 1 and 5, indicating the customer's tenure with the bank relative to other customers, 1 being new and 5 being a long standing customer	Integer
TOTSALAMT	Another calculation of the customer's net salary	Real
TRANSACTION_SCORE	A score between 1 and 5, indicating the number of transactions the customer performed in the last month relative to other customers, 1 being few and 5 being many	Integer

TABLE B.3: Data dictionary of the direct marketing data set (continued)

Variable name	Variable description	Data type
F1	Total surface area	Real
F2	Non polar exposed area	Real
F3	Fractional area of exposed non-polar residue	Real
F4	Fractional area of exposed non-polar part of residue	Real
F5	Molecular mass weighted exposed area	Real
F6	Average deviation from standard exposed area of residue	Real
F7	Euclidian distance	Real
F8	Secondary structure penalty	Integer
F9	Spatial distribution constraints (N, K value)	Real

TABLE B.4: Data dictionary of the protein structure data set obtained from UCI

Variable name	Variable description	Data type
RevolvingUtilizationOfUnsecuredLines	Total balance on credit cards and personal lines of credit except real estate and no installment debt like car loans, divided by the sum of credit limits	percentage
Age	Age of borrower in years	integer
NumberOfTime30-59DaysPastDueNotWorse	Number of times borrower has been 30-59 days past due but no worse in the last 2 years	integer
DebtRatio	Monthly debt payments, alimony and living costs divided by monthly gross income	percentage
MonthlyIncome	Monthly income	real
NumberOfOpenCreditLinesAndLoans	Number of open loans (installment like car loan or mortgage) and lines of credit (e.g. credit cards)	integer
NumberOfTimes90DaysLate	Number of times borrower has been 90 days or more past due	integer
NumberRealEstateLoansOrLines	Number of mortgage and real estate loans including home equity lines of credit	integer
NumberOfTime60-89DaysPastDueNotWorse	Number of times borrower has been 60-89 days past due but no worse in the last 2 years	integer
NumberOfDependents	Number of dependents in family excluding themselves (spouse, children etc.)	integer

TABLE B.5: Data dictionary of original data set obtained from Kaggle

Variable name	Variable description	Data type
fixed acidity	grams of tartaric acid per dm^3	Real
volatile acidity	grams of acetic acid per dm^3	Real
citric acid	grams of citric acid per dm^3	Real
residual sugar	grams sugar per dm^3	Real
chlorides	grams sodium chloride per dm^3	Real
free sulfur dioxide	milligrams of free sulfur dioxide per dm^3	Real
total sulfur dioxide	milligrams of total sulfur dioxide per dm^3	Real
density	wine density in grams per cm^3	Real
pH	the pH level of the wine	Real
sulphates	grams pottassium sulphate per dm^3	Real
alcohol	percentage alcohol per volume	Real

TABLE B.6: Data dictionary of the wine quality data set obtained from UCI

Variable name	Variable description	Data type
WKF	White king file: A character value a-h indicating the column of the white king's position	Char
WKR	White king rank: A numerical value 1-8 indicating the row of the white king's position	Integer
WRF	White rook file: A character value a-h indicating the column of the white rook's position	Char
WRR	White rook rank: A numerical value 1-8 indicating the row of the white rook's position	Integer
BKF	Black king file: A character value a-h indicating the column of the black king's position	Char
BKR	Black king rank: A numerical value 1-8 indicating the row of the black king's position	Integer

TABLE B.7: Data dictionary of the chess (king-rook vs. king) data set obtained from UCI

Variable name	Variable description	Data type
Var1	Continuous vehicle variable	Real
Var2	Continuous vehicle variable	Real
Var3	Continuous vehicle variable	Real
Var4	Continuous vehicle variable	Real
Var5	Continuous vehicle variable	Real
Var6	Continuous vehicle variable	Real
Var7	Continuous vehicle variable	Real
Var8	Continuous vehicle variable	Real
NVVar1	Continuous non-vehicle variable	Real
NVVar2	Continuous non-vehicle variable	Real
NVVar3	Continuous non-vehicle variable	Real
NVVar4	Continuous non-vehicle variable	Real

TABLE B.8: Available information for the claim prediction data set

Algorithm B.1 Preparing chess (king-rook vs. king) data

If $WKF = "a"$ **Then** $N_{WKF} = 1$
If $WKF = "b"$ **Then** $N_{WKF} = 2$
If $WKF = "c"$ **Then** $N_{WKF} = 3$
If $WKF = "d"$ **Then** $N_{WKF} = 4$
If $WKF = "e"$ **Then** $N_{WKF} = 5$
If $WKF = "f"$ **Then** $N_{WKF} = 6$
If $WKF = "g"$ **Then** $N_{WKF} = 7$
If $WKF = "h"$ **Then** $N_{WKF} = 8$

If $BKF = "a"$ **Then** $N_{BKF} = 1$
If $BKF = "b"$ **Then** $N_{BKF} = 2$
If $BKF = "c"$ **Then** $N_{BKF} = 3$
If $BKF = "d"$ **Then** $N_{BKF} = 4$
If $BKF = "e"$ **Then** $N_{BKF} = 5$
If $BKF = "f"$ **Then** $N_{BKF} = 6$
If $BKF = "g"$ **Then** $N_{BKF} = 7$
If $BKF = "h"$ **Then** $N_{BKF} = 8$

If $WRF = "a"$ **Then** $N_{WRF} = 1$
If $WRF = "b"$ **Then** $N_{WRF} = 2$
If $WRF = "c"$ **Then** $N_{WRF} = 3$
If $WRF = "d"$ **Then** $N_{WRF} = 4$
If $WRF = "e"$ **Then** $N_{WRF} = 5$
If $WRF = "f"$ **Then** $N_{WRF} = 6$
If $WRF = "g"$ **Then** $N_{WRF} = 7$
If $WRF = "h"$ **Then** $N_{WRF} = 8$

Algorithm B.2 Enumerating chess (king-rook vs. king) target variable

If *Outcome* = "zero" **Then** *NrMoves* = 0
If *Outcome* = "one" **Then** *NrMoves* = 1
If *Outcome* = "two" **Then** *NrMoves* = 2
If *Outcome* = "three" **Then** *NrMoves* = 3
If *Outcome* = "four" **Then** *NrMoves* = 4
If *Outcome* = "five" **Then** *NrMoves* = 5
If *Outcome* = "six" **Then** *NrMoves* = 6
If *Outcome* = "seven" **Then** *NrMoves* = 7
If *Outcome* = "eight" **Then** *NrMoves* = 8
If *Outcome* = "nine" **Then** *NrMoves* = 9
If *Outcome* = "ten" **Then** *NrMoves* = 10
If *Outcome* = "eleven" **Then** *NrMoves* = 11
If *Outcome* = "twelve" **Then** *NrMoves* = 12
If *Outcome* = "thirteen" **Then** *NrMoves* = 13
If *Outcome* = "fourteen" **Then** *NrMoves* = 14
If *Outcome* = "fifteen" **Then** *NrMoves* = 15
If *Outcome* = "sixteen" **Then** *NrMoves* = 16
If *Outcome* = "draw" **Then** *NrMoves* = 17

Field	Description
RunPostfix	Unique identifier of the run
LastDecChangePerc	Percentage of observations whose assignment was changed due to the addition of the supervised element (during the last run)
LastEventrate1-K	Event rate of each segment
PopPerc1-K	Percentage of the population in each segment
CHValue	Value of the CH measure (or pseudo-f statistic - see Section 2.12) of the segmentation
LastIV	Information value (see Section 2.11.1) of the segmentation
SegDistances	The Euclidean distances between each of the segment centroids
GiniImprovement	The improvement in Gini on the validation set
CombinedBuildGiniRef	The reference Gini on the development set with no segmentation
CombinedValidGiniRef	The reference Gini on the validation set with no segmentation
GiniImprovementDTree	The improvement in Gini on the validation set by using a decision tree to segment
DTreeCHValue	The value of the CH measure obtained by segmenting with a decision tree
DTreeIV	The IV obtained by segmenting with a decision tree
WoeWgt	The supervised weight applied

TABLE B.9: Statistics retained for comparing segmentation results

Segmentation method / su- pervised wgt	Percentage k means decisions over- ridden SSS- KMIV	Percentage k means decisions over- ridden SSSKM- CSQ	Average IV SSS- KMIV	Average CH value SSS- KMIV	Average IV SSS- KMCSQ	Average CH value SSSKM- CSQ
Unsupervised	NA	NA	0.076	0.162	0.073	0.162
0.1	1.68%	3.93%	0.072	0.162	0.079	0.162
0.2	4.95%	11.99%	0.081	0.162	0.081	0.156
0.3	10.56%	18.92%	0.091	0.157	0.084	0.151
0.4	15.33%	22.49%	0.091	0.153	0.085	0.149
0.5	23.3%	25.82%	0.090	0.150	0.091	0.149
0.6	30.12%	29.2%	0.090	0.147	0.097	0.145
0.7	35.07%	30.04%	0.090	0.146	0.097	0.143
0.8	37.69%	32.32%	0.092	0.145	0.096	0.141
0.9	41.15%	40.8%	0.094	0.143	0.092	0.132
Dtree	NA	NA	0.225	0.086	0.225	0.086

TABLE B.10: Comparison of segmentation characteristics by supervised weight for the direct marketing data set

Segmentation method / supervised wgt	Average Gini improvement SSSKMIV	Average Gini improvement SSSKMCSQ	Average IV SSSKMIV	Average IV SSSKMCSQ
Unsupervised	0.76% ¹	0.76%	0.076	0.073
0.1	0.29%	0.74%	0.072	0.079
0.2	0.06%	0.38%	0.081	0.081
0.3	0.36%	0.47%	0.091	0.084
0.4	0.41%	0.12%	0.091	0.085
0.5	-0.22%	0.22%	0.090	0.091
0.6	-0.58%	-0.15%	0.090	0.097
0.7	-0.28%	-0.76%	0.090	0.097
0.8	-0.43%	-0.59%	0.092	0.096
0.9	-0.52%	-1.35%	0.094	0.092
Dtree	2.82%	2.82%	0.225	0.225

TABLE B.11: Success of target separation compared to Gini improvement by supervised weight for the direct marketing data set

NrSegs	Best Gini improvement unsupervised	Best Gini improvement SSSKMIV	Best Gini improvement SSSKMCSQ	Best Gini improvement decision tree
2	3.74%	3.74%	3.74%	8.37%
3	4.00%	4.37%	4.12%	8.88%
4	3.12%	4.39%	4.56%	10.16%
5	3.53%	4.04%	3.69%	9.97%
6	4.20%	4.34%	4.26%	10.30%
7	2.08%	3.85%	4.87%	10.24%
8	4.55%	3.72%	5.40%	10.68%
9	5.13%	5.91%	7.16%	10.52%
10	4.45%	5.49%	6.21%	11.72%

TABLE B.12: Comparison of best improvements in Gini by number of segments for the direct marketing data set

¹The values of the unsupervised and decision tree segmentation is shown here under semi-supervised column names for illustrative convenience. Unsupervised segmentation implies that $w = 0$, whilst decision tree segmentation was performed to obtain the values in the "Decision tree" row

NrSegs	Average Gini improvement unsupervised	Average Gini improvement SSSKMIV	Average Gini improvement SSSKMCSQ	Average Gini improvement decision tree
2	2.13%	0.98%	0.80%	1.32%
3	1.59%	0.15%	0.18%	1.21%
4	-0.16%	-0.17%	0.21%	2.87%
5	-0.58%	-0.78%	-0.63%	2.80%
6	-0.34%	-0.48%	-0.71%	3.02%
7	-0.28%	-0.31%	-0.36%	3.22%
8	0.70%	-0.34%	-0.07%	3.48%
9	0.76%	0.06%	0.09%	3.81%
10	1.46%	-0.01%	-0.43%	3.61%

TABLE B.13: Comparison of average improvements in Gini by number of segments for the direct marketing data set

Segmentation method / supervised wgt	Percentage k means decisions over-ridden SSS-KMIV	Percentage k means decisions over-ridden SSSKM-CSQ	Average IV SSS-KMIV	Average CH value SSS-KMIV	Average IV SSS-KMCSQ	Average CH value SSSKM-CSQ
Unsupervised	NA	NA	0.360	0.776	0.365	0.775
0.1	6.72%	8.4%	0.447	0.773	0.456	0.766
0.2	13.64%	20.84%	0.500	0.747	0.616	0.713
0.3	18.69%	25.33%	0.562	0.721	0.717	0.699
0.4	22.32%	28.42%	0.630	0.709	0.767	0.679
0.5	29.09%	33.61%	0.699	0.696	0.801	0.663
0.6	35.42%	40.36%	0.825	0.646	0.822	0.616
0.7	38.42%	44.81%	0.806	0.635	0.796	0.587
0.8	41.28%	51.14%	0.734	0.637	0.727	0.539
0.9	45.69%	61.98%	0.654	0.622	0.692	0.408
Dtree	NA	NA	0.757	0.321	0.757	0.321

TABLE B.14: Comparison of segmentation characteristics by supervised weight for the protein tertiary structure data set

Segmentation method / supervised wgt	Average Gini improvement SSSKMIV	Average Gini improvement SSSKMCSQ	Average IV SSSKMIV	Average IV SSSKMCSQ
Unsupervised	4.34%	4.35%	0.360	0.365
0.1	4.58%	4.58%	0.447	0.456
0.2	4.90%	4.81%	0.500	0.616
0.3	4.88%	4.91%	0.562	0.717
0.4	4.90%	4.87%	0.630	0.767
0.5	5.01%	5.02%	0.699	0.801
0.6	5.08%	4.92%	0.825	0.822
0.7	4.98%	4.76%	0.806	0.796
0.8	4.75%	4.58%	0.734	0.727
0.9	4.38%	4.13%	0.654	0.692
Dtree	4.15%	4.15%	0.757	0.757

TABLE B.15: Success of target separation compared to Gini improvement by supervised weight for the protein tertiary structure data set

NrSegs	Best Gini improvement unsupervised	Best Gini improvement SSSKMIV	Best Gini improvement SSSKMCSQ	Best Gini improvement decision tree
2	0.72%	1.90%	1.84%	1.58%
3	2.12%	3.41%	3.30%	2.62%
4	2.54%	4.82%	5.18%	3.96%
5	4.76%	5.58%	5.49%	4.62%
6	5.44%	6.77%	6.14%	4.96%
7	6.45%	7.10%	7.11%	5.83%
8	6.83%	11.33%	7.40%	6.06%
9	7.63%	8.05%	8.00%	6.82%
10	7.84%	8.64%	8.95%	7.39%

TABLE B.16: Comparison of best improvements in Gini by number of segments for the protein tertiary structure data set

NrSegs	Average Gini improvement unsupervised	Average Gini improvement SSSKMIV	Average Gini improvement SSSKMCSQ	Average Gini improvement decision tree
2	0.72%	1.19%	1.18%	1.27%
3	1.93%	2.65%	2.60%	2.27%
4	2.47%	3.59%	3.57%	3.16%
5	3.62%	4.46%	4.30%	3.92%
6	4.78%	5.17%	5.06%	4.34%
7	6.07%	5.71%	5.67%	4.75%
8	6.26%	6.43%	6.33%	5.27%
9	6.57%	6.93%	6.71%	5.91%
10	6.70%	7.32%	7.14%	6.44%

TABLE B.17: Comparison of average improvements in Gini by number of segments for the protein tertiary structure data set

Segmentation method / supervised wgt	Percentage k means decisions overridden SSS-KMIV	Percentage k means decisions overridden SSSKM-CSQ	Average IV SSS-KMIV	Average CH value SSS-KMIV	Average IV SSS-KMCSQ	Average CH value SSSKM-CSQ
Unsupervised	NA	NA	0.187	0.274	0.186	0.280
0.1	12.09%	8.05%	0.213	0.285	0.218	0.277
0.2	17.12%	10.37%	0.215	0.283	0.235	0.276
0.3	30.39%	12.78%	0.239	0.282	0.241	0.275
0.4	37.27%	14.29%	0.247	0.281	0.266	0.270
0.5	34.75%	14.44%	0.262	0.274	0.260	0.255
0.6	14.25%	14.85%	0.319	0.227	0.309	0.226
0.7	11.75%	14.26%	0.311	0.226	0.308	0.227
0.8	12.53%	15.08%	0.307	0.229	0.306	0.220
0.9	13.95%	16.01%	0.306	0.222	0.304	0.222
Dtree	NA	NA	1.142	0.032	1.142	0.032

TABLE B.18: Comparison of segmentation characteristics by supervised weight for the credit scoring data set

Segmentation method / supervised wgt	Average Gini improvement SSSKMIV	Average Gini improvement SSSKMCSQ	Average IV SSSKMIV	Average IV SSSKMCSQ
Unsupervised	18.95%	18.99%	0.187	0.186
0.1	19.96%	19.37%	0.213	0.218
0.2	20.26%	20.57%	0.215	0.235
0.3	20.89%	20.96%	0.239	0.241
0.4	21.57%	21.59%	0.247	0.266
0.5	21.72%	22.09%	0.262	0.260
0.6	22.89%	22.92%	0.319	0.309
0.7	23.04%	22.94%	0.311	0.308
0.8	22.89%	22.85%	0.307	0.306
0.9	23.00%	22.83%	0.306	0.304
Dtree	26.20%	26.20%	1.142	1.142

TABLE B.19: Success of target separation compared to Gini improvement by supervised weight for the credit scoring data set

NrSegs	Best Gini improvement unsupervised	Best Gini improvement SSSKMIV	Best Gini improvement SSSKMCSQ	Best Gini improvement decision tree
2	22.39%	22.39%	18.70%	22.80%
3	23.01%	25.26%	24.24%	23.22%
4	23.06%	25.09%	24.55%	32.09%
5	23.01%	24.99%	23.92%	32.65%
6	24.48%	25.25%	24.98%	32.71%
7	25.02%	25.28%	26.81%	32.72%
8	24.96%	26.57%	26.06%	32.78%
9	25.03%	28.17%	27.31%	32.78%
10	24.69%	26.89%	26.55%	32.80%

TABLE B.20: Comparison of best improvements in Gini by number of segments for the credit scoring data set

NrSegs	Average Gini improvement unsupervised	Average Gini improvement SSSKMIV	Average Gini improvement SSSKMCSQ	Average Gini improvement decision tree
2	9.85%	13.31%	13.48%	22.80%
3	13.81%	19.78%	19.98%	23.14%
4	15.15%	22.01%	21.58%	23.22%
5	18.94%	22.52%	22.57%	23.73%
6	21.01%	23.46%	23.01%	25.57%
7	22.72%	23.85%	23.96%	27.15%
8	22.62%	23.79%	23.96%	28.80%
9	23.43%	23.80%	23.89%	30.21%
10	23.23%	23.70%	23.69%	31.20%

TABLE B.21: Comparison of average improvements in Gini by number of segments for the credit scoring data set

Segmentation method / supervised wgt	Percentage k means decisions over-ridden SSS-KMIV	Percentage k means decisions over-ridden SSSKM-CSQ	Average IV SSS-KMIV	Average CH value SSS-KMIV	Average IV SSS-KMCSQ	Average CH value SSSKM-CSQ
Unsupervised	NA	NA	0.642	0.374	0.638	0.374
0.1	2.82%	5.55%	0.658	0.375	0.713	0.375
0.2	6.17%	11.14%	0.693	0.375	0.752	0.367
0.3	8.81%	14.09%	0.718	0.371	0.778	0.356
0.4	11.74%	17.27%	0.753	0.363	0.819	0.348
0.5	17.95%	22.74%	0.831	0.357	0.868	0.339
0.6	20.89%	24.84%	0.928	0.348	0.909	0.326
0.7	22.43%	26.16%	0.907	0.344	0.903	0.322
0.8	27.1%	27.49%	0.832	0.335	0.891	0.313
0.9	28.69%	31.53%	0.776	0.338	0.861	0.295
Dtree	NA	NA	0.979	0.114	0.979	0.114

TABLE B.22: Comparison of segmentation characteristics by supervised weight for the wine quality data set

Segmentation method / supervised wgt	Average Gini improvement SSSKMIV	Average Gini improvement SSSKMCSQ	Average IV SSSKMIV	Average IV SSSKMCSQ
Unsupervised	3.19%	3.07%	0.642	0.638
0.1	3.07%	3.18%	0.658	0.713
0.2	3.09%	3.49%	0.693	0.752
0.3	3.31%	3.56%	0.718	0.778
0.4	3.29%	3.34%	0.753	0.819
0.5	3.43%	3.40%	0.831	0.868
0.6	3.33%	3.31%	0.928	0.909
0.7	3.40%	3.24%	0.907	0.903
0.8	3.36%	3.15%	0.832	0.891
0.9	3.35%	3.08%	0.776	0.861
Dtree	1.65%	1.65%	0.979	0.979

TABLE B.23: Success of target separation compared to Gini improvement by supervised weight for the wine quality data set

