Dissertation for MSc in Statistics

Comparative study of Statistical Data modelling with Machine Learning Techniques

By
Phumzile Mgcima
42629446

Supervisor : Prof JO OLAOMI
Department Of Statistics
University of South Africa

28th January 2022

I, Phumzile Mgcima, hereby declare that this project is my own work and that all the sources that I have used or quoted have been indicated and acknowledged by means of complete references.

Abstract

Nowadays human activities produce massive amounts of data everyday. It is estimated that 2.5 quintillions bytes of data are produced daily. The ability to analyse and interpret such data, usually referred to as 'big data', is a precondition to succeed in the 4^{th} Industrial Revolution (4IR). Statistical data modelling has been a de facto data analysis paradigm for many decades, but it is slowly being overshadowed by machine learning algorithms in the industry and in research funding. In this research, the two modelling paradigms were compared with the aim of establishing which one is better in terms of rational, accuracy and model parsimony. Unlike many studies on this subject which mainly concentrate on comparing accuracy, this research did not look at accuracy as the only metric of comparison.

Both modelling paradigms were applied in prediction (continuous value prediction), classification (categorical class label prediction) and clustering problems in three separate case studies. In the prediction case study, a Realised GARCH (RealGARCH) model was compared to an artificial neural network (ANN) algorithm. In the classification case study, a linear discriminant analysis (LDA) model was compared to a support vector machine (SVM) algorithm. Lastly, a Gaussian mixture model (GMM) was compared to a K-means algorithm. For prediction and classification, the data was divided into training and testing sets, the training sets were used to fit the models and the testing sets were used to measure prediction and classification accuracy. For clustering, model validation was based on bootstrapping, visualisation and distant measures.

The ANN model outperformed the generalised autoregressive conditional heteroscedasticity (GARCH) variant RealGARCH model in the two accuracy measurements, root mean square error (RMSE) and mean absolute error (MAE), while RealGARCH gave more insights into the data. SVM had marginally better classification accuracy in both the two-class and the three-class scenarios but had poorer *F-Measure* for the minority classes in the three-class scenario. The statistical models were more interpretable compared to their machine learning counterparts in both case studies. Both clustering models performed poorly in partitioning the data in the third case study, but K-means did better than the GMM model. Understanding the domain problem was found to be essential to data analysis regardless of the modelling paradigm.

Acknowledgements

I want to express my sincere gratitude to Professor John Olaomi. Firstly, for allowing me to explore this complex topic and secondly, for his guidance and amazing patience throughout the research.

A word of thanks also goes to my family for their support and allowing me time and space to work on the research.

Contents

List of Figures

List of Tables

Acronyms

4IR 4^{th} Industrial Revolution i, 1, 2, 69

AC agglomerative clustering 13, 14

ACF autocorrelation function 26, 27, 28, 29

ADLs activities of daily living 1

AIC Akaike information criterion 26, 29, 30

ANN artificial neural network i, iii, iv, v, vii, 8, 9, 10, 15, 16, 20, 21, 22, 30, 31, 32, 33, 34, 68, 86, 87

ANOVA analysis of variance 70

AR autoregressive 17, 18, 28, 30

ARCH autoregressive conditional heteroscedasticity 17, 18, 19, 20, 28, 29, 30

ARDL autoregressive distributed lag 20

ARMA autoregressive moving average ix, 9, 10, 17, 28, 29, 30, 31, 32, 81

BIC Bayesian information criterion 11, 29, 30, 60, 61, 66

BP Backpropagation 9

CAPM capital asset pricing model 16, 17

CBOE VIX Chicago Board of Exchange Volatility Index 16

CEM classification expectation-maximisation 13

CI's confidence intervals vii, 62, 63, 71, 125

CNN convolutional neural networks 8, 9

DF discriminant function 35, 42, 46, 47, 50, 51

DNN deep neural network 10

EGARCH Exponential GARCH 10, 19, 29, 33

EM expectation-maximisation iv, 13, 14, 53, 54, 55, 56, 57, 59, 61, 70, 121

EMH efficient market hypothesis 16, 17, 25, 27, 28, 68, 70

ENN Elman neural network 9

ETF exchange traded fund 10

FA Forgy approach 13

FN False Negative 44, 95

FP False Positive 44, 95

GARCH generalised autoregressive conditional heteroscedasticity i, iii, v, 3, 8, 9, 10, 15, 16, 17, 18, 19, 20, 24, 25, 28, 29, 32, 33, 68, 87

GMM Gaussian mixture model i, iii, iv, vi, 3, 12, 13, 53, 55, 60, 61, 63, 64, 65, 66, 67, 68, 70, 121, 129

HDA heteroscedastic discriminant analysis 11, 34, 36, 49

HSM high speed machining 13

IoT Internet Of Things 1

JSE Johannesburg Stock Exchange 16

KA Kaufman approach 13

KHM K-harmonic means 13, 66

KNN k-nearest neighor 11

LC latent class 14

LDA linear discriminant analysis i, iii, v, viii, 3, 10, 11, 12, 14, 33, 34, 36, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 68, 94, 112

LR logistic regression 11

LRT likelihood ratio test 62

LSTM long short term memory 8, 9

MA Macqueen approach 13

MA moving average 17, 18, 28

MAE mean absolute error i, 9, 23, 24, 29, 30, 31

MAP maximum a-posteriori 14

MAPE mean absolute percentage error 9

MLE maximum likelihood estimation 7, 53, 62, 63

MLP multilayer perceptron 8, 9, 20, 30

MRI Magnetic resonance imaging 12

MSE mean square error 9, 10

NMSE normalised mean square error 9

OLS ordinary least squares 3

PACF partical autocorrelation function vii, 18, 27, 28, 29

PCA principal component analysis 10, 12, 41

RealGARCH Realised GARCH i, iii, 3, 17, 19, 20, 23, 29, 30, 31, 32, 33

RMSE root mean square error i, 23, 24, 29, 30, 31

RNN recurrent neural networks 8, 9, 20

SAVI South African Volatility Index 16

SCUT SMOTE and cluster-based undersampling technique 40, 45

SMOTE Synthetic minority oversampling technique 40

SVM support vector machine i, iii, v, viii, 3, 7, 9, 10, 11, 12, 33, 34, 42, 43, 44, 45, 48, 49, 50, 51, 52, 68, 112

SVR support vector regression 9

TN True Negative 95

TP True Positive 95

UA uncertainty-aware attention 8, 9

WDS weighted directional symmetry 9

Chapter 1

Introduction

This chapter introduces key developments which justify venturing into this research. Section 1.1 is the background to the research. It touches on big data developments as well as earlier debates about the role of statisticians in such big data environment. Sections 1.2, 1.3, and 1.4 deal with motivation, purpose and objectives of the research respectively. Section 1.6, discusses the limitations of the study, which set boundaries on the scope of the research. The last section of the chapter is the research layout.

1.1 Background of the study

It is becoming increasingly difficult to imagine any aspect of human life that is not driven by data. Nowadays, massive amounts of data are generated through activities of daily living (ADLs) such as personal social media profiles, shopping, banking transactions, medical records and many more. Marr (2018) estimates that 2.5 Quintilian bytes (2.5×10^{18}) of data is generated everyday. Internet Of Things (IoT) which is one of the key components of 4IR[1] promises to produce even larger data volumes. Wielki (2016) describes IoT's architecture as having various sensors for gathering data and transmitting it through various channels and gateways, and ultimately through to some sort of a processing system like an analytics cloud. The sensors are smart devices which are utilised in ADLs and will render individual's life interconnected. The interconnection is from fridges to wrist watches to televisions and other domestic devices, all connected to give life to the concept of a "smart home". Bouchard and Giroux (2015) tackle the challenge of big data from smart homes, they estimate billions of data to come from smart homes through the ADLs. They cite research on emotion recognition which is made possible by video cameras capturing individual's ADLs at home. One of the possible benefits of such an initiative is a potential of being able to monitor a person with a disease like Alzheimer's through gestures in a smart home. Wielki (2016) extends this interconnectedness beyond a smart home, he generalises it to smart products that could also help organisations in optimizing operations through automated monitoring. The monitoring could include monitoring of employees in such organisations. He also mentions augmented reality being used for the purposes of employee training. Whichever aspect of human life one looks at, the ADLs will be big data generating activities of the 4IR.

Ordinarily, this should be an exciting time for statisticians like it was for the software engineers back in the 1990s (Davenport and Patil, 2021). Unfortunately evidence shows a lack of such enthusiasm

[1]**Industry 4.0** is another variation of the term.

about the profession but rather the focus is on subjects which use tools created by statisticians. Davenport and Patil (2021) devoted time to highlighting data scientist as being the sexiest job of the century, in the same breadth stating that one needs statistics to do “data science”. Hand (2000) observed how statisticians glaze in awe at meetings and conferences organised by people other than themselves, discussing data analysis under different guises like neural networks, data mining and so on. Such meetings turn to pull away research funding which would have been awarded to statisticians. In the early 60s already, Tukey (1962) urged and encouraged statisticians to look at how other fields analysed data. He advocated for novelty on the way statistics was done. This research focussed on statistical data modelling and machine learning techniques, comparing the two paradigms with the aim of creating a tool-box for a statistician in the contemporary data analysis environment.

1.2 Motivation/Justification

Machine Learning is one of the key buzz words in the 4IR jargon, from techies, business strategists to big corporate executives. The meaning vary depending on which group one is engaging with. In this research, machine learning referred to algorithmic models that are used in data analysis. The usage of the term ‘data analysis’ in this text is consistent with the definition given by Tukey (1962), where data analysis is described as being more than just inference, but to also include procedures and techniques of gathering and analysing the data that led to such inference. Machine learning algorithms tackle the same problems that statistical modelling has been tackling for decades; prediction, classification, clustering etc. These have been standard activities for statisticians all over the world, so what is the buzz about? Why would the advent of such algorithms threaten funding to statistical research? Donoho (2017) relates a story of how the University of Michigan ploughed $100 million in data science programs with no participation of the statistics department although the curricula of such programs are strikingly similar to the statistics curriculum. The poor perception that statisticians have attracted in the recent past comes through in Lin and Li (2021). In distinguishing between statisticians and data scientists, one of the points they make is that statisticians are focused on modelling while data scientists are focused on results. Such view paints statisticians as less efficient and data scientist as results driven. This clearly is a misconception, and a serious indictment of the profession. Such sentiment clearly discounts the successes and advancements of Tukey and his colleagues at Bell Labs (Kettenring, 2012). Their contribution was across the board at the labs while they were making advances in statistical computing. These advances led to the development of S-language which is a predecessor to R, a popular language even in data science (machine learning). Tukey (1962) had expressed concerns about the appeal and attractiveness of statistics to bright students coming out of universities who would otherwise be drawn to other subjects like mathematics, physics and lucrative jobs at wall street, such list can now also include data science (which employs machine learning algorithms). In fact, Davenport and Patil (2021) compare the popularity of data scientist to that of “quants” in the 80s and 90s.