NrSegs	Best Gini improvement unsupervised	Best Gini improvement SSSKMIV	Best Gini improvement SSSKMCSQ	Best Gini improvement decision tree
2	0.59%	3.67%	4.11%	2.30%
3	2.96%	4.79%	4.43%	2.70%
4	5.18%	5.27%	4.92%	3.07%
5	5.72%	4.93%	5.37%	3.50%
6	5.69%	5.94%	6.07%	4.49%
7	5.66%	6.31%	5.95%	4.52%
8	5.69%	6.42%	6.15%	5.87%
9	6.32%	6.18%	6.22%	5.76%
10	6.67%	6.37%	6.37%	5.96%

TABLE B.24: Comparison of best improvements in Gini by number of segments for the wine quality data set

NrSegs	Average Gini improvement unsupervised	Average Gini improvement SSSKMIV	Average Gini improvement SSSKMCSQ	Average Gini improvement decision tree
2	0.45%	1.19%	1.10%	1.21%
3	2.25%	2.48%	2.50%	0.83%
4	2.30%	2.85%	2.74%	0.83%
5	3.26%	3.22%	3.39%	1.06%
6	3.51%	3.55%	3.52%	1.39%
7	3.79%	3.92%	3.96%	1.68%
8	4.16%	4.04%	4.17%	2.27%
9	3.98%	4.08%	4.10%	2.71%
10	4.45%	4.31%	4.25%	2.90%

TABLE B.25: Comparison of average improvements in Gini by number of segments for the wine quality data set

Segmentation method / supervised wgt	Percentage k means decisions over-ridden SSS-KMIV	Percentage k means decisions over-ridden SSSKM-CSQ	Average IV SSS-KMIV	Average CH value SSS-KMIV	Average IV SSS-KMCSQ	Average CH value SSSKM-CSQ
Unsupervised	NA	NA	0.340	0.209	0.343	0.209
0.1	2.85%	3.01%	0.483	0.208	0.483	0.209
0.2	5.98%	7.06%	0.667	0.207	0.703	0.207
0.3	7.83%	11.87%	0.861	0.205	1.036	0.201
0.4	8.98%	17.3%	0.930	0.204	1.442	0.190
0.5	13.17%	23.41%	1.139	0.200	1.824	0.180
0.6	19.72%	27.57%	1.468	0.192	1.947	0.177
0.7	26.06%	29.03%	1.671	0.186	1.895	0.175
0.8	30.84%	30.3%	1.601	0.184	1.858	0.174
0.9	35.44%	45.45%	1.532	0.183	1.576	0.144
Dtree	NA	NA	1.944	0.111	1.944	0.111

TABLE B.26: Comparison of segmentation characteristics by supervised weight for the chess (king-rook vs. king) data set

Segmentation method / supervised wgt	Average Gini improvement SSSKMIV	Average Gini improvement SSSKMCSQ	Average IV SSSKMIV	Average IV SSSKMCSQ
Unsupervised	9.67%	9.66%	0.340	0.343
0.1	9.62%	9.59%	0.483	0.483
0.2	9.65%	9.80%	0.667	0.703
0.3	9.60%	9.60%	0.861	1.036
0.4	9.75%	9.54%	0.930	1.442
0.5	9.42%	9.33%	1.139	1.824
0.6	9.22%	9.19%	1.468	1.947
0.7	9.09%	9.24%	1.671	1.895
0.8	9.12%	9.08%	1.601	1.858
0.9	9.06%	7.89%	1.532	1.576
Dtree	11.52%	11.52%	1.944	1.944

TABLE B.27: Success of target separation compared to Gini improvement by supervised weight for the chess (king-rook vs. king) data set

NrSegs	Best Gini improvement unsupervised	Best Gini improvement SSSKMIV	Best Gini improvement SSSKMCSQ	Best Gini improvement decision tree
2	5.22%	5.22%	5.22%	5.20%
3	9.14%	9.02%	8.74%	8.66%
4	9.62%	10.50%	9.69%	13.34%
5	11.04%	11.42%	11.14%	13.68%
6	12.28%	12.56%	11.98%	14.03%
7	13.19%	13.22%	13.05%	15.83%
8	13.98%	14.22%	14.15%	15.36%
9	14.92%	15.04%	14.94%	16.96%
10	15.61%	15.26%	15.15%	17.00%

TABLE B.28: Comparison of best improvements in Gini by number of segments for the chess (king-rook vs. king) data set

NrSegs	Average Gini improvement unsupervised	Average Gini improvement SSSKMIV	Average Gini improvement SSSKMCSQ	Average Gini improvement decision tree
2	2.52%	3.36%	3.21%	4.46%
3	5.50%	5.52%	5.59%	7.08%
4	7.60%	7.27%	7.74%	9.98%
5	9.40%	8.83%	8.95%	11.76%
6	10.66%	10.07%	9.85%	11.88%
7	11.70%	11.23%	10.71%	13.62%
8	12.61%	12.10%	11.76%	14.55%
9	13.36%	12.80%	12.48%	14.84%
10	13.61%	13.36%	12.97%	15.50%

TABLE B.29: Comparison of average improvements in Gini by number of segments for the chess (king-rook vs. king) data set

Segmentation method / supervised wgt	Percentage k means decisions overridden SSS-KMIV	Percentage k means decisions overridden SSSKM-CSQ	Average IV SSS-KMIV	Average CH value SSS-KMIV	Average IV SSS-KMCSQ	Average CH value SSSKM-CSQ
Unsupervised	NA	NA	0.032	0.382	0.033	0.380
0.1	5.03%	5.42%	0.037	0.384	0.037	0.386
0.2	12.64%	17.16%	0.039	0.369	0.035	0.363
0.3	17.01%	25.72%	0.038	0.361	0.039	0.353
0.4	22.64%	31.7%	0.040	0.355	0.038	0.337
0.5	31.87%	37.68%	0.039	0.342	0.034	0.332
0.6	40.3%	42.75%	0.039	0.313	0.035	0.307
0.7	43.83%	44.97%	0.038	0.309	0.034	0.300
0.8	45.74%	51.1%	0.034	0.305	0.031	0.282
0.9	48.75%	62.99%	0.033	0.308	0.031	0.232
Dtree	NA	NA	0.060	0.092	0.060	0.092

TABLE B.30: Comparison of segmentation characteristics by supervised weight for the claim prediction data set

Segmentation method / supervised wgt	Average Gini improvement SSSKMIV	Average Gini improvement SSSKMCSQ	Average IV SSSKMIV	Average IV SSSKMCSQ
Unsupervised	2.25%	2.27%	0.032	0.033
0.1	2.39%	2.27%	0.037	0.037
0.2	2.73%	2.62%	0.039	0.035
0.3	2.84%	2.93%	0.038	0.039
0.4	3.09%	2.97%	0.040	0.038
0.5	3.11%	2.77%	0.039	0.034
0.6	2.99%	2.69%	0.039	0.035
0.7	2.82%	2.56%	0.038	0.034
0.8	2.77%	2.53%	0.034	0.031
0.9	2.64%	2.63%	0.033	0.031
Dtree	1.06%	1.06%	0.060	0.060

TABLE B.31: Success of target separation compared to Gini improvement by supervised weight for the claim prediction data set

NrSegs	Best Gini improvement unsupervised	Best Gini improvement SSSKMIV	Best Gini improvement SSSKMCSQ	Best Gini improvement decision tree
2	1.01%	3.70%	3.54%	0.77%
3	3.20%	3.43%	4.39%	1.84%
4	3.39%	5.24%	5.27%	4.12%
5	4.60%	5.21%	5.60%	4.56%
6	4.89%	5.70%	5.70%	5.26%
7	4.21%	5.76%	6.25%	5.26%
8	4.37%	7.04%	6.54%	5.66%
9	4.84%	7.04%	6.86%	4.68%
10	4.77%	7.16%	6.31%	5.69%

TABLE B.32: Comparison of best improvements in Gini by number of segments for the claim prediction data set

NrSegs	Average Gini improvement unsupervised	Average Gini improvement SSSKMIV	Average Gini improvement SSSKMCSQ	Average Gini improvement decision tree
2	0.76%	1.92%	1.94%	-0.17%
3	2.28%	1.49%	1.52%	0.70%
4	2.53%	2.70%	2.61%	1.30%
5	2.63%	2.96%	2.70%	1.42%
6	2.62%	2.96%	2.91%	1.28%
7	2.27%	3.08%	2.86%	1.33%
8	2.38%	3.45%	3.21%	1.23%
9	2.42%	3.24%	3.12%	1.15%
10	2.44%	3.59%	3.09%	1.26%

TABLE B.33: Comparison of average improvements in Gini by number of segments for the claim prediction data set

Property	Description	Valid values	Value used
Variables	Variable Properties		
Architecture	Specifies the architecture used in building the network. If Single Layer is selected, the hidden nodes are added in parallel. If Block Layers is selected, the hidden nodes are added as additional layers with a uniform number of neurons. If Funnel Layers is selected, hidden nodes are added to each existing layer and to a new layer so that the layers form a funnel pattern. If Cascade is selected, hidden nodes are added in a cascade fashion.	Single Layer, Block Layers, Funnel Layers, Cascade	Single Layer
Termination	Specifies the method used to terminate training. Training will always stop when the maximum run time is exceeded. If "overfitting" is selected, training will stop when overfitting is detected. If validation data exist, this is determined by the value of the VMISC variable when the target is a class variable and by the VAVERR variable if the target is interval. If no validation data exist, overfitting is determined by the value of the MISC variable if the target is a class variable and by the value of the AIC variable if the target is interval. If Training Error is selected, training stops when the reduction in training data set error is less than 0.001. This is determined by looking at the value of the AVERR variable. If Time Limit is selected, models will be trained until training time is exhausted. When training is terminated, the model that has the best selection criteria is retained.	Overfitting, Training Error, Time Limit	Overfitting

Table B.34 Continued from previous page

Property	Description	Valid values	Value used
Train Action	Specifies the method used when training. If the method is TRAIN, all selected functions are trained until stopping. If the method is INCREMENT, the nodes are added one at a time. No activation function is reused. If a layer lowers the average error the layer is retained; otherwise the layer is dropped from the model. If the method is SEARCH, the nodes are added according to the architecture. The best network is retained as the final model.	Train, Increment, Search	Search
Target Layer Error Function	Specifies the Error Function to be used in the Target Layer. For a categorical variable, the Default error function is Multiple Bernoulli. For interval targets, the Default error function is Normal.	Default, Normal, Cauchy, Logistic, Huber, Biweight, Wave, Gamma, Poisson, Bernoulli, Entropy, MBernoulli, Multinomial, Mentropy	Default
Maximum Iterations	Maximum number of iterations to perform during training. This number can be any integer between 1 and 50.	$1.0 \leq n \leq 50.0$	8
Number of Hidden Units	Number of hidden units. This number can be any integer between 1 and 8.	$1.0 \leq n \leq 8.0$	2

Table B.34 Continued from previous page

Property	Description	Valid values	Value used
Tolerance	Indicates the extent of the preliminary search. If the value is Low, no preliminary search is performed. If the value is Medium, preliminary statements are executed. If the value is High, preliminary statements are executed and absconv=0.001 is applied to training.	Low, Medium, High	Medium
Total Time	Total amount of time allowed for training.	Five Minutes, Ten Minutes, Thirty Minutes, One Hour, Two Hours, Four Hours, Seven Hours, One Day, Two Days, Four Days, Seven Days	One Hour
Adjust Iterations	When set to No, the Adjust Iterations property suppresses the adjustments that are made to the Maximum Iterations property value setting when the Train Action is set to Search or Increment. If the Train Action is set to Search or Increment and Adjust Iterations is set to Yes, then the Maximum Iterations value is adjusted higher if the selected iteration equals the previously used Maximum Iterations. Similarly, the Maximum Iterations value can be adjusted lower if the selected iteration is significantly lower than the previously used Maximum Iterations.	Yes, No	Yes
Freeze Connections	Indicates whether connections should be frozen.	Yes, No	No

Table B.34 Continued from previous page

Property	Description	Valid values	Value used
Total Number of Hidden Units	If Train Action is Search, this number indicates the maximum total number of hidden units that can be used in training. When Search is performed, the total number of hidden units trained is calculated at each run. If the calculated number of total hidden units is greater than or equal to the value stored in Total Hidden, then training stops and a final model is selected. The default setting is 30.	5, 10, 20, 30, 40, 50, 75, 100	30
Final Training	Indicates whether the final model should be trained again to allow the model to converge. If this is set to Yes, the number of iterations used will correspond to the value set in Final Iterations.	Yes, No	Yes
Final Iterations	Number of iterations to use if Final Training is set to Yes.	5, 10, 20, 40	5
Direct	Indicates whether to use the Direct activation function during training.	Yes, No	Yes
Exponential	Indicates whether to use the Exponential activation function during training.	Yes, No	No
Identity	Indicates whether to use the Identity activation function during training.	Yes, No	No
Logistic	Indicates whether to use the Logistic activation function during training.	Yes, No	No
Normal	Indicates whether to use the Normal activation function during training.	Yes, No	Yes
Reciprocal	Indicates whether to use the Reciprocal activation function during training.	Yes, No	No
Sine	Indicates whether to use the Sine activation function during training.	Yes, No	Yes
Softmax	Indicates whether to use the Softmax activation function during training.	Yes, No	No

Table B.34 Continued from previous page

Property	Description	Valid values	Value used
Square	Indicates whether to use the Square activation function during training.	Yes, No	No
Tanh	Indicates whether to use the Tanh activation function during training.	Yes, No	Yes
Hidden Units	Specifies whether hidden unit variables should be created.	Yes, No	No
Residuals	Specifies whether residual variables should be created.	Yes, No	Yes
Standardization	Specifies whether standardization variables should be created.	Yes, No	No

TABLE B.34: Property values for the auto neural model

Property	Description	Valid values	Value used
Variables	Variable Properties		
Estimation Method	Specifies a SVM modeling method: Full Dense Quadratic Programming (FQP), Decomposed Quadratic Programming (DQP), Lagrangian SVM (LSVM), Least Squares SVM (LSSVM).	FQP, DQP, LSVM, LSSVM	LSSVM
Tuning Method	Specifies a tuning method. None means using the constant values currently specified.	None, Grid Search, Optimal Search	Optimal Search
Maximum QP Size	This option is valid for METHOD=DQP only and specifies the maximum size of the small decomposed QP	$2.0 \leq n$	100
Maximum Iteration	Specifies an upper bound for the number of iterations in each optimization.	$2.0 \leq n$	200
Maximum Function Call	Specifies an upper bound for the number of function calls in each optimization. The default is 500.	$2.0 \leq n$	500
Gradient Convergence Criterion	Specifies a relative gradient convergence criterion for the optimization process. The default is $1e-8$.	$0.0 < n$	$1e^{-8}$
Absolute Gradient Convergence Criterion	Specifies an absolute gradient convergence criterion for the optimization process. The default is $5e-4$.	$0.0 < n$	0.0005
Relative Convergence Criterion	Specifies a relative parameter convergence criterion for the optimization process. The default is $1e-8$.	$0.0 < n$	$1e^{-8}$

Table B.35 Continued from previous page

Property	Description	Valid values	Value used
Absolute Convergence Criterion	Specifies an absolute gradient convergence criterion for the optimization process. The default is 5e-4.	$0.0 < n$	0.0005
Use Conjugate Gradient Method	Specifies whether the iterative conjugate gradient method is used. When there are more than 3000 observations in training data, the Default option uses the iterative conjugate gradient method automatically.	Default, Yes, No	Default
Upper Bound of Conjugate Gradient Iteration	This option is valid only for the methods LS SVM with any kernel, as well as LSVM method with nonlinear kernel. Instead of solving the large linear system via Cholesky decomposition, the system is solved by the iterative conjugate gradient method. This option specifies an upper bound for the number of iterations.	$2.0 \leq n$	400
Conjugate Gradient Termination Tolerance	this option is valid only when the iterative conjugate gradient method is being solved. The option specifies a termination tolerance for the iteration. By default, 1e- 6 is used.	$0.0 < n$	0.000001
Scale Predictors	Specifies that the predictor variables are scaled before entering the analysis. This option is not valid with a polynomial kernel. Polynomial kernels always use scaled predictors.	Yes, No	Yes

Table B.35 Continued from previous page

Property	Description	Valid values	Value used
Regularization Parameter	Specifies how to find the value of regularization parameter, C. Constant means using a fixed value specified by the user. Tuning will use the specified tuning method.	Constant, Tuning	Tuning
Constant value	Specifies the value for regularization parameter C.	$1.0e-5 \leq n$	0.1
Start Value	Specifies the start value of tuning range.	$0.0 \leq n$	0.1
End Value	Specifies the end value of tuning range.	$0.0 \leq n$	1
Increment Value	Specifies the increment value in each step for parameter tuning.	$0.0 \leq n$	0.1
Kernel	Specifies a kernel function, the default is linear.	Linear, Polynomial, Radial Basis Function, Sigmoid	Linear
Polynomial Order	Specifies the degree of a polynomial in the kernel function. Constant means using a fixed value specified by the user.	Constant, Tuning	Constant
Constant Value	The kernel parameter value specified by user.	1, 2, 3, 4, 5, 6, 7, 8, 9, 10	2
Tuning Start Value	Specifies the start value of tuning range for the kernel parameter.	1, 2, 3, 4, 5, 6, 7, 8, 9, 10	2
Tuning End Value	Specifies the end value of tuning range for the kernel parameter.	1, 2, 3, 4, 5, 6, 7, 8, 9, 10	3
Tuning Increment value	Specifies the increment value in each step for the kernel parameter tuning.	1, 2	1

Table B.35 Continued from previous page

Property	Description	Valid values	Value used
Scale Parameter	Specifies the scale parameter value in the kernel function. Constant means using a fixed value specified by the user	Constant, Tuning	Constant
Constant Value	The scale parameter value specified by user.	$0.01 \leq n$	1
Tuning Start Value	Specifies the start value of tuning range for the scale parameter.	$0.0 \leq n$	0.2
Tuning End Value	Specifies the end value of tuning range for the scale parameter.	$0.0 \leq n$	1
Tuning Increment value	Specifies the increment value in each step for the scale parameter tuning.	$0.0 \leq n$	0.2
Kernel Scale Parameter	Specifies the scale parameter value in the kernel function. Constant means using a fixed value specified by the user	Constant, Tuning	Constant
Constant Value	The kernel scale parameter value specified by user.	$1.0e-12 \leq n$	1
Tuning Start Value	Specifies the start value of tuning range for the kernel scale parameter.	$0.0 \leq n$	0.1
Tuning End Value	Specifies the end value of tuning range for the kernel scale parameter.	$0.0 \leq n$	1
Tuning Increment value	Specifies the increment value in each step for the kernel scale parameter tuning.	$0.0 \leq n$	0.1

Table B.35 Continued from previous page

Property	Description	Valid values	Value used
Kernel Location Parameter	Specifies the location parameter value in the kernel function. Constant means using a fixed value specified by the user.	Constant, Tuning	Constant
Constant Value	The kernel location parameter value specified by the user.	$0.0 \leq n$	1
Tuning Start Value	Specifies the start value of tuning range for the kernel location parameter.	$0.0 \leq n$	0.1
Tuning End Value	Specifies the end value of tuning range for the kernel location parameter.	$0.0 \leq n$	1
Tuning Increment value	Specifies the increment value in each step for the kernel location parameter tuning	$0.0 \leq n$	0.1
Kernel Scale Parameter	Specifies the scale parameter value in the kernel function. Constant means using a fixed value specified by the user	Constant, Tuning	Constant
Constant Value	The kernel scale parameter value specified by user.	$1.0\text{e-}12 \leq n$	1
Tuning Start Value	Specifies the start value of tuning range for the kernel scale parameter.	$0.0 \leq n$	0.1
Tuning End Value	Specifies the end value of tuning range for the kernel scale parameter.	$0.0 \leq n$	1
Tuning Increment value	Specifies the increment value in each step for the kernel scale parameter tuning.	$0.0 \leq n$	0.1

Table B.35 Continued from previous page

Property	Description	Valid values	Value used
Kernel Location Parameter	Specifies the location parameter value in the kernel function. Constant means using a fixed value specified by the user.	Constant, Tuning	Constant
Constant Value	The kernel location parameter value specified by the user.	$0.0 \leq n$	1
Tuning Start Value	Specifies the start value of tuning range for the kernel location parameter.	$0.0 \leq n$	0.1
Tuning End Value	Specifies the end value of tuning range for the kernel location parameter.	$0.0 \leq n$	1
Tuning Increment value	Specifies the increment value in each step for the kernel location parameter tuning	$0.0 \leq n$	0.1
Cross Validation	Specifies whether cross validation is used. Cross Validation is only for classification.	Yes, No	No
Method	Cross validation method	Random, Testset	Random
Fold	Specifies the number of cross validation fold.	Default, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 15, 20	Default
Apply Sampling	Specifies whether sampling is used for SVM training, Stratified sampling is used for a class target and simple random sampling is used for an interval target.	Yes, No	Yes
Sample Size	Specifies the sampling size for SVM training	$10.0 \leq n$	1000
Print Option	Specifies a output print option	Default, All, No Print	Default

Table B.35 Continued from previous page

Property	Description	Valid values	Value used
Optimization History	Prints the detailed optimization history.	Yes, No	No

TABLE B.35: Property values for the support vector machines model

Property	Description	Valid values	Value used
Variables	Variable Properties		
Method	Determines what data representation is used to store the training set observations and then retrieve the nearest neighbors.	RD-Tree, Scan	RD-Tree
Number of Neighbors	The number of nearest neighbors to retrieve.	$1.0 \leq n$	16
Epsilon	If a nonzero number, it indicates an approximate nearest neighbor search, where the nearest neighbors determined so far must be at most "epsilon" away from the actual nearest neighbors to terminate the search. For large dimensionality, judicious use of epsilon can result in radically improved performance. Not relevant for the scan method.	$0.0 \leq n$	0
Number of Buckets	Specifies the number of buckets to allow a leaf node to grow to before splitting into a branch with two new leaves. This option only applies to the rdtree method.	$8.0 \leq n$	8
Weighted	Specifies to have the dimensions weighted according to the absolute value of their correlation with the target variable. This option is ignored (and a note printed) if the target is a class variable with more than two levels.	Yes, No	Yes
Create Nodes	Include a variable nnodes in the scored data sets that shows the number of point comparisons that had to be done to determine the answer. This is useful as a point of comparison.	Yes, No	No
Last Error	Error message from last run.		

TABLE B.36: Property values for the memory based reasoning model

Property	Description	Valid values	Value used
Variables	Variable Properties		
Inter-active	Launch interactive training window.		
Use Frozen Tree	Specifies whether a frozen tree definition should be used or if a new tree should be created during training.	Yes, No	No
Use Multiple Targets	Specifies whether multiple targets should be available during training of the decision tree node.	Yes, No	No
Precision	Specifies the number of decimals displayed in the Variable Importance, Subtree Assessment and Observation Based Importance tables and plots.	$1.0 \leq n \leq 12.0$	4
Interval Criterion	Specifies the method of searching for and evaluating candidate splitting rules in the presence of an interval target.	Variance, ProbF	ProbF
Nominal Criterion	Specifies the method of searching for and evaluating candidate splitting rules in the presence of a nominal target.	ProbChisq, Entropy, Gini	Gini
Ordinal Criterion	Specifies the method of searching for and evaluating candidate splitting rules in the presence of an ordinal target.	Entropy, Gini	Entropy
Significance Level	Specifies the maximum acceptable p-value for the worth of a candidate splitting rule when a PROBCHISQ or PROBF criterion method is selected.	$0.0 < n \leq 1.0$	0.2
Missing Values	Specifies how a splitting rule handles an observation with missing values. Missing values are either used as a value during the split search or assigned to a node based on this selection. For example, if Largest Branch is selected, all of those observations with missing values for the splitting rule would be placed in the branch with the largest number of training observations.	Use in search, Largest Branch, Most correlated branch	Use in search

Table B.37 Continued from previous page

Property	Description	Valid values	Value used
Use Input Once	Specifies that no splitting rule will be based on an input variable used in a splitting rule of an ancestor node.	Yes, No	No
Maximum Branch	Restricts the number of subsets that a splitting rule can produce to the specified number or fewer. For example, a value of 2 results in binary trees.	$2.0 \leq n \leq 100.0$	6
Maximum Depth	Specifies the maximum number of generations of nodes. The original node, generation 0, is called the root node. The children of the root node are the first generation.	$1.0 \leq n \leq 50.0$	6
Minimum Categorical Size	Minimum number of observations for a categorical value. A category must occur in at least the specified number of observations for it to be used in the split search.	$1.0 \leq n$	5
Split Precision	Specifies the number of decimals displayed in the splitting values and in average values displayed in the nodes of the tree.	$1.0 \leq n \leq 12.0$	4
Leaf Size	Specifies the smallest number of training observations that a leaf can have.	$1.0 \leq n$	5

Table B.37 Continued from previous page

Property	Description	Valid values	Value used
Number of Rules	Specifies how many splitting rules are saved with each node. The tree uses only one rule. The remaining rules are saved for comparison. For each node, a number of splitting rules are evaluated as the tree is constructed. A statistic that measures the strength of the rule is computed. If the Splitting Criterion is set to ProbChisq or ProbF, then the computed statistic is the LOGWORTH. LOGWORTH = $-\log(\text{p-value from Chi-square or F test})$. If the Entropy or Gini reduction criterion is selected, then the computed statistic is WORTH. WORTH measures the reduction in variance for the split. For both LOGWORTH and WORTH, larger values are better.	$1.0 \leq n$	5
Number of Surrogate Rules	Specifies the maximum number of surrogate rules that are sought in each non-leaf node. A surrogate rule is a backup to the main splitting rule. When the main splitting rule relies on an input whose value is missing, the first surrogate rule is invoked. If the first surrogate also relies on an input whose value is missing, the next surrogate is invoked. If missing values prevent the main rule and all of the surrogates from applying to an observation, then the main rule assigns the observation to the branch it has designated as receiving missing values.	$0.0 \leq n$	0
Split Size	Specifies the smallest number of training observations that a node must have for the procedure to consider splitting it.	$2.0 \leq n$	
Use Decisions	Specifies whether to use Decision information during the split search.	Yes, No	No

Table B.37 Continued from previous page

Property	Description	Valid values	Value used
Use Priors	Specifies whether to use Priors during the split search.	Yes, No	No
Exhaustive	Specifies the highest number of candidate splits to find in an exhaustive search. If more candidates should be considered, a heuristic search is used instead. The Exhaustive option applies to multi-way splits, and for binary splits on nominal targets with more than 2 values. The default value is 5000.	$0.0 \leq n \leq 2.0e9$	5000
Node Sample	Specifies the within-node sample size used for finding splits. If the number of training observations in a node is larger than n, then the split search for that node is based on a random sample of size n.	$2.0 \leq n$	20000
Method	Specifies how to construct the subtree in terms of selection methods. The following methods are possible: ASSESSMENT (the smallest subtree with the best assessment value is chosen), LARGEST (selects the full tree) and N (selects the largest subtree with at most n leaves).	Assessment, Largest, N	Assessment
Number of Leaves	Specifies the number of leaves that are used in creating the subtree when the SubTree option is set to N.	$1.0 \leq n$	1

Table B.37 Continued from previous page

Property	Description	Valid values	Value used
Assessment Measure	Use the Assessment Measure property to specify the method that you want to use to select the best tree, based on the validation data. If no validation data are available, training data are used. The available assessment measurements are Decision, Average Square Error, Misclassification, and Lift. The default setting of Decision selects the tree that has the largest average profit and smallest average loss if a profit or loss matrix is defined. If no profit or loss matrix is defined, the value of model assessment measure will be reset in the training process, depending on the measurement level of the target. If the target is interval, the measure is set to Average Square Error. If the target is categorical, the measure is set to Misclassification. The Average Square Error method selects the tree that has the smallest average square error. The Misclassification method selects the tree that has the smallest misclassification rate. The Lift method evaluates the tree based on the prediction of the top n% of the ranked observations. Observations are ranked based on their posterior probabilities or predicted target values. For an interval target, it is the average predicted target value of the top n% observations. For a categorical target, it is the proportion of events in the top n% data. When you set the Measure property to Lift, you must use the fraction property to specify the percentage for the top n% of cases.	Decision, Average Square Error, Misclassification, Lift	Decision

Table B.37 Continued from previous page

Property	Description	Valid values	Value used
Assess- ment Fraction	Specifies the fraction of observations to use when the Measure property is set to LIFT.	$0.0 < n < 1.0$	0.25
Perform Cross Valida- tion	Specifies whether to perform cross validation for each subtree in the sequence.	Yes, No	No
Number of Sub- sets	Specifies the number of cross validation subsets.	$2.0 \leq n \leq 200.0$	10
Number of Re- peats	Specifies the number of times to repeat cross validation. The estimates from repeated cross validation are the averages of the estimates from the individual cross validation runs.	$1.0 \leq n \leq 250.0$	1
Seed	Specifies the random number seed for generating the validation subsets.	$1.0 \leq n$	12345
Obser- vation Based Importance	Specifies whether to calculate observations based importance statistics.	Yes, No	No
Number Single Var Im- portance	Specifies the number of variables that one way importance statistics should be generated.	$1.0 \leq n$	5
Bonfer- roni Adjust- ment	Requests that a Bonferroni adjustment be applied to the p-values.	Yes, No	Yes
Time of Kass Adjust- ment	Indicates whether the Bonferroni adjustment should take place before or after the split is chosen.	After, Before	Before

Table B.37 Continued from previous page

Property	Description	Valid values	Value used
Inputs	Requests that an adjustment for the number of inputs be applied to the p-values.	Yes, No	No
Number of Inputs	Specifies the number of inputs to consider to be uncorrelated when Inputs=Yes.	$1.0 \leq n$	1
Split Adjustment	Requests that an adjustment for the number of ancestor splits be applied to the p-values.	Yes, No	Yes
Leaf Variable	Indicates whether a NODE variable is created. This variable contains a numeric ID of the leaf to which the observation is assigned.	Yes, No	Yes
Performance	Specifies where to put the working copy of the training data. The RAM option requests the working copy be stored in memory if enough memory is available for it and a minimum number of calculations. The DISK option requests the working copy be stored in a disk utility file. Storing the copy on disk may free a considerable amount of memory for calculations, possibly shortening the execution time.	Disk, Ram	Disk
Variable Selection	Specifies whether variable selection should be performed based on importance values. If this is set to Yes, all variables that have an importance value greater than or equal to 0.05 will be set to Input. All other variables will be set to Rejected.	Yes, No	Yes
Leaf Role	If Leaf Variable is set to Yes, this indicates the role of the NODE variable. This is set to SEGMENT by default.	Segment, Input, Rejected	Segment

TABLE B.37: Property values for the decision tree model

Property	Description	Valid values	Value used
Variables	Variable Properties		
N Iterations	Specifies the number of terms in the boosting series. For interval and binary targets, the number of iterations equals the trees. For a nominal target, a separate tree is created for each target category in each iteration.	$1.0 \leq n \leq 1000.0$	50
Seed	Specifies the seed for generating random numbers. The Training Proportion property uses this value to select a training sample at each iteration.	$0.0 \leq n$	12345
Shrinkage	Specifies how much to reduce the prediction of each tree.	$0.0010 \leq n \leq 1.0$	0.05
Train Proportion	Specifies the proportion of training observations to train a tree with. A different training sample is taken in each iteration. Trees trained in the same iteration have the same training data.	$0.0 < n \leq 100.0$	60
Huber M-Regression	Requests using the Huber M-Regression loss function instead of square error loss with an interval target. The Huber loss function is less sensitive to extreme target values. Huber=No requests using square error loss instead of Huber loss.	No, 0.6, 0.7, 0.8, 0.9, 1	No
Maximum Branch	Restricts the number of subsets that a splitting rule can produce to the specified number or fewer. For example, a value of 2 results in binary trees.	$2.0 \leq n \leq 100.0$	2
Maximum Depth	Specifies the maximum number of generations of nodes. The original node, generation 0, is called the root node. The children of the root node are the first generation.	$1.0 \leq n \leq 50.0$	2

Table B.38 Continued from previous page

Property	Description	Valid values	Value used
Minimum Categorical Size	Minimum number of observations for a categorical value. A category must occur in at least the specified number of observations for it to be used in the split search.	$1.0 \leq n$	5
Re-use Variable	Determines how many splitting rules in a path may use the same variable.	$0.0 \leq n$	1
Categorical Bins	Specifies the number of preliminary bins to collect categorical input values when preparing to search for a split.	$2.0 \leq n$	30
Interval Bins	Specifies the number of preliminary bins to consolidate interval input values into.	$2.0 \leq n$	100
Missing Values	Specifies how a splitting rule handles an observation with missing values. Missing values are either used as a value during the split search or assigned to a node based on this selection. For example, if Largest Branch is selected, all of those observations with missing values for the splitting rule would be placed in the branch with the largest number of training observations.	Use in search, Largest branch , Most correlated branch	Use in search
Performance	Specifies where to put the working copy of the training data. The RAM option requests the working copy be stored in memory if enough memory is available for it and a minimum number of calculations. The DISK option requests the working copy be stored in a disk utility file. Storing the copy on disk may free a considerable amount of memory for calculations, possibly shortening the execution time.	Disk, RAM	Disk
Leaf Fraction	Specifies the smallest number of training observations a new branch may have, expressed as the proportion of the number of available training observations in the data.	$0.0 \leq n < 1.0$	0.1

Table B.38 Continued from previous page

Property	Description	Valid values	Value used
Number of Surrogate Rules	Specifies the maximum number of surrogate rules that are sought in each non-leaf node. A surrogate rule is a backup to the main splitting rule. When the main splitting rule relies on an input whose value is missing, the first surrogate rule is invoked. If the first surrogate also relies on an input whose value is missing, the next surrogate is invoked. If missing values prevent the main rule and all of the surrogates from applying to an observation, then the main rule assigns the observation to the branch it has designated as receiving missing values.	$0.0 \leq n$	0
Split Size	Specifies the smallest number of training observations that a node must have for the procedure to consider splitting it.	$2.0 \leq n$	bigstrut
Exhaustive	Specifies the highest number of candidate splits to find in an exhaustive search. If more candidates should be considered, a heuristic search is used instead. The Exhaustive option applies to multi-way splits, and for binary splits on nominal targets with more than 2 values.	$0.0 \leq n \leq 2.0e9$	5000
Node Sample	Specifies the within-node sample size used for finding splits. If the number of training observations in a node is larger than n, then the split search for that node is based on a random sample of size n.	$2.0 \leq n$	20000
Assessment Measure	Use the Assessment Measure property to specify the method that you want to use to select the best tree.	Decision , Average Square Error, Misclassification	Decision

Table B.38 Continued from previous page

Property	Description	Valid values	Value used
Subseries	Specifies how many iterations in the series to use in the final model.	Best Assessment Value, Complete Series, N Iterations	Best Assessment Value
Number of Iterations	Specifies the specific iteration used in final model. This value is used if Subseries=N Iterations.	$1.0 \leq n$	1
Create H Statistic	Specifies whether the H statistic should be generated. This is an indication of which inputs may be involved in important interactions. If calculated, this will appear as an additional column in the Variable Importance table.	Yes, No	No
Variable Selection	Specifies whether variable selection should be performed based on importance values. If this is set to Yes, all variables that have an importance value greater than or equal to 0.05 will be set to Input. All other variables will be set to Rejected.	Yes, No	Yes
Observation Based Importance	Specifies whether observation-based importance statistics should be generated.	Yes, No	No
Number Single Var Importance	Specifies the number of variables that one way importance statistics should be generated.	$1.0 \leq n$	5

TABLE B.38: Property values for the gradient boosting model

NrSegs	Unsupervised	SSSKMIV	SSSKMCSQ	Dtree
2	25.72%	25.72%	25.72%	30.34%
3	25.97%	26.34%	26.09%	30.85%
4	25.09%	26.36%	26.54%	32.14%
5	25.50%	26.02%	25.67%	31.95%
6	26.17%	26.31%	26.23%	32.27%
7	24.06%	25.83%	26.85%	32.22%
8	26.53%	25.69%	27.37%	32.65%
9	27.11%	27.89%	29.14%	32.49%
10	26.42%	27.47%	28.18%	33.70%

TABLE B.39: Best Ginis obtained for the direct marketing data set

NrSegs	Unsupervised	SSSKMIV	SSSKMCSQ	Dtree
2	59.75%	60.94%	60.88%	60.62%
3	61.16%	62.44%	62.34%	61.66%
4	61.58%	63.86%	64.22%	62.99%
5	63.80%	64.62%	64.53%	63.66%
6	64.47%	65.80%	65.18%	64.00%
7	65.48%	66.14%	66.15%	64.87%
8	65.87%	70.37%	66.43%	65.10%
9	66.66%	67.09%	67.03%	65.86%
10	66.88%	67.68%	67.99%	66.43%

TABLE B.40: Best Ginis obtained for the protein tertiary structure data set

NrSegs	Unsupervised	SSSKMIV	SSSKMCSQ	Dtree
2	60.47%	60.47%	56.78%	60.88%
3	61.09%	63.34%	62.32%	61.30%
4	61.15%	63.17%	62.64%	70.17%
5	61.10%	63.08%	62.00%	70.73%
6	62.56%	63.34%	63.07%	70.79%
7	63.10%	63.36%	64.90%	70.81%
8	63.04%	64.65%	64.15%	70.86%
9	63.11%	66.25%	65.39%	70.87%
10	62.78%	64.97%	64.64%	70.89%

TABLE B.41: Best Ginis obtained for the credit scoring data set

NrSegs	Unsupervised	SSSKMIV	SSSKMCSQ	Dtree
2	61.13%	64.22%	64.66%	62.84%
3	63.51%	65.33%	64.97%	63.24%
4	65.72%	65.82%	65.46%	63.62%
5	66.26%	65.47%	65.91%	64.05%
6	66.24%	66.48%	66.62%	65.03%
7	66.20%	66.86%	66.49%	65.06%
8	66.24%	66.97%	66.70%	66.41%
9	66.87%	66.73%	66.76%	66.31%
10	67.21%	66.92%	66.92%	66.50%

TABLE B.42: Best Ginis obtained for the wine quality data set

NrSegs	Unsupervised	SSSKMIV	SSSKMCSQ	Dtree
2	76.56%	76.56%	76.56%	76.54%
3	80.48%	80.36%	80.08%	80.00%
4	80.96%	81.84%	81.03%	84.68%
5	82.38%	82.76%	82.48%	85.02%
6	83.62%	83.90%	83.32%	85.37%
7	84.53%	84.56%	84.39%	87.17%
8	85.32%	85.56%	85.49%	86.70%
9	86.26%	86.38%	86.28%	88.30%
10	86.95%	86.60%	86.49%	88.34%

TABLE B.43: Best Ginis obtained for the chess (king-rook vs. king) data set

NrSegs	Unsupervised	SSSKMIV	SSSKMCSQ	Dtree
2	9.04%	11.73%	11.57%	8.80%
3	11.23%	11.46%	12.41%	9.87%
4	11.42%	13.27%	13.30%	12.14%
5	12.63%	13.24%	13.62%	12.59%
6	12.92%	13.73%	13.73%	13.29%
7	12.24%	13.79%	14.27%	13.29%
8	12.40%	15.07%	14.57%	13.68%
9	12.87%	15.07%	14.89%	12.70%
10	12.80%	15.19%	14.34%	13.72%

TABLE B.44: Best Ginis obtained for the claim prediction data set

Appendix C

Appendix for additional information in Chapter 6

This Appendix contains relevant additional information as referred to in Chapter 6.