The motivation behind this study was not only due to the apparent threat faced by statistics as a subject but was also motivated by the fact that Tukey (1962) encouraged statisticians to look at how other fields analyse data without having their own theory distorted.

1.3 Purpose

The purpose of the study was to compare statistical data modelling with machine learning algorithms, the robustness of each was explored both theoretically and empirically. Statistical modelling is both inference and data modelling, Hand (2000) described the former as trying to address whether a structure exists in the mechanism which led to the data so that random chance can be distinguished from reality and described the latter as the overall representation of such data. Breiman (2001) distinguished between two cultures; stochastic data modelling culture and machine learning algorithmic culture. He seemed to have nailed his colours to the mast of machine learning algorithmic culture. On the contrary to Breiman, this research was not a toe to toe contest between the two cultures but sought to discover when and where does each approach have merit over the other.

1.4 Objectives

The main objective of the study was to compare machine learning algorithms with statistical data modelling.

There were three main problems which formed the basis of comparison:

- Prediction problem ; GARCH modelling versus artificial neural network.
- Classification problem; LDA versus SVM.
- Clustering problem; GMM modelling versus K-means'

For brevity, in this research the *prediction problem* refers to the problem of predicting a continuous value given predictor variables while the *classification problem* refers to the problem of predicting categorical class labels given some predictor variables. This is because from a technical point of view these are both prediction problems.

The GARCH volatility modelling is discussed in a broader sense of the word but the empirical implementation for comparison is a RealGARCH model.

1.5 Statement of the problem

Machine Learning algorithms purport to solve problems that statistics (statistical data modelling) have been solving and deem as standard activities for a long time. Megahan (2016) tries to illustrate the confusion that this overlap has brought among both statisticians and the so called data scientists [2].

For an example, take a regression prediction problem. In Machine Learning, regression model parameter estimation is done via an algorithm, the popular one being called gradient descent, while in statistical modelling this is usually done via ordinary least squares (OLS), with matrix algebra when it comes to multivariate setting.

Given selected models for comparison from both schools of thought, the research problem is to establish:

[2]Data science being the broader area encompassing machine learning algorithms and statistics in most cases.

- Empirically, to what an extent can a distinction be drawn between problems requiring statistical data modelling and those requiring machine learning algorithms.
- Robustness of theory underpinning the techniques.
- User friendlies and simplicity of the techniques (model parsimony).

1.6 Limitations of the research

Nongxa (2017) spoke of *data analytics work flow life cycle* which consists of data management and data analytics. Each of these two is broken in separate distinguishable steps; data management and analytics. The analytics phase consists of:

1. Data modelling and analysis.
2. Interpretation.
3. Decision-making

The focus of this research was mainly on data modelling and analysis phase of this life cycle. The assumption was that data gathering and cleaning activities are standardized on both sides and the analysis dataset is the same on both sides. Limiting the scope to data modelling and analysis was by no means implying that the other activities are of less importance but it was because these are subject to particular environmental set ups and cultures. For example, each organization/institution might have its own way of doing data annotation dictated by production software/hardware configuration and history. Once the baseline data has been established (through various means), the data modelling and analysis should be consistent.

The research did not present exhaustive study of either machine learning and statistical modelling disciplines holistically, but the models and algorithms selected for the case studies were explored with as much detail as possible. Machine learning has clearly contributed immensely to areas of computer science and artificial intelligence like in speech recognition in cellphones and other areas, those aspects were not studied in this research.

1.7 Layout of the research

This chapter served as an introduction to the whole research, it outlined the background, motivation, purpose, objectives as well as limitations of the research. Chapter 2 presents literature review of the research problem, the literature review is both theoretical and empirical in nature. Chapter 3 presents the research methodology and the case studies, each case study is a self-contained empirical study which compares the two modelling paradigms on a specific data modelling problem. Chapter 4 is the conclusion and recommendations of the whole research based on the results and conclusions of the individual case studies.

Chapter 2

Literature Review

The literature review in this research comprises of two main sections. Section 2.1 explores the theoretical debates about the practice of statistical modelling over the years, from Tukey (1962) to Breiman (2001) and others. Section 2.2 looks at empirical literature where statistical modelling techniques are compared to their machine learning counterparts.

2.1 The Two Schools of Thought

Chambers (1993) distinguished between two forms of statistics, greater statistics and lesser statistics. Lesser statistics in his view is the one that is confined in statistics departments in academia. It is isolated and has more of a mathematical orientation, a forte of those in universities' statistics departments. He saw this formality as a limitation in the practice of statistics. On the other hand, the greater statistics he described is wide ranging in methodology, collaborative and mostly with practitioners located outside statistics departments. This greater statistics includes preparing data, analysing the data (by models or other summaries) and presenting such data. He contended that the yearning to give statistics its own theoretical base has had the consequences of limiting its impact in society.

Breiman (2001) summed up data analysis in two cultures, namely, stochastic data modelling and algorithmic modelling. He claimed that statistical community had been exclusively committed to the former which had led to 'irrelevant theory, questionable conclusions' and had impeded them from working on stimulating contemporary problems. On the other hand, algorithmic modelling fast development in his time was due to the areas outside statistics, and had been applied to big and demanding data sets as a better substitute to data models. He challenged statisticians to desist from only using data models and begin to embrace the various new available tools in data driven problem solving. He argued that statisticians had not been able to enter some of the profitable new commercial and scientific fields due to incompatibility of data models in such fields.

Hand (2000), on the other hand, criticized statistics for putting too much emphasis on modelling and inference compared to description and exploration. While Breiman (2001) seemed to advocate for a wholesale acceptance of alternative techniques (algorithms), Hand (2000) was cautious and noted that these techniques meted out by these outside professionals may have a poor theoretical base even though they do the trick in commercial environments.

Mallows (2006) expressed how Breiman opposed 'overmathematization' of statistics, a point he reportedly stressed in the 2002 report which was titled *Statistics: Challenges and Opportunities for the 21st Century.* This report is supposed to have endorsed that the fundamental activity of statistics should be inward focussed into the subject itself rather than on other fields of application. Breiman felt that its recommendations were taking statistics back to the yesteryears of Hotelling[1] and others. Mallows (2006) recounted the contribution of Sir Ronald Aylmer Fisher [2] between 1922 and 1925, how he gave statistics a theoretical basis which led to its development as a mathematical subject. This theory gave statistics a mathematical character ready for doctoral studies in many topics with many theorems. This led to a decline in applications oriented studies, as the interest of the talented academics shifted towards mathematical oriented theory which had more publications in journals such as Annals of Mathematical Statistics. He cited Tukey at the Madison Conference of 1967 as having argued that "statisticians should aspire to be first rate scientists, rather than second-rate mathematicians". Tukey argued against statistics being taught as a sub field of mathematics, this was in direct contrast to some of his predecessors like Hotelling who successfully motivated for it to be connected to mathematics departments.

Breiman (2001) expressed discomfort about aspects of statistical data modelling like goodness of fit and normality assumptions. According to him data modelling culture assumes a priori stochastic data model and relies on goodness of fit and residual analysis to validate the model. Machine learning algorithmic culture, on the other hand, assumes the model is difficult and unidentifiable, instead it tries to find an algorithm that operates on the input variable to predict the response, and only considers predictive accuracy for model validation. He challenged both the usage of goodness of fit and residual analysis saying that goodness of fit procedures have little power when the direction of the alternative is not specifically defined, and that the residual analysis lacks power as dimensions become more than a few. He believed the persistence on data models has left statistician stuck with discriminant analysis and logistic regression for classification and multiple linear regression for prediction. According to this view, although a few really trust the assumption of a multivariate normal distribution when faced with multivariate data, this assumption still takes up a large part of lower graduate texts. In fact, he questioned any assumption of a prior parametric distribution in complex data. He estimates that 98% of statisticians fall into data modelling culture.

The criticism of statistical significance is highlighted in Johnson (1999). He reports about the American Psychological Association considering banning the use hypothesis testing at one stage, although the view was widely held, but it was rejected because it would have seemed like censorship. He also mentions that speakers at the 1998 annual conference of The Wildlife Society were all of the view that hypothesis testing was overused, misused and inappropriate. Breiman (2001) emphasised that point saying if all you have is a hammer, every problem looks like a nail. Wilk[3] was a contemporary of Tukey and his colleague at AT&T Bell Laboratories. He later became the president of the company and a Chief Statistician of Canada. Mallows (2006) quotes Wilk to having once remarked, "Significance tests are things to do while your are thinking about what you really want to do".

The Occum's Razor principle gives 'precedence to simplicity' (Duignan, 2018), the simpler the better. Breiman (2001) viewed algorithmic models as mysterious and complex but more accurate and data models as interpretable and simple but less accurate which introduces an Occum's Razor principle di-

[1]Horold Hotelling is known to have played a big role in giving statistics mathemathical character
[2]https://www.britannica.com/biography/Ronald-Aylmer-Fisher
[3]`https://en.wikipedia.org/wiki/Martin_Wilk`

lemma. He makes an example of how linear regression is interpretable compared to the neural network but is deemed less accurate. He did not see dimensionality as a curse, contrary to popular practice of variable deletion to reduce dimensionality; he argued one loses valuable prediction information by reducing dimensionality. According to Nongxa (2017) the curse of dimensionality in the context of traditional statistics occurs when the number of observations is lower than the number of attributes (or dimensions) which is deemed to result in false correlations between the attributes, such data forms short and fat matrices.

Cox (2001) argued that typical statistical model fitting often has to also respond to subject matter questions and causality rather than just predictive accuracy which is the main feature in the algorithmic modelling.

Efron (2001) pointed out that the concept of 'unbiasedness' using amongst others maximum likelihood estimation (MLE) in statistical theory has led statistics to be a central analysis procedure in many areas but came with a burden of requirements like 'good experimental design' which are not required at all in algorithmic modelling. He argued that the apparent success of the algorithm modelling could be centred on some form of biased estimation with little theory to support it. He raised the following points:

- **New Methods always look better than the old ones.**
  He made an example of logistic regression versus neural networks. Logistic regression might seem dull and often not much effort is applied in optimizing it due to excitement about the newly discovered neural network. But artificial neural networks suffer the same fate when compared to more recent SVM and so on.

- **Complicated methods are harder to criticize than the simpler ones.** One cannot confidently analyse a black box model like SVM as confidently as they would a data model like logistic regression. This limitation in analysis makes it difficult to also criticise. He further argued it is uniqueness of statistical modelling that any new methodology comes with its inferential basis completely clarified.