IV group	Unsupervised average	SSSKMIV average	Decision tree average	True IV
0.0-0.05	0.009	0.01	0.027	0.035
0.05-0.5	0.037	0.041	0.137	0.287
0.5-0.8	0.069	0.087	0.281	0.631
0.8-Inf	0.113	0.168	0.426	1.117

TABLE C.1: IV values obtained by each segmentation method by IV Group compared to actual IV for $O = 5$

CH group	Unsupervised average	SSSKMIV average	Decision tree average	True value	CH
Low	0.106	0.101	0.075	0.052	
Mid	0.111	0.104	0.081	0.07	
High	0.116	0.11	0.085	0.088	

TABLE C.2: CH values obtained by each segmentation method by IV Group compared to actual CH for $O = 5$

IV group	Average unsuper-vised Gini	Average SSSKMIV Gini	Average decision tree Gini	Average Gini with no segmen-tation	Average true Gini
0.0-0.05	48.63%	49.55%	47.21%	48.91%	68.14%
0.05-0.5	42.30%	43.87%	45.31%	42.47%	66.83%
0.5-0.8	38.71%	41.12%	46.33%	37.76%	69.85%
0.8-Inf	35.32%	39.95%	46.43%	33.03%	73.32%

TABLE C.3: Gini values obtained by each of the segmentation methodologies compared with true Gini by IV group for $O = 5$

CH group	Average unsuper-vised Gini	Average SSSKMIV Gini	Average decision tree Gini	Average Gini with no segmen-tation	Average true Gini
Low	39.41%	41.54%	44.08%	40.70%	69.76%
Mid	40.91%	43.31%	46.37%	40.17%	69.33%
High	43.39%	46.01%	48.50%	40.75%	69.52%

TABLE C.4: Gini values obtained by each of the segmentation methodologies compared with true Gini by CH group for $O = 5$

IV group	Unsupervised Gini im-provement percentage	SSSKMIV Gini im-provement percentage	Decision tree Gini im-provement percentage
0.0-0.05	-1.56%	3.16%	-9.04%
0.05-0.5	-0.85%	5.59%	10.83%
0.5-0.8	2.86%	10.34%	26.25%
0.8-Inf	5.35%	16.59%	32.20%

TABLE C.5: Improvement in Ginis by IV group as a percentage of true Gini improvement for $O = 5$

CH group	Unsupervised Gini im-provement percentage	SSSKMIV Gini im-provement percentage	Decision tree Gini im-provement percentage
Low	-4.84%	1.94%	8.20%
Mid	1.16%	8.51%	15.59%
High	8.03%	16.31%	21.40%

TABLE C.6: Improvement in Ginis by CH group as a percentage of true Gini improvement for $O = 5$

IV group	CH group	Unsupervised Gini improvement percentage	SSSKMIV Gini improvement percentage	Decision tree Gini improvement percentage
0.0-0.05	Low	-6.79%	-1.79%	-11.08%
	Mid	-2.06%	1.75%	-8.85%
	High	4.16%	9.53%	-7.20%
0.05-0.5	Low	-6.61%	0.10%	5.31%
	Mid	-0.56%	5.28%	7.62%
	High	4.62%	11.38%	19.56%
0.5-0.8	Low	-1.10%	3.79%	19.09%
	Mid	-0.11%	8.40%	27.59%
	High	9.79%	18.84%	32.08%
0.8-Inf	Low	-4.87%	5.67%	19.49%
	Mid	7.38%	18.63%	35.98%
	High	13.54%	25.48%	41.15%

TABLE C.7: Improvement in Ginis by CH and IV group as a percentage of true Gini improvement for the unsupervised example and $O = 5$

IV group	Unsupervised average	SSSKMIV average	Decision tree average	True IV
0.0-0.05	0.023	0.023	0.033	0.034
0.05-0.5	0.143	0.162	0.177	0.282
0.5-0.8	0.296	0.37	0.352	0.629
0.8-Inf	0.549	0.685	0.636	1.136

TABLE C.8: IV values obtained by each segmentation method by IV Group compared to actual IV for $O = 15$

CH group	Unsupervised average	SSSKMIV average	Decision tree average	True CH value
Low	0.139	0.135	0.092	0.055
Mid	0.155	0.15	0.098	0.068
High	0.171	0.167	0.105	0.082

TABLE C.9: CH values obtained by each segmentation method by IV Group compared to actual CH for $O = 15$

IV group	Average unsupervised Gini	Average SSSKMIV Gini	Average decision tree Gini	Average Gini with no segmentation	Average true Gini
0.0-0.05	56.57%	56.96%	47.91%	48.81%	68.25%
0.05-0.5	52.96%	54.88%	47.83%	42.79%	66.89%
0.5-0.8	52.09%	55.76%	48.88%	37.71%	69.55%
0.8-Inf	54.06%	58.94%	52.47%	32.03%	73.33%

TABLE C.10: Gini values obtained by each of the segmentation methodologies compared with true Gini by IV group for $O = 15$

CH group	Average unsupervised Gini	Average SSSKMIV Gini	Average decision tree Gini	Average Gini with no segmentation	Average true Gini
Low	50.33%	53.13%	47.70%	40.14%	69.38%
Mid	54.42%	57.19%	49.62%	40.87%	69.84%
High	57.02%	59.59%	50.51%	39.99%	69.29%

TABLE C.11: Gini values obtained by each of the segmentation methodologies compared with true Gini by CH group for $O = 15$

IV group	Unsupervised Gini improvement percentage	SSSKMIV Gini improvement percentage	Decision tree Gini improvement percentage
0.0-0.05	39.67%	41.73%	-4.94%
0.05-0.5	42.03%	49.82%	19.85%
0.5-0.8	44.91%	56.60%	34.64%
0.8-Inf	52.84%	64.69%	48.87%

TABLE C.12: Improvement in Ginis by IV group as a percentage of true Gini improvement for $O = 15$

CH group	Unsupervised Gini improvement percentage	SSSKMIV Gini improvement percentage	Decision tree Gini improvement percentage
Low	32.88%	41.29%	20.41%
Mid	45.15%	53.83%	23.93%
High	56.56%	64.51%	29.46%

TABLE C.13: Improvement in Ginis by CH group as a percentage of true Gini improvement for $O = 15$

IV group	CH group	Unsupervised Gini improvement percentage	SSSKMIV Gini improvement percentage	Decision tree Gini improvement percentage
0.0-0.05	Low	28.22%	29.91%	-7.09%
	Mid	38.37%	41.94%	-5.05%
	High	52.43%	53.35%	-2.67%
0.05-0.5	Low	27.91%	36.29%	16.99%
	Mid	42.88%	50.61%	17.22%
	High	55.29%	62.55%	25.32%
0.5-0.8	Low	31.43%	43.86%	29.29%
	Mid	47.73%	57.78%	33.43%
	High	55.58%	68.16%	41.20%
0.8-Inf	Low	43.94%	55.10%	42.46%
	Mid	51.63%	64.99%	50.14%
	High	62.96%	73.99%	54.01%

TABLE C.14: Improvement in Ginis by CH and IV group as a percentage of true Gini improvement for the unsupervised example and $O = 15$

IV group	Unsupervised average	SSSKMIV average	Decision tree average	True IV
0.0-0.05	0.016	0.017	0.031	0.035
0.05-0.5	0.096	0.111	0.169	0.304
0.5-0.8	0.194	0.233	0.344	0.624
0.8-Inf	0.321	0.417	0.566	1.101

TABLE C.15: IV values obtained by each segmentation method by IV Group compared to actual IV for $O = 10$

CH group	Unsupervised average	SSSKMIV average	Decision tree average	True value	CH
Low	0.122	0.116	0.083	0.056	
Mid	0.132	0.124	0.09	0.071	
High	0.144	0.138	0.095	0.088	

TABLE C.16: CH values obtained by each segmentation method by IV Group compared to actual CH for $O = 10$

IV group	Average unsuper-vised Gini	Average SSSKMIV Gini	Average decision tree Gini	Average Gini with no segmen-tation	Average true Gini
0.0-0.05	52.19%	52.90%	47.57%	48.72%	68.33%
0.05-0.5	47.41%	49.41%	46.87%	42.48%	67.25%
0.5-0.8	45.59%	48.52%	48.19%	37.09%	69.04%
0.8-Inf	45.49%	50.46%	51.11%	33.18%	73.32%

TABLE C.17: Gini values obtained by each of the segmentation methodologies compared with true Gini by IV group for $O = 10$

CH group	Average unsuper-vised Gini	Average SSSKMIV Gini	Average decision tree Gini	Average Gini with no segmen-tation	Average true Gini
Low	44.31%	47.12%	46.89%	40.43%	69.62%
Mid	47.49%	50.26%	48.86%	40.45%	69.46%
High	51.21%	53.58%	49.56%	40.22%	69.38%

TABLE C.18: Gini values obtained by each of the segmentation methodologies compared with true Gini by CH group for $O = 10$

IV group	Unsupervised Gini im-provement percentage	SSSKMIV Gini im-provement percentage	Decision tree Gini im-provement percentage
0.0-0.05	17.63%	21.23%	-6.06%
0.05-0.5	19.58%	27.56%	16.97%
0.5-0.8	26.52%	35.60%	34.37%
0.8-Inf	29.93%	42.20%	43.81%

TABLE C.19: Improvement in Ginis by IV group as a percentage of true Gini improvement for $O = 10$

CH group	Unsupervised Gini improvement percentage	SSSKMIV Gini improvement percentage	Decision tree Gini improvement percentage
Low	11.71%	20.46%	17.24%
Mid	22.85%	31.31%	23.57%
High	35.70%	43.16%	26.00%

TABLE C.20: Improvement in Ginis by CH group as a percentage of true Gini improvement for $O = 10$

IV group	CH group	Unsupervised Gini improvement percentage	SSSKMIV Gini improvement percentage	Decision tree Gini improvement percentage
0.0-0.05	Low	5.52%	9.97%	-9.29%
	Mid	17.70%	21.26%	-4.24%
	High	29.66%	32.47%	-4.66%
0.05-0.5	Low	8.85%	16.80%	11.78%
	Mid	20.38%	27.99%	20.45%
	High	29.52%	37.88%	18.68%
0.5-0.8	Low	14.09%	23.54%	29.21%
	Mid	25.37%	34.42%	33.78%
	High	40.11%	48.83%	40.12%
0.8-Inf	Low	18.37%	31.54%	37.27%
	Mid	27.94%	41.58%	44.30%
	High	43.49%	53.46%	49.86%

TABLE C.21: Improvement in Ginis by CH and IV group as a percentage of true Gini improvement for the unsupervised example and $O = 10$

IV group	Unsupervised average	SSSKMIV average	Decision tree average	True IV
0.0-0.01	0.008	0.009	0.02	0.007
0.01-0.02	0.014	0.013	0.024	0.016
0.02-0.03	0.018	0.017	0.028	0.026
0.03-0.05	0.028	0.027	0.036	0.04

TABLE C.22: IV values obtained by each segmentation method by IV Group compared to actual IV for the unsupervised example and $O = 15$

CH group	Unsupervised average	SSSKMIV average	Decision tree average	True CH value
Low	0.141	0.132	0.087	0.055
Mid	0.155	0.145	0.091	0.068
High	0.17	0.16	0.093	0.082

TABLE C.23: CH values obtained by each segmentation method by IV Group compared to actual CH for the unsupervised example and $O = 15$

IV group	Average unsuper-vised Gini	Average SSSKMIV Gini	Average decision tree Gini	Average Gini with no segmen-tation	Average true Gini
0.0-0.01	59.50%	59.44%	49.99%	51.38%	70.50%
0.01-0.02	59.31%	59.23%	50.52%	51.12%	70.50%
0.02-0.03	58.82%	58.81%	49.58%	50.60%	70.34%
0.03-0.05	58.58%	58.41%	49.68%	50.00%	69.96%

TABLE C.24: Gini values obtained by each of the segmentation methodologies compared with true Gini by IV group for the unsupervised example and $O = 15$

CH group	Average unsuper-vised Gini	Average SSSKMIV Gini	Average decision tree Gini	Average Gini with no segmen-tation	Average true Gini
Low	56.86%	56.98%	49.48%	50.79%	70.47%
Mid	59.23%	58.99%	49.93%	50.69%	70.29%
High	61.08%	60.95%	50.42%	50.84%	70.22%

TABLE C.25: Gini values obtained by each of the segmentation methodologies compared with true Gini by CH group for the unsupervised example and $O = 15$

IV group	Unsupervised Gini im-provement percentage	SSSKMIV Gini im-provement percentage	Decision tree Gini im-provement percentage
0.0-0.01	42.45%	42.20%	-7.32%
0.01-0.02	42.30%	41.74%	-3.16%
0.02-0.03	41.44%	41.51%	-5.30%
0.03-0.05	43.04%	42.13%	-1.59%

TABLE C.26: Improvement in Ginis by IV group as a percentage of true Gini improvement for the unsupervised example and $O = 15$

CH group	Unsupervised Gini im-provement percentage	SSSKMIV Gini im-provement percentage	Decision tree Gini im-provement percentage
Low	30.76%	31.32%	-6.73%
Mid	43.38%	42.24%	-4.06%
High	52.78%	52.12%	-2.24%

TABLE C.27: Improvement in Ginis by CH group as a percentage of true Gini improvement for the unsupervised example and $O = 15$

IV group	CH group	Unsupervised Gini improvement percentage	SSSKMIV Gini improvement percentage	Decision tree Gini improvement percentage
0.0-0.01	Low	32.72%	31.78%	-8.30%
	Mid	41.20%	42.13%	-7.51%
	High	53.44%	52.71%	-6.16%
0.01-0.02	Low	28.74%	30.72%	-3.92%
	Mid	44.00%	42.41%	-4.94%
	High	54.16%	52.09%	-0.63%
0.02-0.03	Low	30.08%	30.89%	-8.79%
	Mid	43.89%	41.43%	-4.46%
	High	50.35%	52.21%	-2.66%
0.03-0.05	Low	31.52%	31.90%	-5.92%
	Mid	44.43%	43.01%	0.66%
	High	53.15%	51.48%	0.50%

TABLE C.28: Improvement in Ginis by CH and IV group as a percentage of true Gini improvement for the unsupervised example and $O = 15$

Appendix D

Appendix for additional information in Chapter 7

This Appendix contains relevant additional information as referred to in Chapter 7.

SSE Wgt (ν)	Average population % in smallest segment	Average population % in largest segment	Average normalised value of ϱ
0.0	8.7%	36.58%	35.03%
0.1	9.49%	34.86%	41.29%
0.2	11.13%	31.53%	58.08%
0.4	14.34%	29.8%	74.74%

TABLE D.1: Impact on population percentages by increasing ν for the direct marketing data set

Distortion weight (w)	Average SSS-KMIV-SSE IV $\nu = 0.1$	Average SSS-KMIV-SSE IV $\nu = 0.2$	Average SSS-KMIV-SSE IV $\nu = 0.4$	Average SSS-KMIV IV	Average SSS-KM-CSQ IV	Unsupervised IV	Decision tree IV
0.1	0.08	0.09	0.10	0.07	0.08	0.08	0.23
0.2	0.09	0.09	0.09	0.08	0.08	0.08	0.23
0.4	0.09	0.09	0.08	0.09	0.09	0.08	0.23

TABLE D.2: IV values by distortion weight for the direct marketing data set

Distortion weight (w)	Average SSS-KMIV-SSE CH value $\nu = 0.1$	Average SSS-KMIV-SSE CH value $\nu = 0.2$	Average SSS-KMIV-SSE CH value $\nu = 0.4$	Average SSS-KMIV CH value	Average SSKM-CSQ CH value	Unsupervised CH value	Decision tree CH value
0.1	0.16	0.16	0.16	0.16	0.16	0.16	0.09
0.2	0.16	0.16	0.15	0.16	0.16	0.16	0.09
0.4	0.15	0.15	0.15	0.15	0.15	0.16	0.09

TABLE D.3: CH values by distortion weight for the direct marketing data set

Number of segments	Best Gini improvement unsupervised	Best Gini improvement SSS-KMIV	Best Gini improvement SSS-KMCSQ	Best Gini improvement decision tree	Best Gini improvement SSS-KMIVSSE
2	3.74%	3.74%	3.74%	8.37%	-1.15%
3	4.0%	4.37%	4.12%	8.88%	1.55%
4	3.12%	4.39%	4.56%	10.16%	.06%
5	3.53%	4.04%	3.69%	9.97%	.25%
6	4.2%	4.34%	4.26%	10.3%	.5%
7	2.08%	3.85%	4.87%	10.24%	-1.5%
8	4.55%	3.72%	5.4%	10.68%	-1.16%
9	5.13%	5.91%	7.16%	10.52%	-.97%
10	4.45%	5.49%	6.21%	11.72%	.13%

TABLE D.4: Comparison of Gini improvement by number of segments for the direct marketing data set

Number of segments	Best Gini improvement $\nu = 0.1$	Best Gini improvement $\nu = 0.2$	Best Gini improvement $\nu = 0.4$
2	-1.15%	-1.24%	-1.25%
3	1.55%	-.69%	-.05%
4	-2.34%	-.88%	.06%
5	-2.53%	.25%	-1.07%
6	-1.35%	-2.1%	.5%
7	-2.49%	-2.35%	-1.5%
8	-2.48%	-1.16%	-1.31%
9	-2.95%	-.99%	-.97%
10	-5.11%	-2.61%	.13%

TABLE D.5: Comparison of Gini improvement by number of segments and value of ν for the direct marketing data set

Number of segments	Best Gini $\nu = 0.1$	Best Gini $\nu = 0.2$	Best Gini $\nu = 0.4$
2	25.8%	25.7%	25.69%
3	28.49%	26.25%	26.9%
4	24.6%	26.06%	27.%
5	24.41%	27.2%	25.88%
6	25.59%	24.84%	27.45%
7	24.46%	24.59%	25.45%
8	24.47%	25.79%	25.64%
9	23.99%	25.96%	25.97%
10	21.83%	24.33%	27.07%

TABLE D.6: Comparison of actual Ginis by number of segments and value of ν for the direct marketing data set

SSE Wgt (ν)	Average population % in smallest segment	Average population % in largest segment	Average normalised value of ϱ
0.0	13.05%	29.43%	69.7%
0.1	16.04%	26.62%	90.65%
0.2	17.2%	25.65%	94.8%
0.4	16.2%	27.1%	86.96%

TABLE D.7: Impact on population percentages by increasing ν for the protein tertiary structure data set

Distortion weight (w)	Average SSS-KMIV-SSE IV $\nu = 0.1$	Average SSS-KMIV-SSE IV value $\nu = 0.2$	Average SSS-KMIV-SSE IV $\nu = 0.4$	Average SSS-KMIV IV	Average SSS-KM-CSQ IV	Unsupervised IV	Decision tree IV
0.1	0.45	0.44	0.50	0.45	0.46	0.36	0.76
0.2	0.50	0.50	0.56	0.50	0.62	0.36	0.76
0.4	0.60	0.61	0.63	0.63	0.77	0.36	0.76

TABLE D.8: IV values by distortion weight for the protein tertiary structure data set

Distortion weight (w)	Average SSS-KMIV-SSE CH value $\nu = 0.1$	Average SSS-KMIV-SSE CH value $\nu = 0.2$	Average SSS-KMIV-SSE CH value $\nu = 0.4$	Average SSS-KMIV CH value	Average SSKM-CSQ CH value	Unsupervised CH value	Decision tree CH value
0.1	0.76	0.74	0.68	0.77	0.77	0.78	0.32
0.2	0.72	0.72	0.64	0.75	0.71	0.78	0.32
0.4	0.69	0.68	0.59	0.71	0.68	0.78	0.32

TABLE D.9: CH values by distortion weight for the protein tertiary structure data set

Number of segments	Best Gini improvement unsupervised	Best Gini improvement SSS-KMIV	Best Gini improvement SSS-KMCSQ	Best Gini improvement decision tree	Best Gini improvement SSS-KMIVSSE
2	.72%	1.9%	1.84%	1.58%	1.12%
3	2.12%	3.41%	3.3%	2.62%	3.74%
4	2.54%	4.82%	5.18%	3.96%	7.3%
5	4.76%	5.58%	5.49%	4.62%	8.26%
6	5.44%	6.77%	6.14%	4.96%	9.36%
7	6.45%	7.1%	7.11%	5.83%	8.32%
8	6.83%	11.33%	7.4%	6.06%	9.88%
9	7.63%	8.05%	8.%	6.82%	10.76%
10	7.84%	8.64%	8.95%	7.39%	11.1%

TABLE D.10: Comparison of Gini improvement by number of segments for the protein tertiary structure data set

Number of segments	Best Gini improvement $\nu = 0.1$	Best Gini improvement $\nu = 0.2$	Best Gini improvement $\nu = 0.4$
2	1.12%	1.07%	1.12%
3	3.48%	3.22%	3.74%
4	4.01%	4.93%	7.3%
5	5.7%	8.26%	6.93%
6	6.17%	9.36%	7.84%
7	6.77%	8.32%	8.18%
8	9.88%	9.72%	9.22%
9	7.75%	10.76%	10.07%
10	8.5%	11.1%	10.1%

TABLE D.11: Comparison of Gini improvement by number of segments and value of ν for the protein tertiary structure data set

Number of segments	Best Gini $\nu = 0.1$	Best Gini $\nu = 0.2$	Best Gini $\nu = 0.4$
2	60.16%	60.11%	60.16%
3	62.51%	62.26%	62.77%
4	63.05%	63.97%	66.34%
5	64.74%	67.3%	65.97%
6	65.2%	68.4%	66.87%
7	65.81%	67.35%	67.22%
8	68.92%	68.76%	68.26%
9	66.79%	69.8%	69.11%
10	67.54%	70.13%	69.14%

TABLE D.12: Comparison of actual Ginis by number of segments and value of ν for the protein tertiary structure data set

SSE Wgt (ν)	Average population % in smallest segment	Average population % in largest segment	Average normalised value of ϱ
0.0	6.35%	35.68%	21.83%
0.1	9.49%	34.87%	31.27%
0.2	12.33%	29.03%	48.42%
0.4	16.86%	24.37%	79.21%

TABLE D.13: Impact on population percentages by increasing ν for the credit application data set

Distortion weight (w)	Average SSS-KMIV-SSE IV $\nu = 0.1$	Average SSS-KMIV-SSE IV value $\nu = 0.2$	Average SSS-KMIV-SSE IV $\nu = 0.4$	Average SSS-KMIV IV	Average SSS-KM-CSQ IV	Unsupervised IV	Decision tree IV
0.1	0.24	0.22	0.22	0.21	0.22	0.19	1.14
0.2	0.25	0.22	0.22	0.21	0.24	0.19	1.14
0.4	0.24	0.23	0.21	0.25	0.27	0.19	1.14

TABLE D.14: IV values by distortion weight for the credit application data set

Distortion weight (w)	Average SSS-KMIV-SSE CH value $\nu = 0.1$	Average SSS-KMIV-SSE CH value $\nu = 0.2$	Average SSS-KMIV-SSE CH value $\nu = 0.4$	Average SSS-KMIV CH value	Average SSKM-CSQ CH value	Unsupervised CH value	Decision tree CH value
0.1	0.27	0.24	0.21	0.29	0.28	0.27	0.03
0.2	0.28	0.27	0.20	0.28	0.28	0.27	0.03
0.4	0.28	0.24	0.24	0.28	0.27	0.27	0.03

TABLE D.15: CH values by distortion weight for the credit application data set

Number of segments	Best Gini improvement unsupervised	Best Gini improvement SSS-KMIV	Best Gini improvement SSS-KMCSQ	Best Gini improvement decision tree	Best Gini improvement SSS-KMIVSSE
2	22.39%	22.39%	18.7%	22.8%	22.39%
3	23.01%	25.26%	24.24%	23.22%	23.15%
4	23.06%	25.09%	24.55%	32.09%	24.8%
5	23.01%	24.99%	23.92%	32.65%	24.75%
6	24.48%	25.25%	24.98%	32.71%	25.28%
7	25.02%	25.28%	26.81%	32.72%	25.55%
8	24.96%	26.57%	26.06%	32.78%	26.09%
9	25.03%	28.17%	27.31%	32.78%	25.75%
10	24.69%	26.89%	26.55%	32.8%	27.36%

TABLE D.16: Comparison of Gini improvement by number of segments for the credit application data set

Number of segments	Best Gini improvement $\nu = 0.1$	Best Gini improvement $\nu = 0.2$	Best Gini improvement $\nu = 0.4$
2	22.39%	14.76%	14.84%
3	23.15%	22.88%	18.97%
4	24.8%	23.75%	20.02%
5	24.73%	24.75%	23.42%
6	24.97%	25.28%	25.02%
7	25.28%	25.55%	24.33%
8	25.46%	25.2%	26.09%
9	25.75%	25.53%	25.7%
10	25.67%	27.36%	25.23%

TABLE D.17: Comparison of Gini improvement by number of segments and value of ν for the credit application data set

Number of segments	Best Gini $\nu = 0.1$	Best Gini $\nu = 0.2$	Best Gini $\nu = 0.4$
2	81.42%	73.79%	73.88%
3	82.19%	81.92%	78.01%
4	83.83%	82.79%	79.05%
5	83.77%	83.79%	82.45%
6	84.01%	84.32%	84.06%
7	84.32%	84.59%	83.37%
8	84.5%	84.24%	85.13%
9	84.79%	84.57%	84.04%
10	84.7%	86.4%	84.27%

TABLE D.18: Comparison of actual Ginis by number of segments and value of ν for the credit application data set

SSE Wgt (ν)	Average population % in smallest segment	Average population % in largest segment	Average normalised value of ϱ
0.0	11.53%	30.04%	58.02%
0.1	12.08%	30.1%	69.51%
0.2	13.05%	28.94%	75.22%
0.4	11.48%	29.22%	65.24%

TABLE D.19: Impact on population percentages by increasing ν for the wine quality data set

Distortion weight (w)	Average SSS-KMIV-SSE IV $\nu = 0.1$	Average SSS-KMIV-SSE IV value $\nu = 0.2$	Average SSS-KMIV-SSE IV $\nu = 0.4$	Average SSS-KMIV IV	Average SSS-KM-CSQ IV	Unsupervised IV	Decision tree IV
0.1	0.66	0.67	0.65	0.66	0.71	0.64	0.98
0.2	0.72	0.72	0.67	0.69	0.75	0.64	0.98
0.4	0.77	0.77	0.67	0.75	0.82	0.64	0.98

TABLE D.20: IV values by distortion weight for the wine quality data set

Distortion weight (w)	Average SSS-KMIV-SSE CH value $\nu = 0.1$	Average SSS-KMIV-SSE CH value $\nu = 0.2$	Average SSS-KMIV-SSE CH value $\nu = 0.4$	Average SSS-KMIV CH value	Average SSKM-CSQ CH value	Unsupervised CH value	Decision tree CH value
0.1	0.38	0.38	0.37	0.38	0.38	0.37	0.11
0.2	0.38	0.38	0.37	0.38	0.37	0.37	0.11
0.4	0.36	0.36	0.36	0.36	0.35	0.37	0.11

TABLE D.21: CH values by distortion weight for the wine quality data set

Number of segments	Best Gini improvement unsupervised	Best Gini improvement SSS-KMIV	Best Gini improvement SSS-KMCSQ	Best Gini improvement decision tree	Best Gini improvement SSS-KMIVSSE
2	.59%	3.67%	4.11%	2.3%	.82%
3	2.96%	4.79%	4.43%	2.7%	4.01%
4	5.18%	5.27%	4.92%	3.07%	5.1%
5	5.72%	4.93%	5.37%	3.5%	5.67%
6	5.69%	5.94%	6.07%	4.49%	6.13%
7	5.66%	6.31%	5.95%	4.52%	6.23%
8	5.69%	6.42%	6.15%	5.87%	6.18%
9	6.32%	6.18%	6.22%	5.76%	6.27%
10	6.67%	6.37%	6.37%	5.96%	6.32%

TABLE D.22: Comparison of Gini improvement by number of segments for the wine quality data set

Number of segments	Best Gini improvement $\nu = 0.1$	Best Gini improvement $\nu = 0.2$	Best Gini improvement $\nu = 0.4$
2	.71%	.71%	.82%
3	3.14%	3.66%	4.01%
4	4.16%	3.96%	5.1%
5	5.64%	5.07%	5.67%
6	6.13%	5.19%	6.%
7	6.23%	6.22%	5.9%
8	6.18%	6.15%	5.86%
9	6.18%	5.74%	6.27%
10	5.83%	6.32%	6.01%

TABLE D.23: Comparison of Gini improvement by number of segments and value of ν for the wine quality data set

Number of segments	Best Gini $\nu = 0.1$	Best Gini $\nu = 0.2$	Best Gini $\nu = 0.4$
2	61.26%	61.26%	61.37%
3	63.69%	64.21%	64.56%
4	64.7%	64.5%	65.64%
5	66.19%	65.62%	66.22%
6	66.67%	65.73%	66.54%
7	66.78%	66.76%	66.44%
8	66.73%	66.69%	66.41%
9	66.73%	66.28%	66.81%
10	66.37%	66.86%	66.55%

TABLE D.24: Comparison of actual Ginis by number of segments and value of ν for the wine quality data set

SSE Wgt (ν)	Average population % in smallest segment	Average population % in largest segment	Average normalised value of ϱ
0.0	18.31%	24.19%	94.91%
0.1	19.68%	23.17%	97.92%
0.2	19.84%	23.05%	98.45%
0.4	19.15%	23.76%	97.19%

TABLE D.25: Impact on population percentages by increasing ν for the chess data set

Distortion weight (w)	Average SSS-KMIV-SSE IV $\nu = 0.1$	Average SSS-KMIV-SSE IV value $\nu = 0.2$	Average SSS-KMIV-SSE IV $\nu = 0.4$	Average SSS-KMIV IV	Average SSS-KM-CSQ IV	Unsupervised IV	Decision tree IV
0.1	0.50	0.55	0.70	0.48	0.48	0.34	1.94
0.2	0.71	0.77	0.91	0.67	0.70	0.34	1.94
0.4	1.06	1.10	1.18	0.93	1.44	0.34	1.94

TABLE D.26: IV values by distortion weight for the chess data set

Distortion weight (w)	Average SSS-KMIV-SSE CH value $\nu = 0.1$	Average SSS-KMIV-SSE CH value $\nu = 0.2$	Average SSS-KMIV-SSE CH value $\nu = 0.4$	Average SSS-KMIV CH value	Average SSKM-CSQ CH value	Unsupervised CH value	Decision tree CH value
0.1	0.21	0.21	0.20	0.21	0.21	0.21	0.11
0.2	0.21	0.21	0.20	0.21	0.21	0.21	0.11
0.4	0.20	0.20	0.18	0.20	0.19	0.21	0.11

TABLE D.27: CH values by distortion weight for the chess data set

Number of segments	Best Gini improvement unsupervised	Best Gini improvement SSS-KMIV	Best Gini improvement SSS-KMCSQ	Best Gini improvement decision tree	Best Gini improvement SSS-KMIVSSE
2	5.22%	5.22%	5.22%	5.2%	5.22%
3	9.14%	9.02%	8.74%	8.66%	8.98%
4	9.62%	10.5%	9.69%	13.34%	9.52%
5	11.04%	11.42%	11.14%	13.68%	10.53%
6	12.28%	12.56%	11.98%	14.03%	12.29%
7	13.19%	13.22%	13.05%	15.83%	13.28%
8	13.98%	14.22%	14.15%	15.36%	14.15%
9	14.92%	15.04%	14.94%	16.96%	14.46%
10	15.61%	15.26%	15.15%	17.%	14.96%

TABLE D.28: Comparison of Gini improvement by number of segments for the chess data set

Number of segments	Best Gini improvement $\nu = 0.1$	Best Gini improvement $\nu = 0.2$	Best Gini improvement $\nu = 0.4$
2	5.22%	5.22%	4.68%
3	8.98%	8.96%	7.23%
4	9.19%	9.52%	8.76%
5	10.53%	10.09%	10.12%
6	11.76%	12.29%	11.28%
7	13.26%	13.28%	12.38%
8	14.12%	14.15%	13.65%
9	14.46%	14.3%	13.55%
10	14.96%	14.9%	14.37%

TABLE D.29: Comparison of Gini improvement by number of segments and value of ν for the chess data set

Number of segments	Best Gini $\nu = 0.1$	Best Gini $\nu = 0.2$	Best Gini $\nu = 0.4$
2	76.56%	76.56%	76.02%
3	80.32%	80.3%	78.57%
4	80.53%	80.86%	80.1%
5	81.87%	81.43%	81.46%
6	83.1%	83.63%	82.62%
7	84.6%	84.62%	83.72%
8	85.46%	85.49%	84.99%
9	85.8%	85.64%	84.89%
10	86.3%	86.24%	85.71%

TABLE D.30: Comparison of actual Ginis by number of segments and value of ν for the chess data set

SSE Wgt (ν)	Average population % in smallest segment	Average population % in largest segment	Average normalised value of ϱ
0.0	9.73%	34.64%	42.41%
0.1	12.47%	31.68%	62.85%
0.2	14.38%	28.2%	76.89%
0.4	14.31%	28.75%	72.5%

TABLE D.31: Impact on population percentages by increasing ν for the claim prediction data set

Distortion weight (w)	Average SSS-KMIV-SSE IV $\nu = 0.1$	Average SSS-KMIV-SSE IV value $\nu = 0.2$	Average SSS-KMIV-SSE IV $\nu = 0.4$	Average SSS-KMIV IV	Average SSS-KM-CSQ IV	Unsupervised IV	Decision tree IV
0.1	0.04	0.04	0.04	0.04	0.04	0.03	0.06
0.2	0.04	0.04	0.03	0.04	0.04	0.03	0.06
0.4	0.04	0.04	0.03	0.04	0.04	0.03	0.06

TABLE D.32: IV values by distortion weight for the claim prediction data set

Distortion weight (w)	Average SSS-KMIV-SSE CH value $\nu = 0.1$	Average SSS-KMIV-SSE CH value $\nu = 0.2$	Average SSS-KMIV-SSE CH value $\nu = 0.4$	Average SSS-KMIV CH value	Average SSKM-CSQ CH value	Unsupervised CH value	Decision tree CH value
0.1	0.39	0.38	0.35	0.38	0.39	0.38	0.09
0.2	0.37	0.36	0.33	0.37	0.36	0.38	0.09
0.4	0.35	0.34	0.33	0.35	0.34	0.38	0.09

TABLE D.33: CH values by distortion weight for the claim prediction data set

Number of segments	Best Gini improvement unsupervised	Best Gini improvement SSS-KMIV	Best Gini improvement SSS-KMCSQ	Best Gini improvement decision tree	Best Gini improvement SSS-KMIVSSE
2	1.01%	3.7%	3.54%	.77%	3.45%
3	3.2%	3.43%	4.39%	1.84%	3.%
4	3.39%	5.24%	5.27%	4.12%	4.64%
5	4.6%	5.21%	5.6%	4.56%	5.65%
6	4.89%	5.7%	5.7%	5.26%	5.86%
7	4.21%	5.76%	6.25%	5.26%	5.57%
8	4.37%	7.04%	6.54%	5.66%	5.61%
9	4.84%	7.04%	6.86%	4.68%	5.79%
10	4.77%	7.16%	6.31%	5.69%	7.21%

TABLE D.34: Comparison of Gini improvement by number of segments for the claim prediction data set

Number of segments	Best Gini improvement $\nu = 0.1$	Best Gini improvement $\nu = 0.2$	Best Gini improvement $\nu = 0.4$
2	2.98%	3.31%	3.45%
3	3.%	2.29%	2.86%
4	4.16%	4.58%	4.64%
5	5.08%	4.73%	5.65%
6	5.86%	5.55%	4.69%
7	5.57%	4.59%	4.31%
8	5.57%	5.41%	5.61%
9	5.34%	5.21%	5.79%
10	7.21%	6.12%	5.8%

TABLE D.35: Comparison of Gini improvement by number of segments and value of ν for the claim prediction data set

Number of segments	Best Gini $\nu = 0.1$	Best Gini $\nu = 0.2$	Best Gini $\nu = 0.4$
2	11.01%	11.34%	11.48%
3	11.03%	10.32%	10.89%
4	12.19%	12.61%	12.67%
5	13.11%	12.76%	13.68%
6	13.89%	13.58%	12.72%
7	13.59%	12.62%	12.34%
8	13.6%	13.44%	13.64%
9	13.37%	13.23%	13.82%
10	15.24%	14.15%	13.83%

TABLE D.36: Comparison of actual Ginis by number of segments and value of ν for the claim prediction data set

Distortion weight (w)	Avg IV SSS-KMIV	Avg CH value SSS-KMIV	Avg IV SSS-KM-CSQ	Avg CH value SSS-KM-CSQ	Avg IV SSS-Wong	Avg CH value SSS-Wong	Avg unsupervised IV	Avg unsupervised CH value	Avg dtree IV	Avg dtree CH value
0.1	0.1	0.2	0.1	0.16	0.38	0.07	0.08	0.16	0.23	0.09
0.2	0.1	0.2	0.1	0.16	0.39	0.07	0.08	0.16	0.23	0.09
0.3	0.1	0.2	0.1	0.15	0.39	0.07	0.08	0.16	0.23	0.09
0.4	0.1	0.2	0.1	0.15	0.41	0.07	0.08	0.16	0.23	0.09
0.5	0.1	0.2	0.1	0.15	0.42	0.07	0.08	0.16	0.23	0.09
0.6	0.1	0.1	0.1	0.14	0.43	0.08	0.08	0.16	0.23	0.09
0.7	0.1	0.1	0.1	0.14	0.44	0.08	0.08	0.16	0.23	0.09
0.8	0.1	0.1	0.1	0.14	0.44	0.08	0.08	0.16	0.23	0.09

TABLE D.37: CH and IV values by distortion weight for SSSWong for the direct marketing data set

Distortion weight (w)	Average IV SSS-Wong	Average Gini improvement SSSWong
0.1	0.38	-4.27%
0.2	0.39	-4.65%
0.3	0.39	-4.51%
0.4	0.41	-4.16%
0.5	0.42	-4.25%
0.6	0.43	-4.51%
0.7	0.44	-4.03%
0.8	0.44	-3.25%
0.9	0.45	-3.17%

TABLE D.38: Comparison of Gini and IV improvement for SSSWong for the direct marketing data set

Number of segments	Best Gini improvement unsupervised	Best Gini improvement SSS-KMIV	Best Gini improvement SSSKM-CSQ	Best Gini improvement decision tree	Best Gini improvement SSS-Wong	Best Gini improvement SSSKM-IVSSE
2	3.74%	3.74%	3.74%	8.37%	-2.94%	NA
3	4.0%	4.37%	4.12%	8.88%	-4.13%	NA
4	3.12%	4.39%	4.56%	10.16%	-4.36%	NA
5	3.53%	4.04%	3.69%	9.97%	-3.22%	NA
6	4.2%	4.34%	4.26%	10.3%	-4.49%	NA
7	2.08%	3.85%	4.87%	10.24%	-4.63%	NA
8	4.55%	3.72%	5.4%	10.68%	-4.57%	NA
9	5.13%	5.91%	7.16%	10.52%	-4.54%	NA
10	4.45%	5.49%	6.21%	11.72%	-3.93%	NA

TABLE D.39: Gini improvement by nr segments for SSSWong for the direct marketing data set

Number of segments	Actual Gini SSS-Wong
2	24.01%
3	22.81%
4	22.58%
5	23.72%
6	22.45%
7	22.32%
8	22.38%
9	22.41%
10	23.01%

TABLE D.40: Actual Ginis by nr of segments for SSSWong for the direct marketing data set

Distortion weight (w)	Avg IV SSS-KMIV	Avg CH value SSS-KMIV	Avg IV SSS-KM-CSQ	Avg CH value SSS-KM-CSQ	Avg IV SSS-Wong	Avg CH value SSS-Wong	Avg un-supervised IV	Avg un-supervised CH value	Avg dtree IV	Avg dtree CH value
0.1	0.4	0.8	0.5	0.77	0.91	0.31	0.36	0.78	0.76	0.32
0.2	0.5	0.7	0.6	0.71	0.95	0.30	0.36	0.78	0.76	0.32
0.3	0.6	0.7	0.7	0.70	0.97	0.28	0.36	0.78	0.76	0.32
0.4	0.6	0.7	0.8	0.68	1.03	0.24	0.36	0.78	0.76	0.32
0.5	0.7	0.7	0.8	0.66	1.10	0.25	0.36	0.78	0.76	0.32
0.6	0.8	0.6	0.8	0.62	1.13	0.26	0.36	0.78	0.76	0.32
0.7	0.8	0.6	0.8	0.59	1.14	0.27	0.36	0.78	0.76	0.32
0.8	0.7	0.6	0.7	0.54	1.16	0.28	0.36	0.78	0.76	0.32
0.9	0.7	0.6	0.7	0.41	1.19	0.27	0.36	0.78	0.76	0.32

TABLE D.41: CH and IV values by distortion weight for SSSWong for the protein tertiary structure data set

Distortion weight (w)	Average IV SSS-Wong	Average Gini improvement SSSWong
0.1	0.91	4.21%
0.2	0.95	4.33%
0.3	0.97	4.16%
0.4	1.03	4.26%
0.5	1.10	4.42%
0.6	1.13	4.51%
0.7	1.14	4.74%
0.8	1.16	4.82%
0.9	1.19	5.27%

TABLE D.42: Comparison of Gini and IV improvement for SSSWong for the protein tertiary structure data set

Number of segments	Best Gini improvement unsupervised	Best Gini improvement SSS-KMIV	Best Gini improvement SSSKM-CSQ	Best Gini improvement decision tree	Best Gini improvement SSS-Wong	Best Gini improvement SSSKM-IVSSE
2	.72%	1.9%	1.84%	1.58%	1.29%	NA
3	2.12%	3.41%	3.3%	2.62%	2.58%	NA
4	2.54%	4.82%	5.18%	3.96%	3.94%	NA
5	4.76%	5.58%	5.49%	4.62%	4.56%	NA
6	5.44%	6.77%	6.14%	4.96%	5.07%	NA
7	6.45%	7.1%	7.11%	5.83%	5.36%	NA
8	6.83%	11.33%	7.4%	6.06%	5.6%	NA
9	7.63%	8.05%	8.%	6.82%	6.03%	NA
10	7.84%	8.64%	8.95%	7.39%	6.31%	NA

TABLE D.43: Gini improvement by nr segments for SSSWong for the protein tertiary structure data set

Number of segments	Actual Gini SSS-Wong
2	28.24%
3	29.52%
4	30.88%
5	31.5%
6	32.01%
7	32.3%
8	32.54%
9	32.98%
10	33.25%

TABLE D.44: Actual Ginis by nr of segments for SSSWong for the protein tertiary structure data set

Distortion weight (w)	Avg IV SSS-KMIV	Avg CH value SSS-KMIV	Avg IV SSS-KM-CSQ	Avg CH value SSS-KM-CSQ	Avg IV SSS-Wong	Avg CH value SSS-Wong	Avg un-supervised IV	Avg un-supervised CH value	Avg dtree IV	Avg dtree CH value
0.1	0.2	0.3	0.2	0.28	0.40	0.23	0.19	0.27	1.14	0.03
0.2	0.2	0.3	0.2	0.28	0.41	0.23	0.19	0.27	1.14	0.03
0.3	0.2	0.3	0.2	0.28	0.41	0.23	0.19	0.27	1.14	0.03
0.4	0.2	0.3	0.3	0.27	0.41	0.23	0.19	0.27	1.14	0.03
0.5	0.3	0.3	0.3	0.25	0.43	0.24	0.19	0.27	1.14	0.03
0.6	0.3	0.2	0.3	0.23	0.43	0.24	0.19	0.27	1.14	0.03
0.7	0.3	0.2	0.3	0.23	0.44	0.23	0.19	0.27	1.14	0.03
0.8	0.3	0.2	0.3	0.22	0.44	0.23	0.19	0.27	1.14	0.03
0.9	0.3	0.2	0.3	0.22	0.45	0.22	0.19	0.27	1.14	0.03

TABLE D.45: CH and IV values by distortion weight for SSSWong for the credit application data set

Distortion weight (w)	Average IV SSS-Wong	Average Gini improvement SSSWong
0.1	0.40	22.67%
0.2	0.41	22.65%
0.3	0.41	22.7%
0.4	0.41	22.71%
0.5	0.43	21.45%
0.6	0.43	20.91%
0.7	0.44	21.67%
0.8	0.44	21.68%
0.9	0.45	21.69%

TABLE D.46: Comparison of Gini and IV improvement for SSSWong for the credit application data set