- **Prediction by itself is only occasionally sufficient.** As Cox (2001) argued, causality is often the main driver in many statistical modelling problems. For example, if a government wants to devise a response strategy to cancer, knowing how many people will get cancer at a point in time might help with planning treatment programs and facilities. But, knowing also the causes like smoking, diet and others might help with programs to combat the disease. As the adage goes, "Prevention is better than cure."

Huber (2006) remarked about the following definition of statistics:

"Statistics is the discipline concerned with the study of variability, with the study of uncertainty and with the study of decision making in the face of uncertainty."

He took an issue with this definition of statistics saying that it removes 'practice of data analysis', likening it to removing 'experimental physics' from the definition of physics. He agreed with Breiman (2001) that such definition is retrogression instead of progression into the future. He claimed the unwillingness of statisticians to do 'dirty stuff' has allowed other fields like computer sciences to rebrand tools invented by statisticians, market them under different posher guises like 'data mining'. He reckoned statistics community should take a blame for not emphasizing enough on concepts like

Simpson's paradox in statistics education. He contended that the danger intrinsic in the black box models, like ANN, does not really lie in overfitting as popularly believed in the statistics community, but in the interpretation of such black box models.

Remarking on the discussion by Mallows (2006) of whether statisticians should be occupied by theory (overmathematization) or application. Buja (2006) asserts that there should not be a conflict between applied areas in statistics and theorists just like in physics, experimentalists and theorists may occasionally tease each other but both contribute to physics being the most successful of sciences. Contrary to the view expressed by Mallows (2006), Buja (2006) embraces the definition of statistics which points to the study of "methodology divorced from application". His view is that statisticians are "tool makers, not carpenters", of course it does not hurt the tool maker to know how the carpenter uses the tool, but the abstraction from the subject matter allows the tool to be re-purposed in a different context even in future problems not yet defined. He cites an example of a classification problem. Classification model can be used to predict presence of cancer from the genes, and the same principles can be used to predict bankruptcy from credit record. He goes further to contest that it is not uncommon to develop theory (tools) which might not be seen in application, citing Bayesian modelling as having been theoretical for a long time until the power of computers allowed for its applications.

Efron (2006) remarked about statisticians self criticism. He acknowledged that the context in which Tukey (1962) wrote about statistics was that of a tired subject, which made Tukey's call to return to applications timely and exciting. In the same breath, he viewed statistics as having made strides in this area in the first 40 years after Tukey's paper, having been assisted by the advent of powerful computer processing. He welcomed the self-criticism by Tukey (and Breiman in the latter years) as a healthy practice but cautioned against being overzealous in self-criticism and thus mimicking a character in Sullivan and Gilbert (1885); "... who praises with enthusiastic tone, every century but this, and every country but his own.".

Chambers (1993) warned statisticians against being aloof, encouraging them to explore difficult sources of data which may lead to methods not available in traditional topics, and said if they don't act someone else will, and society will lose on mental qualities provided by statisticians.

2.2 Empirical Literature

2.2.1 GARCH and ANN Prior Comparative Studies

Artificial neural networks(ANNs) have been applied with success in many areas and are gaining traction in both econometrics and quantitative finance. On the other hand, since GARCH modelling was formulated by an economist, it is native to econometric and financial applications. This section touches on some of the prior research involving both ANN and GARCH models in predictive modelling.

A comparative study of four machine learning algorithms; multilayer perceptron (MLP), long short term memory (LSTM), convolutional neural networks (CNN), and uncertainty-aware attention (UA) was carried out by Gao et al. (2020). The UA had recurrent neural networks (RNN) as its basic structure. They applied these machine learning algorithms on historical prices of S&P500, CSI300 and Nikkei225 indices. The algorithms are all variants of ANN, for example, LSTM is a form of RNN and UA is a combination of different RNNs. LSTM enjoys success in image processing and text classification, CNN has successes in pattern recognition. The aim of the study was to compare price prediction accuracy of the four algorithms. MAE, mean absolute percentage error (MAPE) and correlation coefficient (R) were used as measures of prediction accuracy. UA was found to have outperformed the other algorithms with MLP being the lowest ranked in performance. A similar study was undertaken by Wu and Duan (2017) where different ANNs were compared in prediction of stock index price trend. Backpropagation (BP) neural network was compared to Elman neural network (ENN) in prediction of CSI300 price trend. BP neural network is an ANN trained by backpropagation and it is generally referred to as multilayer perceptron. ENN is a form of RNN and in the study momentum gradient descent was used for backpropagation instead of standard gradient descent. As measures of performance; mean square error (MSE), confirmation coefficient and the number of epochs needed by each algorithm to converge were used. A higher number of epochs indicate a slower learning rate. They found BP neural network to have outperformed ENN on the error measured by MSE, but the algorithms were on par with one another on confirmation coefficient. The ENN had a higher number of epochs which indicated it had a lower learning rate.

Charef and Ayachi (2016) used GARCH models and ANN algorithms in predicting the exchange rates of Tunisian dinar (TND). The exchange rates under the study were USD/TND, EUR/TND and JPY/TND. They found the ANN algorithms to be more robust compared to the GARCH models. They further recommended a more hybrid model of GARCH and ANN to improve on prediction accuracy. Hossain and Nasser (2011) studied a finite mixture of ARMA and GARCH (ARMA-GARCH) model, a BP neural network algorithm and a SVM. They had two objectives; firstly, to predict stock market index and exchange rate returns and secondly, to predict the direction of both the returns as well as the exchange rates. Direction refers to the indication of the price movement up or down without specifying the magnitude of the movement. The indices were S&P500 and Nikkei225, the exchange rates were GBP/USD and USD/JPY. The problem was modelled as a regression problem, using a variation of SVM called support vector regression (SVR) algorithm. The study focused on two categories of performance measures, deviation and direction. Deviation was measured using MSE, normalised mean square error (NMSE) and MAE. Direction was measured using directional symmetry (DS) and weighted directional symmetry (WDS). The SVR outperformed both BP and ARMA-GARCH in the deviation criteria. The ARMA-GARCH did better than the two algorithms in the direction criteria. The overall performance was checked by combining both direction and deviation metrics. BP was found to have performed better than the other two techniques. All the algorithms were observed to have performed better in the prediction of the exchange rates than in the prediction of the stock market indices.

Hossain et al. (2009) did a comparative study of three different modelling techniques; GARCH, ANN and SVM on four different market indices. The indices were Japan's NIKKEI 225, Hong Kong's Hang Seng (HS), UK FTSE 100[4] and Germany's DAX. They had a mixed bag of results with GARCH outperforming ANN on NIKKEI, Hang Seng and FTSE. ANN was the best in DAX while SVM was the best in all the other three markets. Laily et al. (2018) compared Elman Recurrent Neural Network (ERNN) to GARCH models using Bank Rakyat Indonesia (BRI) stock. They found that GARCH outperformed the ERNN. Yim (2002) studied the prediction performance ANN, ARMA-GARCH and a structural model on IBOVESPA stock index of the Sao Poalo Stock Exchange and found that both ANN and ARMA-GARCH outperformed the STS with the ANN outperforming ARMA-GARCH.

[4]Financial Times Stock Exchange index which measures top 100 companies listed on London Stock Exchange (`https://en.wikipedia.org/wiki/FTSE_100_Index`).

The study went further by using ARMA-GARCH volatility outputs as inputs into an ANN model. This hybrid model improved the results of the ANN and outperformed all the other models in the in the study. A new semi-parametric GARCH-model inspired by developments in ANNs is constructed in Donaldson and Kamstra (1997). The model adds semi-parametric non-linear terms to the GJR-GARCH[5] model to improve on the non-linear effects not captured in the GARCH model. The model was applied to various market indices and compared to the traditional GARCH, Exponential GARCH (EGARCH) and GJR-GARCH. The semi-parametric GARCH model was found to have outperformed the three traditional models.

Another hybrid approach is the one carried out in Zhong and Enke (2019), where ANNs are used together with principal component analysis (PCA). The study focussed on predicting the daily return direction of an exchange traded fund (ETF) called SPDR S&P 500 ETF. The data had 60 features and PCA was used to transform the data into 31 principal components. The daily return direction is either up or down, the problem was modelled as a classification problem. ANNs with varying number of hidden layers were applied to both untransformed data with 60 features and transformed data with 31 principal components. The hidden layers were varied between 10 and 1000 hidden layers. ANN with more 10 hidden layers is considered to be a deep neural network (DNN). MSE, confusion classification matrix and several statistical significance tests were used for results validation. The best classification performance was found to be that of the ANN with 10 hidden layers on the transformed data of 31 principal components. DNN with 20 hidden layers was the second best, just like in the 10 hidden layers case, it had the best performance on the transformed data with 31 principal components.

2.2.2 LDA and SVM Prior Comparative Studies

Obi (2017) used a variety of datasets in exploring situations which are best suited for each of the two classifiers. He had overlapping datasets, datasets with outliers, and non-linearly separable datasets. The datasets were also transformed into high dimensions to monitor the effects on the classifiers. The first noted difference was in the distance measures the two classifiers are based on. LDA is based on Mahalanobis distance while SVM is based on Euclidean distance. The study highlighted that the two distance measures imply that the class mean separation in LDA is a decision boundary passing through the midpoint between the two class means, while SVM depends on a separating hyperplane passing midway through the support vectors. He pointed out that SVM classification is impacted by data points (support vectors) nearest to the hyperplane and therefore less sensitive to outliers while LDA is susceptible to outliers. But, he also added that if the outliers are on the correct side of the separating hyperplane, LDA may still be preferred over SVM. He concluded that the outliers affected the performances of both classifiers to a varying extent, the impact is a bit more on LDA. On high dimensional data ($p > n$), where p is the number of predictor variables and n is the number of observations, the study recommended SVM over LDA. This is because ($p > n$) introduces multicollinearity which thwarts the performance of LDA. On class overlaps, the study found that because LDA is invariant under all non-singular linear transformation, it has an advantage over SVM in cases where dataset transformation leads to class overlaps. But, in cases where datasets are linearly separable after transformation, both classifiers were found to be equally effective. LDA was also preferred in cases where datasets have known parameters, different class means and common covariance matrices. In case of linearly inseparable low dimensional data, transforming the data into

[5]Glosten, Jagannathan and Runkle's Sign-GARCH which is also meant to take into account asymmetry of returns

high dimensional data and applying SVM (with kernel function) for classification was recommended.

Santos et al. (2014) compared LDA, SVM and logistic regression (LR) in sex determination based on craniometric[6] data. Their goal was to discuss the variations that relate to the statistical handling of such data. They used craniometric samples from four regions; France, Portugal, USA and Thailand. The Portuguese sample had the largest volume and was used as a training set for the experiment. The other three samples were used as testing sets. The data had sixteen craniometric predictor variables. Variable selection was performed using Bayesian information criterion (BIC) for LR, retaining variables which minimised the BIC. For LDA, variables whose coefficients had minor impact on the score function were removed. For SVM, recursive feature elimination (RFE) which ranks variables according to decreasing discriminant power was performed. The SVM-RFE leaves the decision to the user as to which variables to discard. Seven predictor variables were used for all the regions in the first case and fourteen variables for the three regions which excluded USA for the second case. The two cases consisted of various scenarios like different posterior probability thresholds, and balanced and unbalanced ratios of male and females in the data. The classification accuracy percentage as well as the percentage of indeterminate cases were measured. They found LR consistently outperformed LDA and SVM with highest classification accuracies as well as lowest indeterminate percentages. The other two techniques interchangeably outperformed each other in different scenarios.

Lesniak et al. (2012) evaluated SVM, neural networks (NNet), k-nearest neighbor (k-NN), random forests (RF) and LDA. The aim of the research was to compare these different techniques in reducing the false positive (FP) phenomenon in computer aided detection (CADe) of breast masses in mammography. They found LDA to have been the best model in terms of model parsimony (least complex) because all of the parameters could be estimated from the data. SVM and the other models required other parameters which could only be estimated via time consuming configurations outside the data. The overall finding of the study was that SVM had the best CADe based malignant detection rate with the lowest FP rates.

Shao et al. (2015) compared LDA, k-nearest neighor (KNN) and SVM amongst several models. The aim of the research was the characterisation of sesame and soybean oils into classes; hot-pressed, cold-pressed, and refined. They found KNN outperformed all the other models, closely followed by LDA. They also found that SVM performance fluctuated heavily depending on the choice of a kernel. A radial kernel gave 25% accuracy while a linear kernel gave 95.1% (To put these numbers into perspective, KNN gave 96.3% accuracy while LDA gave 96.2%).

Madhanagopal et al. (2012) pitted classification accuracy of LDA against that of SVM, by analysing their performances on the Indian Nifty index. In that study, different kernels were applied for the SVM. SVM showed superior classification accuracy which ranged from 97.32% to 100% while LDA exhibited accuracy ranges of between 87.29% to 93.75%.

Stapor (2016) evaluated SVM and the discriminant analysis variant, heteroscedastic discriminant analysis (HDA), on German credit scoring data. HDA takes into account heteroscedasticity of the within class variance and is known to perform better than vanilla LDA in situations where homoscedasticity assumption is not satisfiable. Their observation was that the SVM model with non-linear kernel did not give significant improvement over discriminant analysis, although the nonlinear kernel made the

[6]Craniometry is measurement of the cranium (the main part of the skull), usually the human cranium (`https://en.wikipedia.org/wiki/Craniometry`).

learning process to be more complex by requiring a separate validation procedure. They concluded that SVM models performed slightly better than discriminant analysis.

Suhandy and Yulia (2018) compared LDA and SVM in classification of Luwak coffee. The aim of the study was to separate real Luwak coffee from counterfeit Luwak coffee which would then help in combating Luwak coffee adulteration. They used an ultraviolet-visible spectroscopy (UV-Vis) spectrometer to extract spectral data from aqueous samples of Luwak coffee and counterfeit Luwak coffee. Spectral data is measured in wavelengths of radiation, the wavelengths ranged from of 200 nm to 400 nm. Seven wavelengths were used as predictor variables. They found both LDA and SVM achieved 100% classification accuracy for the coffee samples. In conclusion, they preferred LDA over SVM due to it being simpler and less computationally expensive.

Vinay et al. (2015) combined LDA and SVM in a face recognition experiment. They used Cambridge ORL face database which contains 400 grey scale images of 40 people. Each person had ten images with varying facial expressions and details; like smiles, closing eyes, wearing glasses etc. The dataset had 150 features. In their proposed model, LDA was used as a feature reduction technique and SVM as a classification algorithm. The number of features were reduced from 150 to 90 features. Training sample sizes of four, five and six were repeatedly randomly selected from each person, each time the remaining images served as testing sets. The recognition rate is a number of recognised images over the total number of images in the testing set. This rate served as a metric for measure of success. The recognition rate of the proposed model was compared to the rates of other LDA based models which were previously applied on the database. The proposed model was found to have outperformed the other models. A similar face recognition experiment was carried out by Mazanec et al. (2008). They compared LDA, PCA and SVM on FERET database using **csuFaceIdEval** and **libsvm** software libraries. They also found that a combination of SVM and LDA outperformed individual LDA, PCA, and SVM.

Nikitidis et al. (2014) proposed an algorithm which jointly performs dimensionality reduction and classification (simultaneously). The algorithm made use of LDA for dimensionality reduction on SVM support vectors. They applied the algorithm on three problems; facial expression recognition using the CohnKanade database, face recognition using the XM2VTS database and object recognition using the ETH80 image dataset. All these were formulated as joint optimisation problems. They compared the performance of the proposed algorithm to the performance of SVM with no dimensionality reduction and to the cases where dimensionality reduction is performed. Dimensionality reduction was performed as a preprocessing step using LDA, PCA, Subclass Discriminant Analysis (SDA), Locality Preserving Projections (LPP) and Orthogonal Locality Preserving Projections (OLPP). They measured the classification accuracy as a measure of performance. The proposed joint algorithm outperformed other procedures in the three case studies. The SVM with LDA preprocessing performed better than SVM with no dimensionality reduction.

2.2.3 K-means and GMM Prior Comparative Studies

Magnetic resonance imaging (MRI) is used for brain examination without invasive operation on the brain itself. This is done through brain scans which produce various images for analysis. Baid et al. (2016) used Fuzzy C-means, K-means and GMM in the MRI image segmentation analysis. In the experiment, they had the actual images and synthetic images. The synthetic images were produced using **TumorSim** software. The images contained information about presence and absence of tumours at varying degrees. They used various coefficients as model evaluation; Dice coefficient, Jaccard coefficient, sensitivity and specificity. They found Fuzzy C-means to have outperformed both K-means and GMM, with K-means outperforming GMM.

Wang et al. (2018) did a comparison of K-means and EM algorithm on a GMM clustering in high speed machining (HSM). HSM is used in manufacturing aircraft components and turbine blades to name a few. The aim of the experiment was to observe which one of the two techniques would partition breakages and collisions at different states of machine-tool; machine-tool at rest, machine-tool moving at a constant speed and machine-tool at varying speed. These were subsequently the three clusters used as input in the K-means. There was pre-labelling of the data in the experiment, and a criteria was chosen in the form of thresholds as to which cluster an observation should be assigned to, and error measurements were taken in case of deviations. The experiment was more of a classification problem than a pure clustering experiment. The EM was found to have outperformed K-means. This was attributed to the clusters not being spherical as expected by the K-means algorithm. K-harmonic means (KHM) is another variant of the K-means algorithm. It uses harmonic mean vector as a cluster centre unlike in the ordinary K-means where the mean vector comprises of ordinary averages.

Zhang et al. (1999) did a performance comparison of KHM, K-means and EM. Both K-means and EM algorithms are known to be sensitive to initialisation with EM affected to a lesser extent. The experiment involved observing performance in terms of the number of iterations until convergence for each algorithm. The experiment was repeated a few times with cluster centres varied at each initialisation step, from good initialisation to worst possible initialisation. The KHM algorithm converged in all cases with fewer iterations, followed by the EM algorithm. The KHM algorithm was therefore found to be insensitive to initialisation of the centres while both K-means and EM algorithm were affected by initialisation but EM algorithm not as badly affected like the K-means clustering. The K-means initialisation problem is an area of special interest in the machine learning community.

Peña et al. (1999) focussed on four different initialisation methods for K-means. The methods were Random, Forgy approach (FA), Macqueen approach (MA) and Kaufman approach (KA). In the Random method, the data is initially divided into K clusters at random. FA method randomly chooses K data points as centroids, and for the rest of the data, each data point is assigned to the nearest centroid. MA is similar to FA, the difference is that in FA assignment is carried out in batch while in FA the assignments are incremental. In KA the data points are successively chosen until K centroids are found. The first centroid is chosen as the most central data point in the data. For the rest of the centroids, data points which have a high probability of being surrounded by many data points are chosen as centroids. The research looked at initialisation as well as the time to convergence. The Random and KA methods were found to have partitioned the data more effectively than the other methods, but KA was also quicker to converge than the Random method. Therefore, KA was found to be the best initialisation method for K-means compared to the other three methods. The initialisation problem also gets a special attention in model based clustering.

Meilă and Heckerman (2001) focussed on initialisation while comparing three clustering techniques; EM, classification expectation-maximisation (CEM) and agglomerative clustering (AC). EM and CEM are model based while AC is heuristic. They used the Random approach (not the same as one used in K-means above), the Marginal approach and the AC itself as an initialisation method. The Random approach initialises parameters of the model independent of the data. The parameters were sampled from an 'uninformative' distribution, uniform (Dirichlet) distribution was used in this case. The

Marginal approach is a noisy-marginal of Thiesson et al. (1999), it is a data dependent approach which first determines a maximum likelihood configuration by assuming there is only one class. For AC initialisation, agglomerative clustering was performed on a random sample of the data and sufficient statistics were extracted from the resulting clusters. The model parameters were then set to be maximum a-posteriori (MAP) on the extracted statistics with a uniform prior distribution. The three initialisation methods were applied on the three clustering techniques using two datasets. One dataset was handwritten digits from US Postal Service Office for Advance Technology and the other one was a synthetic dataset using MSNBC news service stories data. The EM outperformed the other two clustering techniques for all the initialisation methods. The best results came from the Marginal approach which efficiently partitioned the datasets and consistently converged with fewer iterations.

Magidson and Vermunt (2002) compared latent class (LC) analysis to K-means. LC analysis is a finite mixture model (FMM) where the within class variables are assumed to have diagonal covariance matrices and therefore linearly uncorrelated. In the study, random samples were drawn from two populations of bivariate normal distribution. The bivariate normal parameters were specified as follows : $\mu_1 = (3,4), \mu_2 = (7,1), \sigma_1 = \sigma_2 = (2,1)$ and $\rho_1 = \rho_2 = 0$. The sample sizes were $N_1 = 200$ and $N_2 = 100$. LDA was applied for binary classification and it achieved $\frac{199}{200}$ accuracy for the first population and $\frac{97}{100}$ accuracy for the second population. The LDA results were used as benchmark to measure the performance of the two clustering techniques. The LC matched the performance of LDA by having 1.3% misclassification rate while K-means had 5% misclassification rate. The LC model therefore outperformed K-means model.

Chapter 3

Case Studies Analysis

This chapter covers the three case studies which formed the basis of comparison for statistical modelling and machine learning in this research. Section 3.1 describes the research methodology, and Sections 3.2, 3.3 and 3.4 are the actual case studies.