Number of segments	Best Gini improvement unsupervised	Best Gini improvement SSS-KMIV	Best Gini improvement SSSKM-CSQ	Best Gini improvement decision tree	Best Gini improvement SSS-Wong	Best Gini improvement SSSKM-IVSSE
2	22.39%	22.39%	18.7%	22.8%	15.95%	NA
3	23.01%	25.26%	24.24%	23.22%	22.09%	NA
4	23.06%	25.09%	24.55%	32.09%	22.42%	NA
5	23.01%	24.99%	23.92%	32.65%	22.85%	NA
6	24.48%	25.25%	24.98%	32.71%	22.75%	NA
7	25.02%	25.28%	26.81%	32.72%	22.74%	NA
8	24.96%	26.57%	26.06%	32.78%	22.85%	NA
9	25.03%	28.17%	27.31%	32.78%	23.23%	NA
10	24.69%	26.89%	26.55%	32.8%	23.26%	NA

TABLE D.47: Gini improvement by nr segments for SSSWong for the credit application data set

Number of segments	Actual Gini SSS-Wong
2	42.89%
3	49.03%
4	49.36%
5	49.8%
6	49.7%
7	49.68%
8	49.79%
9	50.17%
10	50.2%

TABLE D.48: Actual Ginis by nr of segments for SSSWong for the credit application data set

Distortion weight (w)	Avg IV SSS-KMIV	Avg CH value SSS-KMIV	Avg IV SSS-KM-CSQ	Avg CH value SSS-KM-CSQ	Avg IV SSS-Wong	Avg CH value SSS-Wong	Avg un-supervised IV	Avg un-supervised CH value	Avg dtree IV	Avg dtree CH value
0.1	0.7	0.4	0.7	0.38	1.26	0.17	0.64	0.37	0.98	0.11
0.2	0.7	0.4	0.8	0.37	1.31	0.16	0.64	0.37	0.98	0.11
0.3	0.7	0.4	0.8	0.36	1.41	0.16	0.64	0.37	0.98	0.11
0.4	0.8	0.4	0.8	0.35	1.45	0.17	0.64	0.37	0.98	0.11
0.5	0.8	0.4	0.9	0.34	1.62	0.17	0.64	0.37	0.98	0.11
0.6	0.9	0.3	0.9	0.33	1.63	0.16	0.64	0.37	0.98	0.11
0.7	0.9	0.3	0.9	0.32	1.65	0.16	0.64	0.37	0.98	0.11
0.8	0.8	0.3	0.9	0.31	1.63	0.16	0.64	0.37	0.98	0.11
0.9	0.8	0.3	0.9	0.30	1.67	0.17	0.64	0.37	0.98	0.11

TABLE D.49: CH and IV values by distortion weight for SSSWong for the wine quality data set

Distortion weight (w)	Average IV SSS-Wong	Average Gini improvement SSSWong
0.1	1.26	2.37%
0.2	1.31	2.87%
0.3	1.41	3.18%
0.4	1.45	3.25%
0.5	1.62	3.51%
0.6	1.63	3.29%
0.7	1.65	2.84%
0.8	1.63	3.31%
0.9	1.67	2.69%

TABLE D.50: Comparison of Gini and IV improvement for SSSWong for the wine quality data set

Number of segments	Best Gini improvement unsupervised	Best Gini improvement SSS-KMIV	Best Gini improvement SSSKM-CSQ	Best Gini improvement decision tree	Best Gini improvement SSS-Wong	Best Gini improvement SSSKM-IVSSE
2	.59%	3.67%	4.11%	2.3%	1.04%	NA
3	2.96%	4.79%	4.43%	2.7%	2.29%	NA
4	5.18%	5.27%	4.92%	3.07%	3.24%	NA
5	5.72%	4.93%	5.37%	3.5%	3.38%	NA
6	5.69%	5.94%	6.07%	4.49%	3.63%	NA
7	5.66%	6.31%	5.95%	4.52%	3.72%	NA
8	5.69%	6.42%	6.15%	5.87%	3.55%	NA
9	6.32%	6.18%	6.22%	5.76%	3.29%	NA
10	6.67%	6.37%	6.37%	5.96%	3.16%	NA

TABLE D.51: Gini improvement by nr segments for SSSWong for the wine quality data set

Number of segments	Actual Gini SSS-Wong
2	27.98%
3	29.24%
4	30.19%
5	30.33%
6	30.58%
7	30.67%
8	30.5%
9	30.24%
10	30.1%

TABLE D.52: Actual Ginis by nr of segments for SSSWong for the wine quality data set

Distortion weight (w)	Avg IV SSS-KMIV	Avg CH value SSS-KMIV	Avg IV SSS-KM-CSQ	Avg CH value SSS-KM-CSQ	Avg IV SSS-Wong	Avg CH value SSS-Wong	Avg un-supervised IV	Avg un-supervised CH value	Avg dtree IV	Avg dtree CH value
0.1	0.5	0.2	0.5	0.21	0.55	0.05	0.34	0.21	1.94	0.11
0.2	0.7	0.2	0.7	0.21	0.77	0.07	0.34	0.21	1.94	0.11
0.3	0.9	0.2	1.0	0.20	0.92	0.07	0.34	0.21	1.94	0.11
0.4	0.9	0.2	1.4	0.19	1.09	0.07	0.34	0.21	1.94	0.11
0.5	1.1	0.2	1.8	0.18	1.19	0.08	0.34	0.21	1.94	0.11
0.6	1.5	0.2	1.9	0.18	1.25	0.08	0.34	0.21	1.94	0.11
0.7	1.7	0.2	1.9	0.18	1.25	0.08	0.34	0.21	1.94	0.11
0.8	1.6	0.2	1.9	0.17	1.39	0.08	0.34	0.21	1.94	0.11
0.9	1.5	0.2	1.6	0.14	1.47	0.08	0.34	0.21	1.94	0.11

TABLE D.53: CH and IV values by distortion weight for SSSWong for the chess data set

Distortion weight (w)	Average IV SSS-Wong	Average Gini improvement SSSWong
0.1	0.55	1.87%
0.2	0.77	2.19%
0.3	0.92	2.5%
0.4	1.09	2.55%
0.5	1.19	2.71%
0.6	1.25	2.79%
0.7	1.25	3.18%
0.8	1.39	3.47%
0.9	1.47	3.47%

TABLE D.54: Comparison of Gini and IV improvement for SSSWong for the chess data set

Number of segments	Best Gini improvement unsupervised	Best Gini improvement SSS-KMIV	Best Gini improvement SSSKM-CSQ	Best Gini improvement decision tree	Best Gini improvement SSS-Wong	Best Gini improvement SSSKM-IVSSE
2	5.22%	5.22%	5.22%	5.2%	1.04%	NA
3	9.14%	9.02%	8.74%	8.66%	1.53%	NA
4	9.62%	10.5%	9.69%	13.34%	1.99%	NA
5	11.04%	11.42%	11.14%	13.68%	2.36%	NA
6	12.28%	12.56%	11.98%	14.03%	2.87%	NA
7	13.19%	13.22%	13.05%	15.83%	3.25%	NA
8	13.98%	14.22%	14.15%	15.36%	3.59%	NA
9	14.92%	15.04%	14.94%	16.96%	3.9%	NA
10	15.61%	15.26%	15.15%	17.%	4.18%	NA

TABLE D.55: Gini improvement by nr segments for SSSWong for the chess data set

Number of segments	Actual Gini SSS-Wong
2	27.98%
3	28.48%
4	28.94%
5	29.31%
6	29.81%
7	30.19%
8	30.53%
9	30.85%
10	31.13%

TABLE D.56: Actual Ginis by nr of segments for SSSWong for the chess data set

Distortion weight (w)	Avg IV SSS-KMIV	Avg CH value SSS-KMIV	Avg IV SSS-KM-CSQ	Avg CH value SSS-KM-CSQ	Avg IV SSS-Wong	Avg CH value SSS-Wong	Avg un-supervised IV	Avg un-supervised CH value	Avg dtree IV	Avg dtree CH value
0.1	0.0	0.4	0.0	0.39	0.08	0.06	0.03	0.38	0.06	0.09
0.2	0.0	0.4	0.0	0.36	0.07	0.06	0.03	0.38	0.06	0.09
0.3	0.0	0.4	0.0	0.35	0.08	0.06	0.03	0.38	0.06	0.09
0.4	0.0	0.4	0.0	0.34	0.09	0.07	0.03	0.38	0.06	0.09
0.5	0.0	0.3	0.0	0.33	0.10	0.07	0.03	0.38	0.06	0.09
0.6	0.0	0.3	0.0	0.31	0.10	0.08	0.03	0.38	0.06	0.09
0.7	0.0	0.3	0.0	0.30	0.10	0.08	0.03	0.38	0.06	0.09
0.8	0.0	0.3	0.0	0.28	0.11	0.09	0.03	0.38	0.06	0.09
0.9	0.0	0.3	0.0	0.23	0.11	0.08	0.03	0.38	0.06	0.09

TABLE D.57: CH and IV values by distortion weight for SSSWong for the claim prediction data set

Distortion weight (w)	Average IV SSS-Wong	Average Gini improvement SSSWong
0.1	0.08	2.19%
0.2	0.07	2.5%
0.3	0.08	2.62%
0.4	0.09	2.54%
0.5	0.10	3.04%
0.6	0.10	3.37%
0.7	0.10	3.92%
0.8	0.11	4.15%
0.9	0.11	4.26%

TABLE D.58: Comparison of Gini and IV improvement for SSSWong for the claim prediction data set

Number of segments	Best Gini improvement unsupervised	Best Gini improvement SSS-KMIV	Best Gini improvement SSSKM-CSQ	Best Gini improvement decision tree	Best Gini improvement SSS-Wong	Best Gini improvement SSSKM-IVSSE
2	1.01%	3.7%	3.54%	.77%	.54%	NA
3	3.2%	3.43%	4.39%	1.84%	1.5%	NA
4	3.39%	5.24%	5.27%	4.12%	2.22%	NA
5	4.6%	5.21%	5.6%	4.56%	2.93%	NA
6	4.89%	5.7%	5.7%	5.26%	3.4%	NA
7	4.21%	5.76%	6.25%	5.26%	4.21%	NA
8	4.37%	7.04%	6.54%	5.66%	4.38%	NA
9	4.84%	7.04%	6.86%	4.68%	4.63%	NA
10	4.77%	7.16%	6.31%	5.69%	4.77%	NA

TABLE D.59: Gini improvement by nr segments for SSSWong for the claim prediction data set

Number of segments	Actual Gini SSS-Wong
2	27.49%
3	28.44%
4	29.16%
5	29.88%
6	30.35%
7	31.15%
8	31.33%
9	31.57%
10	31.71%

TABLE D.60: Actual Ginis by nr of segments for SSSWong for the claim prediction data set

Distortion weight (w)	Avg IV SSS-KMIV	Avg CH value SSS-KMIV	Avg IV SSS-KM-CSQ	Avg CH value SSS-KM-CSQ	Avg IV LK-Means	Avg CH value LK-Means	Avg unsupervised IV	Avg unsupervised CH value	Avg dtree IV	Avg dtree CH value
0.1	0.1	0.2	0.1	0.16	0.09	0.16	0.08	0.16	0.23	0.09
0.2	0.1	0.2	0.1	0.16	0.08	0.16	0.08	0.16	0.23	0.09
0.3	0.1	0.2	0.1	0.15	0.08	0.16	0.08	0.16	0.23	0.09
0.4	0.1	0.2	0.1	0.15	0.08	0.16	0.08	0.16	0.23	0.09
0.5	0.1	0.2	0.1	0.15	0.08	0.15	0.08	0.16	0.23	0.09
0.6	0.1	0.1	0.1	0.14	0.10	0.14	0.08	0.16	0.23	0.09
0.7	0.1	0.1	0.1	0.14	0.11	0.13	0.08	0.16	0.23	0.09
0.8	0.1	0.1	0.1	0.14	0.12	0.11	0.08	0.16	0.23	0.09

TABLE D.61: CH and IV values by distortion weight for LKMeans for the direct marketing data set

Distortion weight (w)	Average IV LK-Means	Average Gini improvement LKMeans
0.1	0.09	-4.31%
0.2	0.08	-4.52%
0.3	0.08	-4.72%
0.4	0.08	-4.59%
0.5	0.08	-3.75%
0.6	0.10	-3.91%
0.7	0.11	-3.89%
0.8	0.12	-3.85%
0.9	0.13	-4.46%

TABLE D.62: Comparison of Gini and IV improvement for LKMeans for the direct marketing data set

Number of segments	Best Gini improvement unsupervised	Best Gini improvement SSS-KMIV	Best Gini improvement SSS-KM-CSQ	Best Gini improvement decision tree	Best Gini improvement LK-Means	Best Gini improvement SSS-KMIV-SSE	Best Gini improvement SSS-Wong
2	3.74%	3.74%	3.74%	8.37%	.15%	-1.15%	-2.94%
3	4.%	4.37%	4.12%	8.88%	.85%	1.55%	-4.13%
4	3.12%	4.39%	4.56%	10.16%	.68%	.06%	-4.36%
5	3.53%	4.04%	3.69%	9.97%	1.84%	.25%	-3.22%
6	4.2%	4.34%	4.26%	10.3%	1.8%	.5%	-4.49%
7	2.08%	3.85%	4.87%	10.24%	1.21%	-1.5%	-4.63%
8	4.55%	3.72%	5.4%	10.68%	.98%	-1.16%	-4.57%
9	5.13%	5.91%	7.16%	10.52%	2.31%	-.97%	-4.54%
10	4.45%	5.49%	6.21%	11.72%	1.49%	.13%	-3.93%

TABLE D.63: Gini improvement by nr segments for LKMeans for the direct marketing data set

Distortion weight (w)	Avg IV SSS-KMIV	Avg CH value SSS-KMIV	Avg IV SSS-KM-CSQ	Avg CH value SSS-KM-CSQ	Avg IV LK-Means	Avg CH value LK-Means	Avg unsupervised IV	Avg unsupervised CH value	Avg dtree IV	Avg dtree CH value
0.1	0.4	0.8	0.5	0.77	0.34	0.78	0.36	0.78	0.76	0.32
0.2	0.5	0.7	0.6	0.71	0.32	0.78	0.36	0.78	0.76	0.32
0.3	0.6	0.7	0.7	0.70	0.30	0.78	0.36	0.78	0.76	0.32
0.4	0.6	0.7	0.8	0.68	0.28	0.78	0.36	0.78	0.76	0.32
0.5	0.7	0.7	0.8	0.66	0.25	0.78	0.36	0.78	0.76	0.32
0.6	0.8	0.6	0.8	0.62	0.27	0.76	0.36	0.78	0.76	0.32
0.7	0.8	0.6	0.8	0.59	0.56	0.67	0.36	0.78	0.76	0.32
0.8	0.7	0.6	0.7	0.54	0.73	0.61	0.36	0.78	0.76	0.32
0.9	0.7	0.6	0.7	0.41	0.83	0.48	0.36	0.78	0.76	0.32

TABLE D.64: CH and IV values by distortion weight for LKMeans for the protein tertiary structure data set

Distortion weight (w)	Average IV LK-Means	Average Gini improvement LKMeans
0.1	0.34	4.46%
0.2	0.32	4.46%
0.3	0.30	4.6%
0.4	0.28	4.61%
0.5	0.25	4.67%
0.6	0.27	4.71%
0.7	0.56	5.46%
0.8	0.73	5.74%
0.9	0.83	5.85%

TABLE D.65: Comparison of Gini and IV improvement for LKMeans for the protein tertiary structure data set

Number of segments	Best Gini improvement unsupervised	Best Gini improvement SSS-KMIV	Best Gini improvement SSS-KM-CSQ	Best Gini improvement decision tree	Best Gini improvement LK-Means	Best Gini improvement SSS-KMIV-SSE	Best Gini improvement SSS-Wong
2	.72%	1.9%	1.84%	1.58%	3.11%	1.12%	1.29%
3	2.12%	3.41%	3.3%	2.62%	4.4%	3.74%	2.58%
4	2.54%	4.82%	5.18%	3.96%	6.43%	7.3%	3.94%
5	4.76%	5.58%	5.49%	4.62%	6.8%	8.26%	4.56%
6	5.44%	6.77%	6.14%	4.96%	7.55%	9.36%	5.07%
7	6.45%	7.1%	7.11%	5.83%	8.14%	8.32%	5.36%
8	6.83%	11.33%	7.4%	6.06%	9.12%	9.88%	5.6%
9	7.63%	8.05%	8.%	6.82%	9.28%	10.76%	6.03%
10	7.84%	8.64%	8.95%	7.39%	10.47%	11.1%	6.31%

TABLE D.66: Gini improvement by nr segments for LKMeans for the protein tertiary structure data set

Distortion weight (w)	Avg IV SSS-KMIV	Avg CH value SSS-KMIV	Avg IV SSS-KM-CSQ	Avg CH value SSS-KM-CSQ	Avg IV LK-Means	Avg CH value LK-Means	Avg un-supervised IV	Avg un-supervised CH value	Avg dtree IV	Avg dtree CH value
0.1	0.2	0.3	0.2	0.28	0.19	0.17	0.19	0.27	1.14	0.03
0.2	0.2	0.3	0.2	0.28	0.19	0.17	0.19	0.27	1.14	0.03
0.3	0.2	0.3	0.2	0.28	0.18	0.17	0.19	0.27	1.14	0.03
0.4	0.2	0.3	0.3	0.27	0.18	0.17	0.19	0.27	1.14	0.03
0.5	0.3	0.3	0.3	0.25	0.18	0.17	0.19	0.27	1.14	0.03
0.6	0.3	0.2	0.3	0.23	0.18	0.17	0.19	0.27	1.14	0.03
0.7	0.3	0.2	0.3	0.23	0.18	0.16	0.19	0.27	1.14	0.03
0.8	0.3	0.2	0.3	0.22	0.18	0.16	0.19	0.27	1.14	0.03
0.9	0.3	0.2	0.3	0.22	0.18	0.16	0.19	0.27	1.14	0.03

TABLE D.67: CH and IV values by distortion weight for LKMeans for the credit application data set

Distortion weight (w)	Average IV LK-Means	Average Gini improvement LKMeans
0.1	0.19	14.85%
0.2	0.19	14.65%
0.3	0.18	13.88%
0.4	0.18	14.9%
0.5	0.18	14.36%
0.6	0.18	14.94%
0.7	0.18	14.7%
0.8	0.18	15.06%
0.9	0.18	17.23%

TABLE D.68: Comparison of Gini and IV improvement for LKMeans for the credit application data set

Number of segments	Best Gini improvement unsupervised	Best Gini improvement SSS-KMIV	Best Gini improvement SSS-KM-CSQ	Best Gini improvement decision tree	Best Gini improvement LK-Means	Best Gini improvement SSS-KMIV-SSE	Best Gini improvement SSS-Wong
2	22.39%	22.39%	18.7%	22.8%	22.72%	22.39%	15.95%
3	23.01%	25.26%	24.24%	23.22%	23.33%	23.15%	22.09%
4	23.06%	25.09%	24.55%	32.09%	23.91%	24.8%	22.42%
5	23.01%	24.99%	23.92%	32.65%	23.26%	24.75%	22.85%
6	24.48%	25.25%	24.98%	32.71%	25.83%	25.28%	22.75%
7	25.02%	25.28%	26.81%	32.72%	26.38%	25.55%	22.74%
8	24.96%	26.57%	26.06%	32.78%	26.46%	26.09%	22.85%
9	25.03%	28.17%	27.31%	32.78%	25.05%	25.75%	23.23%
10	24.69%	26.89%	26.55%	32.8%	25.93%	27.36%	23.26%

TABLE D.69: Gini improvement by nr segments for LKMeans for the credit application data set

Distortion weight (w)	Avg IV SSS-KMIV	Avg CH value SSS-KMIV	Avg IV SSS-KM-CSQ	Avg CH value SSS-KM-CSQ	Avg IV LK-Means	Avg CH value LK-Means	Avg unsupervised IV	Avg unsupervised CH value	Avg dtree IV	Avg dtree CH value
0.1	0.7	0.4	0.7	0.38	0.62	0.37	0.64	0.37	0.98	0.11
0.2	0.7	0.4	0.8	0.37	0.61	0.38	0.64	0.37	0.98	0.11
0.3	0.7	0.4	0.8	0.36	0.56	0.37	0.64	0.37	0.98	0.11
0.4	0.8	0.4	0.8	0.35	0.52	0.37	0.64	0.37	0.98	0.11
0.5	0.8	0.4	0.9	0.34	0.53	0.37	0.64	0.37	0.98	0.11
0.6	0.9	0.3	0.9	0.33	0.66	0.34	0.64	0.37	0.98	0.11
0.7	0.9	0.3	0.9	0.32	0.79	0.31	0.64	0.37	0.98	0.11
0.8	0.8	0.3	0.9	0.31	0.97	0.28	0.64	0.37	0.98	0.11
0.9	0.8	0.3	0.9	0.30	1.16	0.24	0.64	0.37	0.98	0.11

TABLE D.70: CH and IV values by distortion weight for LKMeans for the wine quality data set

Distortion weight (w)	Average IV LK-Means	Average Gini improvement LKMeans
0.1	0.62	3.19%
0.2	0.61	3.3%
0.3	0.56	3.46%
0.4	0.52	3.49%
0.5	0.53	3.51%
0.6	0.66	3.85%
0.7	0.79	3.98%
0.8	0.97	4.07%
0.9	1.16	4.01%