3.1 Research Methodology

The research comprised of three case studies; the first case study is a prediction problem, the second case study is a classification problem and the third case is a clustering problem. In all of the three case studies, the problem was approached using both a statistical data modelling solution and machine learning solution. In the prediction and classification problems, the hold-out validation method was used. The data was divided into training and test sets (hold-out sample). The training set was used to fit the model and the test set was used to validate the model's prediction accuracy. This was possible because in these two problems the data is labelled with true values up front. The validation involved comparing the true value to the predicted value. On the other hand, in clustering the data was not labelled, the objective was to discover the labels. Nonparametric bootstrap, distance measures and visualisation were used to validate the models. The following sections go into details of each of the case studies.

3.2 GARCH versus Artificial Neural Networks Case Study

3.2.1 Introduction

This part of the research dealt with predictive modelling by exploring volatility prediction using ANN and GARCH models. GARCH models have been used in econometrics for volatility modelling since the early 1980s. GARCH models are statistical models applied in economic and financial data. In the context of this research, the main differentiating factor between machine learning and statistical data modelling was the model adequacy checks in the form of significance testing, a practice unique to statistical data modelling. The two modelling paradigms are often applied to solve the same problems, for example, both do fit a linear regression model for prediction. The main validation of such a regression model in machine learning is prediction accuracy. In statistical modelling on the other hand, prediction accuracy is only entertained after model adequacy has been established. Such model adequacy is established through significance testing. Statistical modelling has faced criticism of being over elaborate on mathematics which results in its inability to solve practical contemporary data problems which are currently solved by machine learning algorithms. The main objective of this case study was to compare the popular machine learning technique ANN with a GARCH model in predicting volatility of South African financial markets indices.

3.2.2 Volatility Modelling Theoretical Framework

3.2.2.1 Characteristics of Volatility

Brailsford et al. (1993) attributes the increased scrutiny on volatility to, amongst other things, the stock market crash of 1987 and institutional changes. Volatility forecasting also plays a major role in activities such as portfolio selection, value at risk calculations in risk management and asset pricing especially option pricing. Tsay (2010) defines volatility in the context of option trading as the conditional standard deviation of the underlying asset return. Volatility is now itself a financial instrument tradeable in the form of various indices like the Chicago Board of Exchange Volatility Index (CBOE VIX), South African Volatility Index (SAVI) from the Johannesburg Stock Exchange (JSE) and various other exchanges all over the world. Figlewski (1997) touched on how volatility is perceived differently by different segments of society. To ordinary people, high volatility is something that needs to be avoided at all costs because it indicates a deficiency in the market, while to the market participants it is something that needs to be understood, managed and precisely forecast in order to get protection or take advantage of the market opportunities.

A major feature of volatility is that it is unobservable and therefore it can only be estimated. A lot of research is focussed on the accuracy of such estimation. In predictive modelling, one way of evaluating model accuracy is comparing the observed values to the model's predicted values. But, since volatility is unobservable, such comparison cannot be carried out directly, a form of volatility proxy is often used. Volatility of asset returns instead of the observed asset prices are often analysed. This is because the returns are easier to manage because they are scale-free and have desirable statistical features (Tsay, 2010). One of those features is that asset returns with zero autocorrelation constitute a white noise and this is a requirement for other forms of capital asset pricing model (CAPM)[1]. Such CAPMs would require testing for absence of autocorrelation to ensure efficient market hypothesis (EMH)[2] is obeyed (Tsay, 2010). An asset return at time t, r_t, is given by

$$r_t = ln\frac{P_t}{P_{t-1}} \text{ (a log return).} \tag{3.1}$$

where P_t is an asset price at time t. Andersen and Bollerslev (1998) argue that although volatility is unobservable, if a model for σ^2 is properly stated, then a squared return is an unbiased estimator of the latent volatility factor but they quickly add that squared returns yield noisy measurements due to idiosyncratic error. This error leads to large observation of variation with respect to σ^2 which makes the fraction of variation due to volatility quite low. They suggest the use of high frequency (usually between 5-min to 10-min returns) volatility proxies to lessen the noise and further suggest that these high frequency proxies perform better when compared to daily frequency measurements. The high

[1]CAPM describes the relationship between systematic risk and expected return for assets, particularly stocks

[2]EMH is a hypothesis in financial economics that states that asset prices fully reflect all available information

frequency returns enable a more accurate calculation of the *realised volatility* of r_t which is given as

$$RV_t = \sum_{i=1}^{n} r_{t,i}^2. \tag{3.2}$$

Tsay (2010) states that RV forms an independent and identically distributed (*iid*) sequence with finite variance. And, the log return $ln(R_t)$ follows a Gaussian autoregressive (AR) integrated moving average (MA) model, specifically an ARIMA(0,1,q) model which can be used to make predictions. Forsberg and Bollerslev (2002) presents a slightly different flavour of RV_t where the equation is adjusted for linearly interpolated mid-points of the bid-ask prices

$$RV_t = \sum_{i=1}^{288} r_{(288)}^2 (t + (i/288)), \tag{3.3}$$

$n = 288$ because of the frequency interval of 5 min in a day. It can be tempting to use even smaller interval which would increase the value of n for better accuracy but Tsay (2010) warns against that, stating that the smaller intervals are sensitive to market micro-structure like bid-ask bounce which normally lead to biased volatility estimates. Optimal interval is an area of ongoing research. Andersen and Bollerslev (1998) brought to attention a problem with the use of realised volatility forecast, which is that realised volatility is not in line with rational financial decision making which is motivated by expected future volatility rather than realised returns. Implied volatilities [3] are available in the market in various volatility indices of stock exchanges and are also often used as volatility proxies.

Apart from the lack of observability, Tsay (2010) discusses other features of volatility which are observable through examining asset returns, namely, *leverage effect*, *volatility clustering* and *stationarity.* Leverage effect refers to the fact that volatility treats big price increases differently to big price drops. Volatility cluster refers to the fact that volatility can be low for some periods and high for other periods. Stationarity refers to the characteristic that volatility does not diverge to the infinity but rather varies within some fixed range. Due to CAPM and EMH, stationarity gets a special attention in volatility modelling. Tsay (2010) distinguishes between two forms of stationarity, namely, strict stationarity and weak stationarity. Strict stationarity prescribes that the joint distribution of $(r_{t_1} \cdots r_{t_k})$ be identical to that of $(r_{t_1+t} \cdots r_{t_k+t})$, for all t with k a positive integer, while weak stationarity only requires that the mean and variance of r_t and r_{t-l} be time invariant for an arbitrary integer l. Strict stationarity is difficult to achieve in practical applications, the weaker form is usually assumed.

3.2.2.2 Volatility Model (RealGARCH Model)

The GARCH model belongs to the class of conditional heteroscedastic models. These autoregressive heteroscedastic models are best suited when the return series shows time-varying volatility and volatility clustering. The precursor to GARCH is autoregressive conditional heteroscedasticity (ARCH) and was formulated by Engle (1982). Before an ARCH model can be entertained, a residuals model is required to determine the presence of ARCH Effect, often an autoregressive model like an ARMA model is fitted. An ARMA(p,q) model is given as

[3]Implied volatility is calculated by taking the market price of the option, entering it into the Black Scholes formula, and back-solving for the value of the volatility

$$r_t = \phi_0 + \sum_{i=1}^{p} \phi_i r_{t-i} + a_t - \sum_{i=1}^{q} \theta_i a_{t-i} \tag{3.4}$$

a_t is called the shock or innovation of r_t, ϕ_i are parameters of the AR(p) component, θ_i are parameters for the MA(q) component. Equation (3.4) is called conditional mean equation, and sometimes the mean equation is just a constant. Testing for ARCH effect is a two step process. First, the $\{a_t\}$ series is tested for the absence of serial correlation, if the absence of serial correlation is established, then the $\{a_t^2\}$ series is tested for the ARCH effect. The ARCH effect is tested using either Ljung−Box test or Lagrange multiplier test.

An ARCH(m) model is represented as

$$a_t = \sigma_t \epsilon_t, \quad \sigma^2 = \alpha_0 + \alpha_1 a_{t-1}^2 + \cdots + \alpha_m a_{t-m}^2 \tag{3.5}$$

where $\{\epsilon_t\}$ is a sequence of independently and identically distributed (iid) random variables with mean zero and variance 1, $\alpha_0 > 0$, and $\alpha_i \geq 0$ for $i > 0$. PACF is used to determine the ARCH order m of the ARCH model. The ARCH model is criticised for requiring many parameters in order to sufficiently describe volatility, meaning it requires a large value of order m.

Tsay (2010) notes the following other weaknesses of the ARCH model:

1. The model assumes that positive and negative shocks have the same effects on volatility because it depends on the square of the previous shocks. In practice, it is well known that the price of a financial asset responds differently to positive and negative shocks.
2. The ARCH model is rather restrictive. For instance, α_2 of an ARCH(1) model must be in the interval $[0, \frac{1}{13}]$if the series has a finite fourth moment. The constraint becomes complicated for higher order ARCH models. In practice, it limits the ability of ARCH models with Gaussian innovations to capture excess kurtosis.
3. The ARCH model does not provide any new insight for understanding the source of variations of a financial time series. It merely provides a mechanical way to describe the behaviour of the conditional variance. It gives no indication about what causes such behaviour to occur.
4. ARCH models are likely to over-predict the volatility because they respond slowly to large isolated shocks to the return series (p.119).

On point 1) above, Franses and Dijk (1996) add that stock market breakdowns happen quicker than stock market booms, put simpler, ‘bad’ news increase volatility more than ‘good’ news. Also, Donaldson and Kamstra (1997) in their study of S&P 500, NIKKEI and FTSE found the asymmetry parameter to be significantly positive and thereby remarked that the negative return innovations do indeed lead to more volatility than positive return innovations.

In 1986 Bollerslev proposed an extension of the ARCH model, the Generalized ARCH model (Tsay, 2010). A Generalized ARCH(GARCH) model is defined as follows: Let $a_t = r_t - \mu_t$, a_t follows a GARCH(m,s) if

$$a_t = \sigma_t \epsilon_t, \quad \sigma^2 = \alpha_0 + \sum_{i=1}^{m} \alpha_i a_{t-i}^2 + \sum_{j=1}^{s} \beta_j \sigma_{t-j}^2 \tag{3.6}$$

where ϵ_t is a sequence of independently and identically distributed random variables with mean 0 and variance 1, $\alpha_0 > 0$, $\alpha_i \geq 0$, $\beta_j \geq 0$, and $\sum_{i=1}^{max(m,s)}(\alpha_i + \beta_i) < 1$ (Tsay, 2010). Unfortunately GARCH is still a symmetric model, and therefore that does not resolve some of the weaknesses of the ARCH model such as the leverage effect (asymmetry). On top of that, according to Arnerić et al. (2014), standard GARCH(1,1) model shows high persistence in conditional variance (which means the variance never diminishes). This leads to the model suffering from a significant upward bias in the persistence parameters. Several variations to the GARCH model have been proposed like EGARCH and Threshold GARCH (TGARCH) to handle leverage effects (asymmetry), Integrated GARCH (IGARCH) to handle unit root in $\{r_t\}$ series and GARCH in the mean (GARCH-M) to handle the case where the security return depends on its volatility. Models like Multiplicative ARCH, Piecewise-Nonlinear ARCH, Flexible Fourier Forms, Hamilton-style regime switching models and others have been studied, while some of these show slight improvement to the standard GARCH, none of these surpasses standard GARCH all the time and in a big way (Donaldson and Kamstra, 1997). The question of complexity versus simplicity in volatility forecasting often comes to the fore, with Brailsford et al. (1993) citing studies where simpler exponential moving averages outperform more complex GARCH models and others where the opposite is true.

A recent variation of the GARCH model is the RealGARCH formulated by Hansen et al. (2012). Real GARCH attempts to capture dynamic properties of returns and realized measure (Realised Volatility). RealGARCH connects observed realised volatility measure to the latent volatility via measurement equation. It also incorporates asymmetric reaction to shocks and therefore addresses leverage effect problem (Ghalanos, 2019). Ghalanos (2019) presents a slightly different but equivalent notation of RealGARCH to the original notation presented by Hansen et al. (2012). The former's notation is consistent with the notation by Tsay (2010) used throughout this research

$$\begin{aligned}
a_t &= \mu_t + \sigma_t\epsilon_t, \epsilon_t \sim iid(0,\, 1) \\
log\, \sigma^2 &= \alpha_0 + \sum_{i=1}^{q} \alpha_i log\, x_{t-i} + \sum_{i=1}^{q} \beta_i \sigma_{t-i}^2 \\
log\, x_t &= \xi + \delta log\, \sigma^2 + \tau(\epsilon_t) + u_t, u_t \sim \mathcal{N}(0,\, \lambda) \qquad (3.7)
\end{aligned}$$

where $log\, x_t$ is the realized measure and Equation 3.7 is called the measurement equation. RealGARCH is rendered asymmetric to shocks via $\tau(\epsilon_t)$ which is based on Hermite polynomials and has the quadratic form

$$\tau(\epsilon_t) = \tau_1\epsilon_t + \tau_2(\epsilon_t^2 - 1). \qquad (3.8)$$

This function is called leverage effect function because it captures the dependence between the returns and future volatility by inducing an EGARCH type structure in the GARCH equation (Hansen et al., 2012). The RealGARCH model is inspired by the availability of high frequency data, and the realised measure proxy in the model is the RV_t which was explored in the previous section.

There are many more variations to the GARCH model than the ones cited in the preceding paragraph. Hansen et al. (2003) compares 55 different volatility models with 54 being some form of a GARCH variant, 14 unique GARCH variants. In assessing whether any other GARCH model beat GARCH(1,1), Hansen and Lunde (2005) compares 330 GARCH models using 16 different variants. Each of these variants try to address a particular problem (or problems) of the GARCH (or other GARCH variant) model and is built around the idea of GARCH effects. Therefore understanding basic ideas around

ARCH gives one coverage of many volatility modelling problems because the knowledge is portable between the variants. Another aspect that gives GARCH modelling advantage over other volatility modelling techniques is that there is vast literature covering it, an advantage that is aknowledged in both Bauwens et al. (2006) and Asai et al. (2006). This vast literature also alludes to high level of peer review of the theory and applications of GARCH models. GARCH models usually form basis of comparison for many newer volatility models, like in Asai et al. (2006) the multivariate stochastic volatility were generally benchmarked against multivariate GARCH in terms of parsimonious model specification, model diagnostics and other aspects. Ghouse et al. (2019) compared autoregressive distributed lag (ARDL) to GARCH in equity markets. They found that ARDL and GARCH had the same power in identifying relationships in the time series data. But, ARDL failed to capture ARCH effect in the daily and weekly data. They also found that ARDL was as good as GARCH in monthly data, concluding that it can be used as an alternative to GARCH in low frequency data. In this research GARCH (RealGARCH in particular) modelling was chosen as an alternative volatility modelling technique to ANN based on its maturity, vast literature and as well as model parsimony.

3.2.2.3 Artificial Neural Networks

The inspiration behind artificial neural networks (ANNs) is the human brain. In ANNs, regression modelling is performed by imitating the brain's computation process. ANNs are nonlinear and assume no underlying distribution and this is deemed as an advantage over their parametric counterparts (Brooks, 1998). At a basic level, an ANN can consist of a single node of input with a single output, but it is usually multiple layers of multiple nodes filtering information into an ultimate layer of output. In keeping up with the brain terminology, the nodes are commonly referred to as neurons, the neurons are joined together by synaptic weights. A synaptic weight indicates the amount of influence each neuron has on the other. An ANN architecture dictates how the information traverse between the layers of neurons in the network, with the two main categories of architecture being MLP also known as Feed-Forward Neural Network and RNN. In a MLP, information is forwarded from the input layer via one or more hidden layers to the output layer, neurons between layers are connected by weights and one or more activation functions (Arnerić et al., 2014). Figure 3.1 shows an example of a 2-3-1 MLP architecture[4]. RNN has a similar structure to MLP except that it has feedback connection between layers, enabling reverting from one layer to the preceding layer. There is a raging debate and research in the machine learning community about which architecture should be preferred in which situations, but MLP is currently the most widely used. Studies indicate that if one has adequate number of nodes in the first hidden layer, subsequent layers are normally not that much of a necessity to achieve satisfactory results (Donaldson and Kamstra, 1997). However, Demuth et al. (2014) had a different view and assert that multilayer networks are more powerful than single layer networks. They cite an example that a two-layer sigmoid network with sigmoid first layer and a linear second layer can be trained to approximate most functions easily, a trait they claim single layer networks do not possess.

[4]Courtesy of `http://www.texample.net`

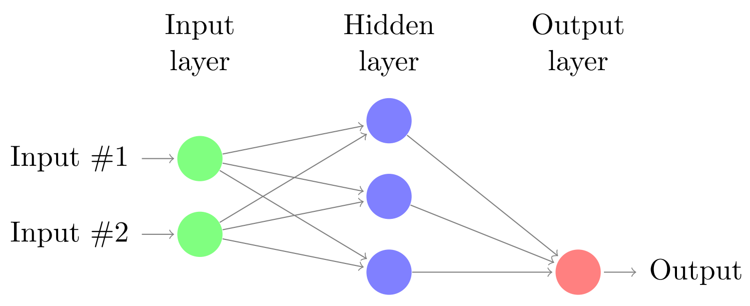


Figure 3.1: 2-3-1 MLP architecture

Tsay (2010) shows a mathematical representation of the 2-3-1 ANN illustrated in Figure 3.1. The jth node (neuron) in the hidden layer is given by

$$h_j = f_j\left(\alpha_{0j} + \sum_{i \to j} w_{ij} x_i\right) \tag{3.9}$$

where x_i is the value of the ith input node, w_{ij} are weights, α_{0j} is called a bias and $f_j(.)$ is an activation function usually logistic function of the form

$$f_j(z_j) = \frac{e^{z_j}}{1 + e^{z_j}}. \tag{3.10}$$

For the chosen 2-3-1 architecture the equation looks like the following

$$h_j = \frac{e^{\alpha_{0j} + w_{1j}x_1 + w_{2j}x_2}}{1 + e^{\alpha_{0j} + w_{1j}x_1 + w_{2j}x_2}}, j = 1, 2, 3. \tag{3.11}$$

For the output layer where we assume the linear activation function $f_o(.)$, the output node is given as

$$o = f_o\left(\alpha_{0o} + \sum_{j \to o} w_{jo} h_j\right) \tag{3.12}$$

$f_o(.)$ is either linear or Heaviside function[5] and simplifies to

$$o = \alpha_{0o} + \sum_{j=1}^{k} w_{jo} h_j, \tag{3.13}$$

where k is the number of nodes in the hidden layer. The output layer for the chosen 2-3-1 architecture becomes

$$o = \alpha_{0o} + w_{1o}h_1 + w_{2o}h_2 + w_{3o}h_3. \tag{3.14}$$

Associated with an ANN is a learning rule which is a process of transforming the weights and biases of the network, this procedure is referred to as training the network (Demuth et al., 2014). There are three kinds of training algorithms, *supervised learning*, *reinforcement learning* and *unsupervised learning*. In supervised learning, the learning rule is provided with both inputs and targets (desired outputs), the weights and biases are adjusted until the error between the network outputs and targets is minimal. Hastie et al. (2001) analogises supervised learning to a “student” learning with a “teacher”,

[5]Heaviside function, is a function such that $f_o(z) = 1$ *if* $z > 0$ *and* $f_o(z) = 0$ otherwise. It’s named after Oliver Heaviside.

where a student provides responses $\hat{y}_i$ for each x_i in a sample and the teacher provides the correct answer or error associated with the student's response. The reinforcement learning is less common, but it is similar to supervised learning except that instead of targets, the learning rule is provided with scores, the scores are a measure of the network performance given the inputs. In unsupervised learning, the learning rule is adjusted according to only inputs, it is given no outputs, it performs a form of clustering (Demuth et al., 2014). *Backpropagation* is an algorithm most widely used in training ANNs, Mazur (2015) goes through a step by step example of training a feed-forward ANN via backpropagation. Backpropagation is basically an estimated steepest descent that minimises the squared error (Demuth et al., 2014). It works backwards from the output layer modifying biases and weights iteratively using a gradient rule. The modification of biases and weights is done in order to minimize a fitting criteria like least squares (Tsay, 2010). The fitting criteria can be as follows:

$$S^2 = \sum_{t=1}^{T} (r_t - o_t)^2 . \tag{3.15}$$

Backprogration does not come without drawbacks. One of them is that since it is a minimization problem, there might be several local minima and the algorithm might be stuck in one of the local minima interpreting it as the actual global minimum. Bi et al. (2005) study this problem and suggest a modification of the error function by adding an extra term. They state that adding such a term to the error function helps to harmonize the adjustment weights in the hidden layer with those in the output layer. Another possible drawback of backpropagation is overfitting, Kayri (2016) proposed integrating Bayesian Regulation algorithm to backpropagation to minimize overfitting. Riedmiller (1994) proposed a variation of the backpropagation called *resilient backpropagation.* The idea behind the resilient backpropagation is to get rid of the 'undesirable' influence of the partial derivative in the weight step, by taking into consideration only the sign of the derivative instead of the magnitude. Resilient backpropagation also carries a promise of avoiding the stagnation around the local optima. After comparing resilient backpropagation to regular backpropagation, Prasad et al. (2013) reached a conclusion that resilient backpropagation does seem a better choice than the ordinary backpropagation.