TABLE D.71: Comparison of Gini and IV improvement for LKMeans for the wine quality data set

Number of segments	Best Gini improvement unsu-per-vised	Best Gini improvement SSS-KMIV	Best Gini improvement SSS-KM-CSQ	Best Gini improvement deci-sion tree	Best Gini improvement LK-Means	Best Gini improvement SSS-KMIV-SSE	Best Gini improvement SSS-Wong
2	.59%	3.67%	4.11%	2.3%	2.7%	.82%	1.04%
3	2.96%	4.79%	4.43%	2.7%	4.82%	4.01%	2.29%
4	5.18%	5.27%	4.92%	3.07%	5.89%	5.1%	3.24%
5	5.72%	4.93%	5.37%	3.5%	6.42%	5.67%	3.38%
6	5.69%	5.94%	6.07%	4.49%	6.95%	6.13%	3.63%
7	5.66%	6.31%	5.95%	4.52%	6.86%	6.23%	3.72%
8	5.69%	6.42%	6.15%	5.87%	7.51%	6.18%	3.55%
9	6.32%	6.18%	6.22%	5.76%	7.18%	6.27%	3.29%
10	6.67%	6.37%	6.37%	5.96%	6.85%	6.32%	3.16%

TABLE D.72: Gini improvement by nr segments for LKMeans for the wine quality data set

Distortion weight (w)	Avg IV SSS-KMIV	Avg CH value SSS-KMIV	Avg IV SSS-KM-CSQ	Avg CH value SSS-KM-CSQ	Avg IV LK-Means	Avg CH value LK-Means	Avg un-supervised IV	Avg un-supervised CH value	Avg dtree IV	Avg dtree CH value
0.1	0.5	0.2	0.5	0.21	0.24	0.21	0.34	0.21	1.94	0.11
0.2	0.7	0.2	0.7	0.21	0.19	0.21	0.34	0.21	1.94	0.11
0.3	0.9	0.2	1.0	0.20	0.17	0.21	0.34	0.21	1.94	0.11
0.4	0.9	0.2	1.4	0.19	0.42	0.20	0.34	0.21	1.94	0.11
0.5	1.1	0.2	1.8	0.18	1.40	0.17	0.34	0.21	1.94	0.11
0.6	1.5	0.2	1.9	0.18	1.76	0.16	0.34	0.21	1.94	0.11
0.7	1.7	0.2	1.9	0.18	1.84	0.15	0.34	0.21	1.94	0.11
0.8	1.6	0.2	1.9	0.17	1.87	0.14	0.34	0.21	1.94	0.11
0.9	1.5	0.2	1.6	0.14	1.95	0.13	0.34	0.21	1.94	0.11

TABLE D.73: CH and IV values by distortion weight for LKMeans for the chess data set

Distortion weight (w)	Average IV LK-Means	Average Gini improvement LKMeans
0.1	0.55	1.87%
0.2	0.77	2.19%
0.3	0.92	2.5%
0.4	1.09	2.55%
0.5	1.19	2.71%
0.6	1.25	2.79%
0.7	1.25	3.18%
0.8	1.39	3.47%
0.9	1.47	3.47%

TABLE D.74: Comparison of Gini and IV improvement for LKMeans for the chess data set

Number of segments	Best Gini improvement unsupervised	Best Gini improvement SSS-KMIV	Best Gini improvement SSS-KM-CSQ	Best Gini improvement decision tree	Best Gini improvement LK-Means	Best Gini improvement SSS-KMIV-SSE	Best Gini improvement SSS-Wong
2	5.22%	5.22%	5.22%	5.2%	5.42%	5.22%	1.04%
3	9.14%	9.02%	8.74%	8.66%	8.89%	8.98%	1.53%
4	9.62%	10.5%	9.69%	13.34%	10.64%	9.52%	1.99%
5	11.04%	11.42%	11.14%	13.68%	11.92%	10.53%	2.36%
6	12.28%	12.56%	11.98%	14.03%	13.67%	12.29%	2.87%
7	13.19%	13.22%	13.05%	15.83%	14.42%	13.28%	3.25%
8	13.98%	14.22%	14.15%	15.36%	15.05%	14.15%	3.59%
9	14.92%	15.04%	14.94%	16.96%	15.35%	14.46%	3.9%
10	15.61%	15.26%	15.15%	17.%	15.99%	14.96%	4.18%

TABLE D.75: Gini improvement by nr segments for LKMeans for the chess data set

Distortion weight (w)	Avg IV SSS-KMIV	Avg CH value SSS-KMIV	Avg IV SSS-KM-CSQ	Avg CH value SSS-KM-CSQ	Avg IV LK-Means	Avg CH value LK-Means	Avg unsupervised IV	Avg unsupervised CH value	Avg dtree IV	Avg dtree CH value
0.1	0.0	0.4	0.0	0.39	0.03	0.37	0.03	0.38	0.06	0.09
0.2	0.0	0.4	0.0	0.36	0.03	0.37	0.03	0.38	0.06	0.09
0.3	0.0	0.4	0.0	0.35	0.03	0.37	0.03	0.38	0.06	0.09
0.4	0.0	0.4	0.0	0.34	0.03	0.37	0.03	0.38	0.06	0.09
0.5	0.0	0.3	0.0	0.33	0.03	0.36	0.03	0.38	0.06	0.09
0.6	0.0	0.3	0.0	0.31	0.04	0.34	0.03	0.38	0.06	0.09
0.7	0.0	0.3	0.0	0.30	0.04	0.31	0.03	0.38	0.06	0.09
0.8	0.0	0.3	0.0	0.28	0.04	0.29	0.03	0.38	0.06	0.09
0.9	0.0	0.3	0.0	0.23	0.04	0.25	0.03	0.38	0.06	0.09

TABLE D.76: CH and IV values by distortion weight for LKMeans for the claim prediction data set

Distortion weight (w)	Average IV LK-Means	Average Gini improvement LKMeans
0.1	0.03	2.28%
0.2	0.03	2.35%
0.3	0.03	2.41%
0.4	0.03	2.62%
0.5	0.03	2.79%
0.6	0.04	3.19%
0.7	0.04	3.36%
0.8	0.04	3.14%
0.9	0.04	3.22%

TABLE D.77: Comparison of Gini and IV improvement for LKMeans for the claim prediction data set

Number of segments	Best Gini improvement unsu-per-vised	Best Gini improvement SSS-KMIV	Best Gini improvement SSS-KM-CSQ	Best Gini improvement deci-sion tree	Best Gini improvement LK-Means	Best Gini improvement SSS-KMIV-SSE	Best Gini improvement SSS-Wong
2	1.01%	3.7%	3.54%	.77%	3.73%	3.45%	.54%
3	3.2%	3.43%	4.39%	1.84%	5.46%	3.%	1.5%
4	3.39%	5.24%	5.27%	4.12%	6.01%	4.64%	2.22%
5	4.6%	5.21%	5.6%	4.56%	5.87%	5.65%	2.93%
6	4.89%	5.7%	5.7%	5.26%	5.92%	5.86%	3.4%
7	4.21%	5.76%	6.25%	5.26%	6.47%	5.57%	4.21%
8	4.37%	7.04%	6.54%	5.66%	6.43%	5.61%	4.38%
9	4.84%	7.04%	6.86%	4.68%	6.73%	5.79%	4.63%
10	4.77%	7.16%	6.31%	5.69%	6.73%	7.21%	4.77%

TABLE D.78: Gini improvement by nr segments for LKMeans for the claim prediction data set

Bibliography

- Allenby, G. M. and Rossi, P. E. (1998), ‘Marketing models of consumer heterogeneity’, *Journal of Econometrics* **89**, 57–78.
- Anderberg, M. R. (1973), Cluster analysis for applications, Technical report, DTIC Document.
- Andersen, R. (2008), *Modern methods for robust regression*, number 152, Sage.
- Anderson, R. (2007), *The Credit Scoring Toolkit*, Oxford University Press, New York.
- Anderson, R. (2015), Piecewise logistic regression: an application in credit scoring, Credit Scoring and Control Conference XIV.
- Anfinsen, C. B. (1973), ‘Principles that govern the folding of protein chains’, *Science* **181**(4096), 223–230.
- Anttila-Hughes, J. K. and Hsiang, S. M. (2013), ‘Destruction, disinvestment, and death: Economic and human losses following environmental disaster’, *Available at SSRN 2220501*.
- Baba, H., Takahara, J.-i., Yamashita, F. and Hashida, M. (2015), ‘Modeling and prediction of solvent effect on human skin permeability using support vector regression and random forest’, *Pharmaceutical research* pp. 1–14.
- Bäck, T. and Schwefel, H.-P. (1993), ‘An overview of evolutionary algorithms for parameter optimization’, *Evolutionary computation* **1**(1), 1–23.
- Badrinarayanan, V., Budvytis, I. and Cipolla, R. (2013), ‘Semi-supervised video segmentation using tree structured graphical models’, *Pattern Analysis and Machine Intelligence, IEEE Transactions on* **35**(11), 2751–2764.
- Bahnsen, A. C., Aouada, D. and Ottersten, B. (2014), Example-dependent cost-sensitive logistic regression for credit scoring, in ‘Machine Learning and Applications (ICMLA), 2014 13th International Conference on’, IEEE, pp. 263–269.
- Bain, M. (1994), Learning logical exceptions in chess, PhD thesis, Citeseer.

- Bain, M. (2014), Experiments in non-monotonic learning, in ‘Proceedings of the Eighth International Workshop on Machine Learning’, pp. 380–384.
- Bair, E. (2013), ‘Semi-supervised clustering methods’, *Wiley Interdisciplinary Reviews: Computational Statistics* **5**(5), 349–361.
- Basu, S., Banerjee, A. and Mooney, R. (2002), Semi-supervised clustering by seeding, in ‘In Proceedings of 19th International Conference on Machine Learning (ICML-2002’, Citeseer.
- Basu, S., Bilenko, M. and Mooney, R. J. (2004), A probabilistic framework for semi-supervised clustering, in ‘Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining’, ACM, pp. 59–68.
- Baum, L. E. and Petrie, T. (1966), ‘Statistical inference for probabilistic functions of finite state Markov chains’, *The Annals of Mathematical Statistics* **37**(6), 1554–1563.
- Beard, J. C., Epstein, C. and Chamberlain, R. D. (2015), Online automated reliability classification of queueing models for streaming processing using support vector machines, in ‘Euro-Par 2015: Parallel Processing’, Springer, pp. 82–93.
- Bijnen, E. J., Stouthard, P. C. and Brand-Maher, C. (1973), *Cluster analysis: Survey and evaluation of techniques*, Tilburg University Press Tilburg.
- Bilenko, M Basu, S. (2004), ‘Integrating constraints and metric learning in semi-supervised clustering’, *Proceedings of ICML* .
- Boser, B. E., Guyon, I. M. and Vapnik, V. N. (1992), A training algorithm for optimal margin classifiers, in ‘Proceedings of the fifth annual workshop on Computational learning theory’, ACM, pp. 144–152.
- Bradley, P. S. and Fayyad, U. M. (1998), Refining initial points for k-means clustering., in ‘ICML’, Vol. 98, Citeseer, pp. 91–99.
- Bramer, M. (2014), ‘Machine-aided refinement of correct strategies for the endgame in chess’, *Advances in computer chess* **3**, 93–112.
- Breed, D., De La Rey, T. and Terblanche, S. (2013), The use of different clustering algorithms and distortion functions in semi supervised segmentation, in ‘Proceedings of the 42nd ORSSA Annual Conference’, Stellenbosch, pp. 122–132.
- Breiman, L. (1996a), ‘Bagging predictors’, *Machine learning* **24**(2), 123–140.
- Breiman, L. (1996b), ‘Technical note: Some properties of splitting criteria’, *Machine Learning* **24**(1), 41–47.

- Breiman, L. (2001), ‘Random forests’, *Machine learning* **45**(1), 5–32.
- Breiman, L., Fredman, J., Olsen, R. and Stone, C. (1984), *Classification and Regression Trees*, Wadsworth, Belmont, California.
- Brito, P. Q., Soares, C., Almeida, S., Monte, A. and Byvoet, M. (2015), ‘Customer segmentation in a large database of an online customized fashion business’, *Robotics and Computer-Integrated Manufacturing* (0), –.
- Bruand, J., Alexandrov, T., Sistla, S., Wisztorski, M., Meriaux, C., Becker, M., Salzet, M., Fournier, I., Macagno, E. and Bafna, V. (2011), ‘AMass: algorithm for msi analysis by semi-supervised segmentation’, *Journal of proteome research* **10**(10), 4734–4743.
- Caliński, T. and Harabasz, J. (1974), ‘A dendrite method for cluster analysis’, *Communications in Statistics-theory and Methods* **3**(1), 1–27.
- Caruana, R. and Niculescu-Mizil, A. (2006), An empirical comparison of supervised learning algorithms, in ‘Proceedings of the 23rd International Conference on Machine Learning’, ICML ’06, ACM, New York, NY, USA, pp. 161–168.
- Casabayó, M., Agell, N. and Sánchez-Hernández, G. (2015), ‘Improved market segmentation by fuzzifying crisp clusters: A case study of the energy market in Spain’, *Expert Systems with Applications* **42**(3), 1637–1643.
- Casti, J. L. and Casti, J. (1996), *Five golden rules: great theories of 20th-century mathematics and why they matter*, John Wiley & Sons.
- Chang, C.-Y., Hsu, M.-T., Esposito, E. X. and Tseng, Y. J. (2013), ‘Oversampling to overcome overfitting: exploring the relationship between data set composition, molecular descriptors, and predictive modeling methods’, *Journal of chemical information and modeling* **53**(4), 958–971.
- Ciurte, A., Bresson, X., Cuisenaire, O., Houhou, N., Nedevschi, S., Thiran, J.-P. and Cuadra, M. B. (2014), ‘Semi-supervised segmentation of ultrasound images based on patch representation and continuous min cut’, *PloS one* **9**(7), e100972.
- Clarke, M. (1977), A Quantitative Study of King and Pawn Against King, in M. R. B. Clarke, ed., ‘Advances in Computer Chess’, Vol. 1, Edinburgh University Press, Edinburgh, pp. 108–118.
- Coello, C. A. C., Van Veldhuizen, D. A. and Lamont, G. B. (2002), *Evolutionary algorithms for solving multi-objective problems*, Vol. 242, Springer.
- Cohen, W. W. (2014), Compiling prior knowledge into an explicit bias, in ‘Proceedings of the Ninth International Conference on Machine Learning’, pp. 102–110.

- Cohn, D., Caruana, R. and McCallum, A. (2003), Semi-supervised clustering with user feedback, Technical report.
- Cortes, C. and Vapnik, V. (1995), 'Support-vector networks', *Machine Learning* **20**(3), 273–297.
- Cortez, P., Cerdeira, A., Almeida, F., Matos, T. and Reis, J. (2009), 'Modeling wine preferences by data mining from physicochemical properties', *Decision Support Systems* **47**(4), 547–553. Smart Business Networks: Concepts and Empirical Evidence.
- Cover, T. M. and Thomas, J. A. (2006), *Elements of Information Theory (Wiley Series in Telecommunications and Signal Processing)*, Wiley-Interscience.
- Cox, D. R. (1958), 'The regression analysis of binary sequences', *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 215–242.
- Cozzetto, D., Kryshchak, A., Fidelis, K., Moulton, J., Rost, B. and Tramontano, A. (2009), 'Evaluation of template-based models in CASp8 with standard measures', *Proteins: Structure, Function, and Bioinformatics* **77**(S9), 18–28.
- Cramer, J. S. (2002), *The origins of logistic regression*, Tinbergen Institute Working Paper.
- Cristianini, N. and Shawe-Taylor, J. (2000), *An introduction to support vector machines and other kernel-based learning methods*, Cambridge university press.
- Cross, G. (2008), 'Understanding your customer: segmentation techniques for gaining customer insight and predicting risk in the telecom industry'.
- Dal Pozzolo, A., Moro, G., Bontempi, G. and Le Borgne, D. Y. A. (2010), Comparison of Data Mining Techniques for Insurance Claim Prediction, PhD thesis, PhD thesis, University of Bologna.
- Davies, D. L. and Bouldin, D. W. (1979), 'A cluster separation measure', *Pattern Analysis and Machine Intelligence, IEEE Transactions on* (2), 224–227.
- Demiriz, A., Bennett, K. and Embrechts, M. J. (1999), Semi-supervised clustering using genetic algorithms, in 'In Artificial Neural Networks in Engineering (ANNIE-99', ASME Press, pp. 809–814.
- Devroye, L. (1996), *A probabilistic theory of pattern recognition*, Vol. 31, Springer.
- Dhingra, P. and Jayaram, B. (2013), 'A homology/ab initio hybrid algorithm for sampling near-native protein conformations', *Journal of computational chemistry* **34**(22), 1925–1936.

- Duda, R. O., Hart, P. E. and Stork, D. G. (2000), 'Pattern classification (pt. 1)'.
- Dudani, S. A. (1976), 'The distance-weighted k-nearest-neighbor rule', *Systems, Man and Cybernetics, IEEE Transactions on* (4), 325–327.
- Dunn, J. C. (1973), 'A fuzzy relative of the ISodata process and its use in detecting compact well-separated clusters'.
- Dutta, S., Bhattacharya, S. and Guin, K. (2015), Data mining in market segmentation: A literature review and suggestions, in K. N. Das, K. Deep, M. Pant, J. C. Bansal and A. Nagar, eds, 'Proceedings of Fourth International Conference on Soft Computing for Problem Solving', Vol. 335 of *Advances in Intelligent Systems and Computing*, Springer India, pp. 87–98.
- Eick, C. F., Zeidat, N. and Zhao, Z. (2004), Supervised clustering algorithms and benefits, in 'In proceedings of the 16th IEEE International Conference on Tools with Artificial Intelligence (ICTAI04) , Boca', pp. 774–776.
- Fasshauer, G. E., Hickernell, F. J. and Ye, Q. (2015), 'Solving support vector machines in reproducing kernel banach spaces with positive definite functions', *Applied and Computational Harmonic Analysis* **38**(1), 115–139.
- Fernández, C., Huerta, I. and Prati, A. (2015), A comparative evaluation of regression learning algorithms for facial age estimation, in Q. Ji, T. B. Moeslund, G. Hua and K. Nasrollahi, eds, 'Face and Facial Expression Recognition from Real World Videos', Vol. 8912 of *Lecture Notes in Computer Science*, Springer International Publishing, pp. 133–144.
- Fico (2014), Using segmented models for better decisions, Technical report, Fair Isaacs.
- Fonseca, C. M. and Fleming, P. J. (1995), 'An overview of evolutionary algorithms in multiobjective optimization', *Evolutionary Computation* **3**, 1–16.
- Fowlkes, E. B. and Mallows, C. L. (1983), 'A method for comparing two hierarchical clusterings', *Journal of the American statistical association* **78**(383), 553–569.
- Fraenkel, A. (1993), 'Complexity of protein folding', *Bulletin of Mathematical Biology* **55**(6), 1199–1210.
- Francis, L. (2003), Martian chronicles: Is mars better than neural networks?, in 'Casualty Actuarial Society Forum', pp. 75–102.
- Freund, Y. and Schapire, R. E. (1997), 'A decision-theoretic generalization of on-line learning and an application to boosting', *Journal of computer and system sciences* **55**(1), 119–139.

- Friedman, J. (1996), 'Another approach to polychotomous classification', *Dept. Statist., Stanford Univ., Stanford, CA, USA, Tech. Rep.*
- Friedman, J. H. (2001), 'Greedy function approximation: a gradient boosting machine', *Annals of statistics* pp. 1189–1232.
- Friedman, J., Hastie, T. and Tibshirani, R. (2001), *The elements of statistical learning*, Vol. 1, Springer series in statistics Springer, Berlin.
- Garcia-Moreno, B., ed. (1995–2014), *Proteins: Structure, Function, and Bioinformatics*, Wiley Online Library.
- Ghose Dasgupta, C., Dispensa, G. S. and Ghose, S. (1994), 'Comparing the predictive performance of a neural network model with some traditional market response models', *International Journal of Forecasting* **10**, 235–244.
- Gou, J., Xiong, T. and Kuang, Y. (2011), 'A novel weighted voting for k-nearest neighbor rule', *Journal of Computers* **6**(5), 833–840.
- Girra, N., Crucianu, M. and Boujemaa, N. (2005), Unsupervised and semi-supervised clustering: a brief survey, in 'in 'A Review of Machine Learning Techniques for Processing Multimedia Content', Report of the MUSCLE European Network of Excellence (FP6'.
- Györfi, L. and Krzyzak, A. (2002), *A distribution-free theory of nonparametric regression*, Springer, New York.
- Halkidi, M., Batistakis, Y. and Vazirgiannis, M. (2002a), 'Cluster validity methods: part i', *ACM Sigmod Record* **31**(2), 40–45.
- Halkidi, M., Batistakis, Y. and Vazirgiannis, M. (2002b), 'Clustering validity checking methods: part ii', *ACM Sigmod Record* **31**(3), 19–27.
- Hall, P., Park, B. U. and Samworth, R. J. (2008), 'Choice of neighbor order in nearest-neighbor classification', *The Annals of Statistics* pp. 2135–2152.
- Hand, D. J. (1997), *Construction and Assessment of Classification Rules*, John Wiley & Sons Ltd, West Sussex.
- Hand, D. J. and Till, R. J. (2001), 'A simple generalisation of the area under the roc curve for multiple class classification problems', *Machine learning* **45**(2), 171–186.
- Haq, R., Aras, R., Besachio, D. A., Borgie, R. C. and Audette, M. A. (2015), 'Minimally supervised segmentation and meshing of 3D intervertebral discs of the lumbar spine for discectomy simulation', *Recent Advances in Computational Methods and Clinical Applications for Spine Imaging* p. 143.