3.2.3 Data Preprocessing

FTSE/JSE All Share Historical Data (ALSI) was sourced from the website `https://za.investing.com/indices/ftse-jse-all-share-historical-data`. The data comprises of 2517 observations, which are daily closing prices between 25-01-2009 and 22-03-2019. Weekends are excluded in the observations, so it is a 5-day week time series. The ALSI price series was converted to the log return series using the log return case of Equation (3.1).

Due to unavailability of high frequency price data for ALSI, equations (3.2) and (3.3) could not be used directly for volatility proxy. In low frequency data set up standard deviation is often used as a fairly good volatility estimate as stated in Daly (2008) and Daly (2011). A 5-day moving standard deviation was used in this research as a measure of realised volatility (volatility proxy), calculated as

$$RV_t = \sigma_t = \sqrt{\sum_{t=1}^{5} \frac{(r_t - \bar{r})^2}{4}} \tag{3.16}$$

where r_t is an asset return on day t, and $\bar{r}$ the returns mean of the corresponding 5 days. The first estimate starts on day 5, for the first five days. The second estimate starts on day 6 and applies Equation 3.16 on the five days between day 2 and day 6, and so on. This gives $n-4$ volatility proxies for n times series data points.

Hill and McCullough (2019) reviewed three popular R libraries for GARCH modelling; **tseries**, **fGarch** and **rugarch**. They mainly compared the ability to control the optimisation procedures of the libraries and as well as assessed their performances against accepted GARCH benchmarks. They found serious technical issues on **fGarch** and poor documentation. They found **tseries** library does not estimate a model with a mean and has one type of standard error as well as only one optimiser. Out of the three, only **rugarch** made mention of the accuracy of the GARCH estimates it produces. And also, only **rugarch** offers a user guide (vignette) on top of the documentation. In conclusion, they chose **rugarch** because it has various optimisers, it has two distinct standard errors, it is tunable and matches the benchmark. Although **rugarch** was not found to be top in the popularity contest in Hill and McCullough (2019), it is gaining traction in this regard as it is being used in various courses like in the Amazon AWS Rstudio course [6]. The library was chosen over the others in this research for the reasons outlined in Hill and McCullough (2019), and also because it offers the implementation of the Realized GARCH model as described in equation (3.7). The **rugarch**'s *realGARCH()* function takes in the returns series $\{r_t\}$ as well as volatility proxy $\{RV_t\}$ as inputs. The computed standard deviations and returns were used as inputs into the package and a RealGARCH model was fitted.

Google's Keras Tensorflow python API seems to be a dominant machine learning and deep learning tool and rapidly growing in popularity[7] in the machine learning community, but it has challenging software and hardware requirements in terms of CPUs and GPUs[8] for setting up workstations. The R implementation of keras is also challenging in terms of setting up and portability between workstations due to quite heavy package dependencies, although it has high prediction accuracy. It was not used in this research due to complexity in the initial setting up as well as memory requirements. The **neuralnet** R package on the other hand is lightweight and simpler and easily portable between workstations because it is just a package and does not have too many dependencies, and it is also not demanding in terms of memory requirements. The major limitation of the **neuralnet** package is that it only supports one hidden layer. This was not considered a hindrance in this research because one hidden layer is often adequate for autoregressive analysis of this kind. The package has resilient backpropagation implementation by default, but other options are also available.

MAE and RMSE were used for evaluating prediction accuracy of the models. The following equations define the two measures:

$$MAE = \frac{\sum_{i=1}^{n} |y_i - \hat{y}_i|}{n}, \tag{3.17}$$

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{n}} \tag{3.18}$$

where y_i is the realised volatility measure (RV_t) and $\hat{y}_i$ is the realised measure estimate from the model.

[6]`https://rstudio-pubs-static.s3.amazonaws.com/534670_78d421a4edc94d27a69d49a1c966c9ff.html`

[7]https://www.forbes.com/sites/janakirammsv/2020/11/27/tensorflow-turns-5five-reasons-why-it-is-the-most-popular-ml-framework/?sh=1b40ce967e67

[8]https://www.tensorflow.org/guide/gpu

RMSE penalises big differences between actual and predicted values (big errors) because it squares the differences before taking a square root while MAE is just an average of absolute differences. A low value is preferred in both evaluation metrics. Wang and Lu (2018) study the two metrics in detail with focus on the impact of rounding bias.

The data was split into 70% for model fitting (training) and 30% hold-out sample for testing. The performance metrics were measured on the hold-out sample.

3.2.4 Data Analysis and Results

3.2.4.1 Realised GARCH Model

3.2.4.1.1 Data Exploration

In this section the statistical properties that make returns series to be more suitable than price series in volatility modelling were explored. Figure 3.3 shows the normal Q-Q plots of both series, and Table 3.1 is the summary statistics. In theory, stock prices are usually modelled as lognormal distribution and the log returns as normal distribution. Both formal normality tests and graphical inspection of normality via the Q-Q plots were carried out.

The Q-Q plot for the log returns showed a slightly symmetric distribution with heavy tails while prices showed a negatively skewed distribution [9]. This graphical observation is corroborated by the skewness values of the log returns and those of prices in Table 3.1, which are 0.07801 and -0.3292 respectively. The closer the value of skewness to zero, the more symmetric is the distribution. Normal distribution has a skewness of zero. Another relevant descriptive statistic to normality checks is kurtosis[10]. The log returns exhibited leptokurtosis (heavy tails), which is an excess kurtosis greater than zero[11]. The leptokurtic log returns were consistent with the analysis by Andersen et al. (2003). This signified the existence of outliers in the log returns, which in turn indicated the log returns were susceptible to extreme values on either sides, very low returns or very high returns. The prices on the other hand were platykurtic (excess kurtosis less than zero) which is signified by flat tails with less likelihood of extreme values.

Quantitative normality tests were carried out in the form of Shapiro-Wilk, Anderson-Darling and Jarque-Bera tests. The tests results are shown in Table 3.1. The null hypothesis for the Shapiro-Wilk test is that the data comes from a normal distribution. The null hypothesis for the Anderson-Darling test is that the data comes from *a specified distribution* (Heckert et al., 2002). In the *ad.test()* function of the **nortest** R library that specified distribution is a normal distribution. The Jarque-Bera test is based jointly on skewness and kurtosis. Its null hypothesis is that skewness is zero and kurtosis is 3 (Yap and Sim, 2011), which are features of a normal distribution. All the p-values were less than 0.05, which meant that all the null hypotheses were rejected at 0.05 level of significance. Both log returns and ALSI prices datasets were not normally distributed although log returns were more symmetric when viewed through skewness but had heavy tails as seen in kurtosis (and on Q-Q plot) which could have caused a deviation from normal distribution.

[9]Skewness is defined as a third central moment which measures the symmetry to the mean (Tsay, 2010).

[10]Kurtosis is defined as a fourth central moment which measures tail behaviour (Tsay, 2010).

[11]R is using the formula $kurtosis = \frac{\sum_{i=1}^{N}(Y_i - \overline{Y_i})^4/N}{s^4} - 3$ which makes normal distribution kurtosis to be zero instead of three.

Another fact worth mentioning from the summary statistics is that the log return series had one less observation than the price series. This is because a log return is a return between any two successive days, so there was a 1-day lag in the log return series.

Table 3.1: Summary Statistics

ALSI Prices	
no. of observations	2517
NAs	0
Minimum	61684.77
Maximum	18120.69
1. Quartile	32138.54
3. Quartile	52437.84
Mean	42964.33
Median	47045.44
Sum	108141200
SE Mean	225.7951
LCL Mean	42521.57
UCL Mean	43407.09
Variance	128325300
Stdev	11328.07
Skewness	-0.32915
Kurtosis	-1.2915
statistic	0.9169
p-value	2.2e-16
statistic	86.915
p-value	2.2e-16
statistic	220.06
p-value	2.2e-16

Log Returns Summary Statistics		
no. of observations	2516	Sample size
NAs	0	Number of missing values
Minimum	-5.601475	
Maximum	3.693919	
1. Quartile	-0.636298	25th percentile
3. Quartile	0.509134	75th percentile
Mean	-0.043421	Sample Mean
Median	-0.071776	Sample Median
Sum	-109.247866	Sum of the percentage simple returns
SE Mean	0.02013	Standard error of the sample mean
LCL Mean	-0.082894	Lower bound of 95% conf. interval for mean
UCL Mean	-0.003949	Upper bound of 95% conf. interval for mean
Variance	1.019504	Sample variance
Stdev	1.009705	Sample standard error
Skewness	0.078007	Sample skewness
Kurtosis	1.503384	Sample excess kurtosis
statistic	0.98403	Shapiro Wilk
p-value	3.276e-16	Shapiro Wilk
statistic	9.5954	Anderson-Darling
p-value	2.2e-16	Anderson-Darling
statistic	240.63	Jarque-Bera
p-value	2.2e-16	Jarque-Bera

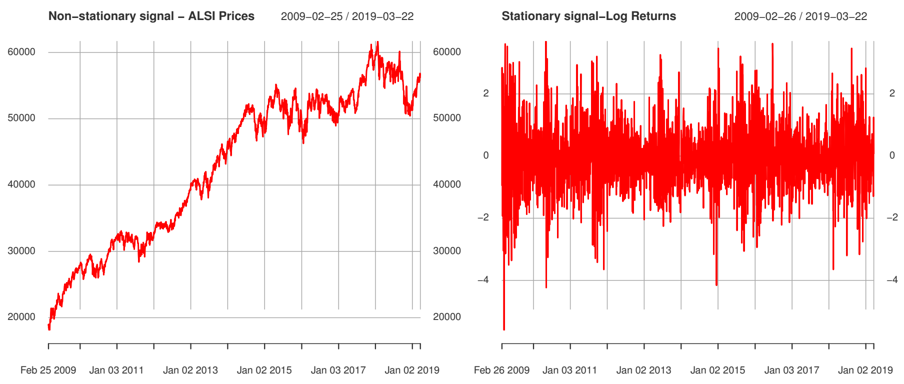


Figure 3.2: All Share Index Prices and Log Returns

Tsay (2010) described stationarity as the foundation of time series analysis. As previously mentioned, EMH is assumed in finance. Under the assumption, there should be no serial correlation[12] in the time series. Stationarity tests show whether a series is stationary or non-stationary. Prior to fitting the data to a GARCH model such tests were performed, this was necessary to ensure that the series was constant in the mean and variance and not dependent on time, and therefore obeyed EMH. If the time series is not stationary, some other remedy like differencing can be taken before fitting the data to the model. The pre-fitting data transformation like differencing also determine which flavour of a GARCH model would eventually be fitted.