- Harremoës, P. and Tishby, N. (2007), The information bottleneck revisited or how to choose a good distortion measure, *in* 'Information Theory, 2007. ISIT 2007. IEEE International Symposium on', IEEE, pp. 566–570.
- Hartigan, J. A. (1975), *Clustering algorithms*, 1 edn, John Wiley & Sons, New York.
- Horák, M. (2009), 'Prediction of wine quality from physicochemical properties', *Alcohol* **8**(5), 10.
- Hosmer, D. W., Lemeshow, S. and Sturdivant, R. X. (2013), *Applied Logistic Regression*, 3 edn, John Wiley & Sons, Hoboken, New Jersey.
- Houhou, N., Bresson, X., Szlam, A., Chan, T. F. and Thiran, J.-P. (2009), Semi-supervised segmentation based on non-local continuous min-cut, *in* 'Scale Space and Variational Methods in Computer Vision', Springer, pp. 112–123.
- Hubert, L. and Arabie, P. (1985), 'Comparing partitions', *Journal of classification* **2**(1), 193–218.
- Hyafil, L. and Rivest, R. . C. O. B. D. T. i. N.-c. I. P. L. . . .-d.-.-. (1976), 'Constructing optimal binary decision trees is np-complete', *Information Processing Letters* **5** (1) pp. 15–17.
- Jaccard, P. (1901), *Etude comparative de la distribution florale dans une portion des Alpes et du Jura*, Impr. Corbaz.
- Jaccard, P. (1912), 'The distribution of the flora in the alpine zone. 1', *New phytologist* **11**(2), 37–50.
- Jain, A. K. (2010), 'Data clustering: 50 years beyond k-means', *Pattern recognition letters* **31**(8), 651–666.
- Jardine, N. and Sibson, R. (1971), *Mathematical taxonomy*, John Wiley and Sons.
- Jayaram, B., Dhingra, P., Lakhani, B. and Shekhar, S. (2012), 'Targeting the near impossible: Pushing the frontiers of atomic models for protein tertiary structure prediction#', *Journal of Chemical Sciences* **124**(1), 83–91.
- Kaggle Website (2016).
URL: <http://www.kaggle.com>
- Kang, P. and Cho, S. (2009), K-means clustering seeds initialization based on centrality, sparsity, and isotropy, *in* E. Corchado and H. Yin, eds, 'Intelligent Data Engineering and Automated Learning - IDEAL 2009', Vol. 5788 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, pp. 109–117.

- Kass, G. V. (1980), ‘An exploratory technique for investigating large quantities of categorical data’, *Applied statistics* pp. 119–127.
- Kearns, M. and Valiant, L. (1994), ‘Cryptographic limitations on learning boolean formulae and finite automata’, *Journal of the ACM (JACM)* **41**(1), 67–95.
- Khan, S. S. and Ahmad, A. (2004), ‘Cluster center initialization algorithm for k-means clustering’, *Pattern Recognition Letters* **25**(11), 1293–1302.
- Kim, D. E., Chivian, D. and Baker, D. (2004), ‘Protein structure prediction and analysis using the robetta server’, *Nucleic acids research* **32**(suppl 2), W526–W531.
- Klein, D., Kamvar, S. D. and Manning, C. D. (2002), ‘From instance-level constraints to space-level constraints: Making the most of prior knowledge in data clustering’.
- Kreßel, U. H.-G. (1999), Pairwise classification and support vector machines, in ‘Advances in kernel methods’, MIT Press, pp. 255–268.
- Kullback, S. (1958), *Information Theory and Statistics*, 1 edn, John Wiley and Sons, New York.
- Lambert, C., Léonard, N., De Bolle, X. and Depiereux, E. (2002), ‘Esyspred3d: Prediction of proteins 3D structures’, *Bioinformatics* **18**(9), 1250–1256.
- Lance, G. N. and Williams, W. T. (1967), ‘A general theory of classificatory sorting strategies i. hierarchical systems’, *The computer journal* **9**, 373–380.
- Li, L. (2015), Support vector machines, in ‘Selected Applications of Convex Optimization’, Springer, pp. 17–52.
- Lichman, M. (2013), ‘UCI machine learning repository’. Date accessed: 2016-06-05.
URL: <http://archive.ics.uci.edu/ml>
- MacLeod, J. E., Luk, A. and Titterington, D. M. (1987), ‘A re-examination of the distance-weighted k-nearest neighbor classification rule’, *Systems, Man and Cybernetics, IEEE Transactions on* **17**(4), 689–696.
- Maclin, R. and Opitz, D. (1997), ‘An empirical evaluation of bagging and boosting’, *AAAI/IAAI* **1997**, 546–551.
- Maimon, O. (2007), *Data Mining With Decision Trees: Theory and Applications. Series in Machine Perception & Artificial Intelligence*, World Scientific.
- Maroco, J., Silva, D., Rodrigues, A., Guerreiro, M., Santana, I. and de Mendonca, A. (2011), ‘Data mining methods in the prediction of dementia: A real-data comparison

- of the accuracy, sensitivity and specificity of linear discriminant analysis, logistic regression, neural networks, support vector machines, classification trees and random forests', *BMC Research Notes* **4**(1), 299.
- Massart, D. L. and Kaufman, L. (1983), *The interpretation of analytical chemical data by the use of cluster analysis*, Wiley.
- McAfee, A., Brynjolfsson, E., Davenport, T. H., Patil, D. and Barton, D. (2012), 'Big data', *The management revolution. Harvard Bus Rev* **90**(10), 61–67.
- McCullagh, P., Nelder, J. A. and McCullagh, P. (1989), *Generalized linear models*, Vol. 2, Chapman and Hall London.
- McCulloch, W. S. and Pitts, W. (1943), 'A logical calculus of the ideas immanent in nervous activity', *The bulletin of mathematical biophysics* **5**(4), 115–133.
- McNab, H. and Wynn, A. (2003), *Principles and Practice of Consumer Credit Risk Management*, 2 edn, Financial World Publishing, Canterbury, Kent.
- Meilă, M. (2005), Comparing clusterings: an axiomatic view, in 'Proceedings of the 22nd international conference on Machine learning', ACM, pp. 577–584.
- Meilă, M. (2007), 'Comparing clusterings an information based distance', *Journal of Multivariate Analysis* **98**(5), 873–895.
- Meyer, D. and Wien, F. T. (2014), Support vector machines, Technical report, R Foundation for Statistical Computing, Boston.
- Milligan, G. and Cooper, M. (1988), 'A study of standardization of variables in cluster analysis', *Journal of Classification* **5**(2), 181–204.
- Mishra, A., Rana, P. S., Mittal, A. and Jayaram, B. (2014), 'D2N: Distance to the native', *Biochimica et Biophysica Acta (BBA)-Proteins and Proteomics* **1844**(10), 1798–1807.
- Mouhab, A. (1995), 'Predictive inference: An introduction', *Journal of the American Statistical Association* **90**(432), 1489–1491.
- Nachev, A. and Hogan, M. (2013), Using data mining techniques to predict product quality from physicochemical data, in 'Proceedings on the International Conference on Artificial Intelligence (ICAI)', The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp), p. 1.
- Neter, J., Kutner, M. H., Nachtsheim, C. J. and Wasserman, W. (1996), *Applied Linear Statistical Models*, 4 edn, McGraw-Hill, USA.

- Neumann, J. v. (1928), ‘Zur theorie der gesellschaftsspiele’, *Mathematische Annalen* **100**(1), 295–320.
- Osborne, M. J. and Rubinstein, A. (1994), *A course in game theory*, MIT press.
- Parr-Rud, O. (2012), Drive your business with predictive analytics, Technical report, OLIVIAGroup.
- Pearl, J. (1984), *Heuristics: Intelligent search strategies for computer problem solving*, Addison-Wesley Pub. Co., Inc., Reading, MA.
- Pelleg, D. and Moore, A. W. (2000), X-means: Extending k-means with efficient estimation of the number of clusters., in ‘ICML’, pp. 727–734.
- Pena, J. M., Lozano, J. A. and Larranaga, P. (1999), ‘An empirical comparison of four initialization methods for the k-means algorithm’, *Pattern Recognition Letters* **20**(10), 1027–1040.
- Peralta, B., Espinace, P. and Soto, A. (2013), ‘Enhancing k-means using class labels’, *Intelligent Data Analysis* **17**(6), 1023–1039.
- PSPC (2015), ‘Protein structure prediction center’. Date accessed: 2016-06-05.
URL: <http://predictioncenter.org/>
- PWC (2015), The future shape of financial services in africa, Technical report, Price Waterhouse Coopers.
- Quinlan, J. (1993), *Programs for Machine Learning*, Morgan Kaufman, San Mateo, California.
- Quinlan, J. R. (1996), Bagging, boosting, and c4. 5, in ‘AAAI/IAAI, Vol. 1’, pp. 725–730.
- Quinlan, J. R. and Cameron-Jones, R. M. (1995), ‘Induction of logic programs: Foil and related systems’, *New Generation Computing* **13**(3-4), 287–312.
- Rana, P. S., Sharma, H., Bhattacharya, M. and Shukla, A. (2014), ‘Quality assessment of modeled protein structure using physicochemical properties’, *Journal of bioinformatics and computational biology* .
- Rand, W. M. (1971), ‘Objective criteria for the evaluation of clustering methods’, *Journal of the American Statistical association* **66**(336), 846–850.
- Redmond, S. J. and Heneghan, C. (2007), ‘A method for initialising the k-means clustering algorithm using kd-trees’, *Pattern Recognition Letters* **28**(8), 965–973.

- Riesen, K. and Bunke, H. (2008), Kernel k-means clustering applied to vector space embeddings of graphs, in ‘Artificial Neural Networks in Pattern Recognition’, Springer, pp. 24–35.
- Rissanen, J. (1983), ‘A universal prior for integers and estimation by minimum description length’, *The Annals of statistics* **11**, 416–431.
- Rodriguez-Galiano, V., Sanchez-Castillo, M., Chica-Olmo, M. and Chica-Rivas, M. (2015), ‘Machine learning predictive models for mineral prospectivity: An evaluation of neural networks, random forest, regression trees and support vector machines’, *Ore Geology Reviews* **71**, 804–818.
- Rohlf, F. J. and Fisher, D. R. (1968), ‘Tests for hierarchical structure in random data sets’, *Systematic Biology* **17**(4), 407–412.
- Rosenblatt, F. (1958), ‘The perceptron: a probabilistic model for information storage and organization in the brain.’, *Psychological review* **65**(6), 386.
- Ruppert, D., Wand, M. P. and Carroll, R. J. (2003), *Semiparametric regression*, number 12, Cambridge university press.
- Russell, S. J. and Norvig, P. (2003), *Artificial Intelligence: A Modern Approach*, 2 edn, Pearson Education.
- Safavian, S. R. and Landgrebe, D. (1991), ‘A survey of decision tree classifier methodology’, *IEEE transactions on systems, man, and cybernetics* **21**(3), 660–674.
- SAICA (2013), Guide on the retention of records, Guidelines, South African Institute for Chartered Accountants.
- Salzberg, S., Searls, D. and Kasif, S. (1998), ‘Grand challenges in computational biology’, *Computational methods in molecular biology* **32**, 3.
- Sands, T. M., Tayal, D., Morris, M. E. and Monteiro, S. T. (2015), Robust stock value prediction using support vector machines with particle swarm optimization, in ‘Evolutionary Computation (CEC), 2015 IEEE Congress on’, IEEE, pp. 3327–3331.
- Sarle, W. (1983), Cubic clustering criterion, SAS technical report a-108, SAS Institute Inc., Cary, NC.
- Schaffer, C. M. and Green, P. E. (1996), ‘An empirical comparison of variable standardization methods in cluster analysis’, *Multivariate Behavioral Research* **31**(2), 149–167.
- Schwarz, G. (1978), ‘Estimating the dimension of a model’, *The annals of statistics* **6**(2), 461–464.

- Sharma, D. (2013), 'Elements of optimal predictive modeling success in data science: An analysis of survey data for the 'give me some credit' competition hosted on kaggle', *Available at SSRN 2227333*.
- Shifa, N. and Rashid, M. (2003), 'Monte carlo evaluation of consistency and normality of dichotomous logistic and multinomial logistic regression models'.
- Siddiqi, N. (2006), *Credit Risk Scorecards*, John Wiley & Sons, New Jersey.
- Sinkkonen, J., Kaski, S. and Nikkilä, J. (2002), Discriminative clustering: Optimal contingency tables by learning metrics, *in* 'Machine Learning: ECML 2002', Springer, pp. 418–430.
- Sitar, M., Rašeta, J. and Klešček, A. (2014), Implementation of data mining techniques in credit scoring, *in* 'Proceedings of the xiv international symposium symorg 2014: new business models and sustainable competitiveness', FON, p. 130.
- Socher, R. and Fei-Fei, L. (2010), Connecting modalities: Semi-supervised segmentation and annotation of images using unaligned text corpora, *in* 'Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on', IEEE, pp. 966–973.
- Sokal, R. R. (1958), 'A statistical method for evaluating systematic relationships', *Univ Kans Sci Bull* **38**, 1409–1438.
- Sokal, R. R. and Sneath, P. H. (1963), 'Principles of numerical taxonomy.'
- Song, X. and Fan, G. (2002), A study of supervised, semi-supervised and unsupervised multiscale Bayesian image segmentation, *in* 'Circuits and Systems, 2002. MWSCAS-2002. The 2002 45th Midwest Symposium on', Vol. 2, IEEE, pp. II–371.
- Steinley, D. (2001), *Local Optima in k-means Clustering: What You Don't Know May Hurt You (or, is Ignorance Bliss?)*, University of Illinois at Urbana-Champaign.
- Steinley, D. (2004), Standardizing variables in k-means clustering, *in* D. Banks, F. McMorris, P. Arabie and W. Gaul, eds, 'Classification, Clustering, and Data Mining Applications', Studies in Classification, Data Analysis, and Knowledge Organisation, Springer Berlin Heidelberg, pp. 53–60.
- Stylianou, N., Akbarov, A., Kontopantelis, E., Buchan, I. and Dunn, K. W. (2015), 'Mortality risk prediction in burn injury: Comparison of logistic regression with machine learning approaches', *Burns* **41**(5), 925–934.
- Taft, L., Evans, R. S., Shyu, C., Egger, M., Chawla, N., Mitchell, J., Thornton, S. N., Bray, B. and Varner, M. (2009), 'Countering imbalanced datasets to improve adverse drug event predictive models in labor and delivery', *Journal of biomedical informatics* **42**(2), 356–364.

- Tao, L., Zhang, P., Qin, C., Chen, S., Zhang, C., Chen, Z., Zhu, F., Yang, S., Wei, Y. and Chen, Y. (2015), ‘Recent progresses in the exploration of machine learning methods as in-silico {ADME} prediction tools’, *Advanced Drug Delivery Reviews* **86**, 83–100. In silico {ADMET} predictions in pharmaceutical research.
- Taylor, J. (2014), Three steps to put predictive analytics to work, Technical report, Decision Management Solutions.
- Thomas, L. C. (2009), *Consumer Credit Models*, Oxford University Press, New York.
- Tishby, N., Pereira, F. C. and Bialek, W. (2000), ‘The information bottleneck method’, *arXiv preprint physics/0004057*.
- Transunion (2006), ‘Segmentation for credit-based delinquency models’, Website.
URL: <http://www.transunion.com/docs/business/serviceSolutions/VantageScore>
- Tu, J. V. (1996), ‘Advantages and disadvantages of using artificial neural networks versus logistic regression for predicting medical outcomes’, *Journal of Clinical Epidemiology* **49**, 1225–1231.
- Unger, R. and Moult, J. (1993), ‘Finding the lowest free energy conformation of a protein is an np-hard problem: Proof and implications’, *Bulletin of Mathematical Biology* **55**(6), 1183–1198.
- Vapnik, V. (1996), *The nature of statistical learning theory*, Springer Science & Business Media.
- Vapnik, V. and Lerner, A. (1963), ‘Generalized portrait method for pattern recognition’, *Automation and Remote Control* **24**(6), 774–780.
- Venter, J. and De la Rey, T. (2007), ‘Detecting outliers using weights in logistic regression’, *SA Statist J* **41**(2), 127–160.
- Verhulst, P. F. (1838), Notice sur la loi que la population suit dans son accroissement, in ‘Correspondance mathematique et Physique’, A Quetelet.
- Vinh, N. X., Epps, J. and Bailey, J. (2009), Information theoretic measures for clusterings comparison: is a correction for chance necessary?, in ‘Proceedings of the 26th Annual International Conference on Machine Learning’, ACM, pp. 1073–1080.
- Wagstaff, K. and Cardie, C. (2000), ‘Clustering with instance-level constraints’, *AAAI/I-AAI* **1097**.
- Wagstaff, K., Cardie, C., Rogers, S. and Schrödl, S. (2001), Constrained k-means clustering with background knowledge, in ‘ICML’, Vol. 1, pp. 577–584.

- Wang, A., Li, J., Wu, P. and Lu, Z. (2012), Semi-supervised segmentation based on level set, in 'Information and Business Intelligence', Springer, pp. 129–135.
- Wang, L. (2005), *Support Vector Machines: theory and applications*, Vol. 177, Springer Science & Business Media.
- Wang, X. and Pardalos, P. M. (2015), 'A survey of support vector machines with uncertainties', *Annals of Data Science* **1**(3-4), 293–309.
- Ward Jr, J. H. (1963), 'Hierarchical grouping to optimize an objective function', *Journal of the American statistical association* **58**(301), 236–244.
- Werbos, P. (1974), 'Beyond regression: New tools for prediction and analysis in the behavioral sciences'.
- Were, K., Bui, D. T., Øystein B. Dick and Singh, B. R. (2015), 'A comparative assessment of support vector regression, artificial neural networks, and random forests for predicting and mapping soil organic carbon stocks across an afro-montane landscape', *Ecological Indicators* **52**, 394 – 403.
- Wong, M. A. (1982), 'A hybrid clustering method for identifying high-density clusters', *Journal of the American Statistical Association* **77**, 841.
- Wong, M. A. and Lane, T. (1981), A kth nearest neighbour clustering procedure, in 'Computer Science and Statistics: Proceedings of the 13th Symposium on the Interface', Springer, pp. 308–311.
- Wu, F.-X. (2008), 'Genetic weighted k-means algorithm for clustering large-scale gene expression data', *BMC bioinformatics* **9**(Suppl 6), S12.
- Xing, E. P., Ng, A. Y., Jordan, M. I. and Russell, S. (2003), 'Distance metric learning with application to clustering with side-information', *Advances in neural information processing systems* **15**, 505–512.
- Yan, M. (2005), Methods of determining the number of clusters in a data set and a new clustering criterion, PhD thesis, Virginia Polytechnic Institute and State University.
- Yap, B. W., Rani, K. A., Rahman, H. A. A., Fong, S., Khairudin, Z. and Abdullah, N. N. (2014), An application of oversampling, undersampling, bagging and boosting in handling imbalanced datasets, in 'Proceedings of the First International Conference on Advanced Data and Information Engineering (DaEng-2013)', Springer, pp. 13–22.
- Ye, J., Zhao, Z. and Wu, M. (2008), Discriminative k-means for clustering, in 'Advances in neural information processing systems', pp. 1649–1656.

- Yeung, K. Y. and Ruzzo, W. L. (2000), An empirical study on principal component analysis for clustering gene expression data, Technical report, Technical report, Department of Computer Science and Engineering, University of Washington.
- Yoon, K.-A., Kwon, O.-S. and Bae, D.-H. (2007), An approach to outlier detection of software measurement data using the k-means clustering method, in ‘Empirical Software Engineering and Measurement, 2007. ESEM 2007. First International Symposium on’, IEEE, pp. 443–445.
- Youness, G. and Saporta, G. (2010), ‘Comparing partitions of two sets of units based on the same variables’, *Advances in data analysis and classification* **4**(1), 53–64.
- Zavrel, J. (1997), An empirical re-examination of weighted voting for k-nn, in ‘Proceedings of the 7th Belgian-Dutch Conference on Machine Learning’, Citeseer, pp. 139–148.
- Zeidat, N. M. and Eick, C. F. (2004), K-medoid-style clustering algorithms for supervised summary generation., in ‘IC-AI’, Citeseer, pp. 932–938.
- Zhou, L. and Wang, H. (2012), ‘Loan default prediction on large imbalanced data using random forests’, *TELKOMNIKA Indonesian Journal of Electrical Engineering* **10**(6), 1519–1525.