Again, in justifying the usage of log returns over prices, analysis of both was done. Figure 3.2 shows

[12]Serial correlation is the relationship between a variable and a lagged version of itself over various time intervals

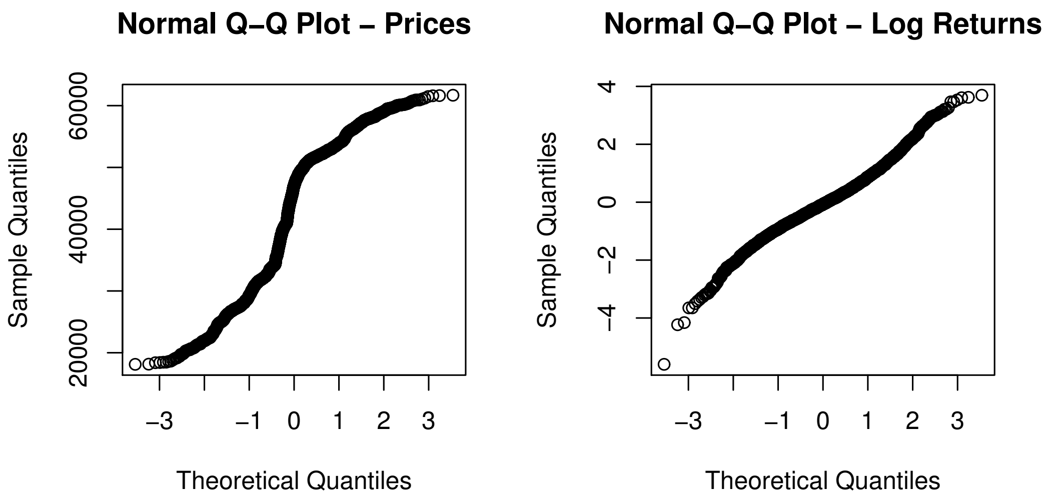


Figure 3.3: Normal Q-Q of ALSI Prices and Log Returns

the time plots of ALSI prices and log returns respectively. By inspection, it can be seen that the prices plot shows a changing upward trend and the log returns plot seems to be oscillating about a constant mean. The log returns on the other hand show a stationery series.

One way to formally examine stationarity is to look at the autocorrelation function (ACF) of the time series. Correlation coefficient, ρ_l, between r_t and r_l is defined as

$$\rho_l = \frac{Cov(r_t, r_l)}{\sqrt{Var(r_t)Var(r_l)}} = \frac{Cov(r_t, r_l)}{Var(r_t)}. \tag{3.19}$$

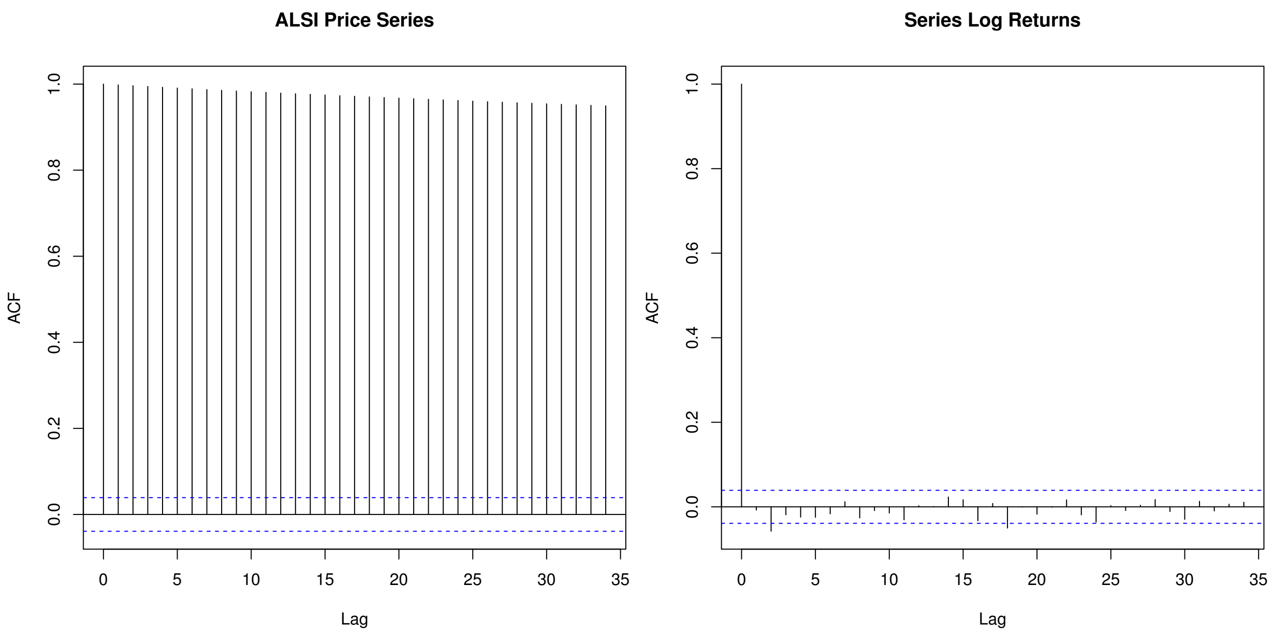
For a weakly stationary series $Var(r_t) = Var(r_l)$, $\rho_0 = 1$,$\rho_l = \rho_{-l}$ and $1 \leq \rho_l \leq 1$. If $\bar{r}$ is a sample mean defined as $\bar{r} = \sum_{t=l}^{T} r_t/T$ then the $lag - l$ sample autocorrelation is given by

$$\hat{\rho}_l = \frac{\sum_{t=2}^{T}(r_t - \bar{r})(r_{t-l} - \bar{r})}{\sum_{t=1}^{T}(r_t - \bar{r})^2} \tag{3.20}$$

(Tsay, 2010). An autocorrelation $\rho_l = 0$ indicates lack of serial correlation.

Figure 3.4 shows autocorrelation functions of the two series, ALSI prices and the corresponding log returns. As can been observed in Figure 3.4a, the price series autocorrelation was decaying slowly and was remarkably above the significant range (dotted lines), this was indicative of a non-stationery series. On the other hand the log returns series was mostly within significance range, this was indicative of a stationery series.

Stationarity was concluded by conducting a quantitative unit root test called Augmented Dickey-Fuller Unit Root Test. This test checks for the presence of a unit root which is a stochastic trend in a time series that indicates an unpredictable systematic pattern. The null hypothesis states that there is a unit root and the alternative is that the series is stationary. The test was carried using the *adfTest()* function of the **fUnitRoots** package. Three tests were conducted for each time series as reflected in Table 3.2. The price series lag order was determined using *ur.df2()* of the **erer** package. The function uses Akaike information criterion (AIC) to select the optimal number of lags, and for the price series the number of optimal lags was 2. For the log return series, the number of lag order was determined



(a) ACF ALSI Prices (b) ACF Log Returns

Figure 3.4: Autocorrelation Function for ALSI Prices and Log Returns

based on the PACF in Figure 3.5, the lag order was 19. The *ur.df2()* function gave 1 as the optimal number of lags for the return series, but using 1 did not change the results. The null hypothesis could not be rejected for the price series but it was rejected for the return series at 0.05 level of significance. In summary, the price series has a unit root, drift and time trend while for the return series there is no evidence of a unit root, drift or time trend.

Table 3.2: Augmented Dickey-Fuller Unit Root Tests

	Lag Order	ADF-statistic	p-value
ALSI Prices (no intercept & no trend)	2	1.3726	0.9568
ALSI Prices (intercept & no trend)	2	-1.8284	0.3773
ALSI Prices (intercept & trend)	2	-3.0822	0.1202
Log Returns (no intercept & no trend)	19	-12.5085	0.01
Log Returns (intercept & no trend)	19	-12.8266	0.01
Log Returns (intercept & trend)	19	-12.997	0.01

Part of the assumption of the EMH is that stock returns are unpredictable and therefore should have no serial correlations Tsay (2010). So, in concluding the data analysis was a much more stronger test which tests for the presence of serial correlation, the test is called Ljung-Box test (named after Greta M. Ljung and George E. P. Box). The null and alternative hypotheses are stated as follows:

$$H_0 : \rho_1 = \ldots = \rho_m = 0$$

$$H_a : \rho_i \neq 0$$

for some $i \in \{1 \ldots m\}$. Table A.1 shows the results of the Ljung-Box tests, the p-value of the price series is less that 0.05%, which means the null hypothesis is rejected at 95% level of confidence, and the p-value for the log return series is greater 0.05 and therefore the null hypothesis could not be rejected. This leads to the conclusion that the log return series has no significant autocorrelations and therefore obeys the EMH.

The analysis in the preceding paragraph has shown that log returns had the desirable properties like stationarity and absence of autocorrelations which made them more suitable for GARCH modelling compared to stock prices.

Table 3.3: ARCH Effect test

ARCH Effect Test type	Residuals	DF	X-squared	p-value
Box-Ljung test $\{a_t^2\}$	19	851.51	2.2e-16	
ARCH LM test $\{a_t\}$	19	260.46	2.2e-16	

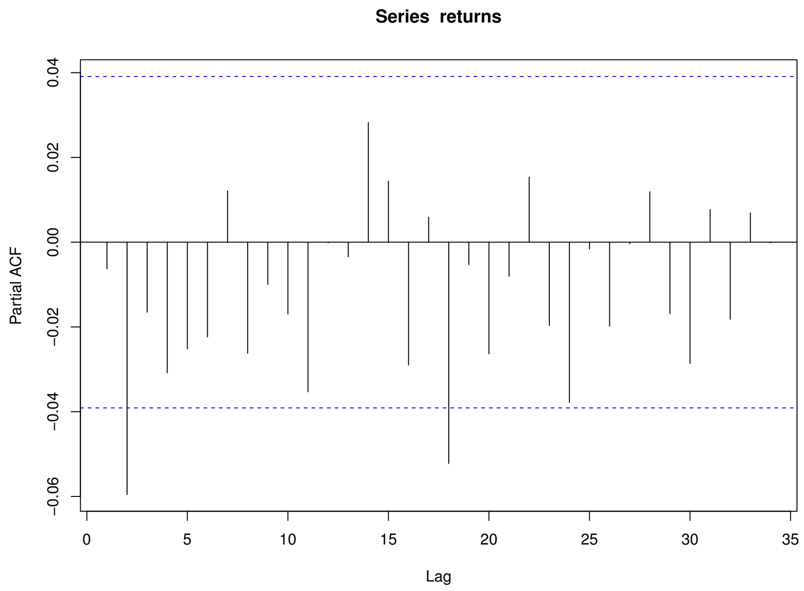


Figure 3.5: Log Returns PACF

3.2.4.1.2 Model Fitting

Since the log returns showed no serial correlations as shown in the preceding Ljung-Box test, the next step was to ascertain the suitability of a GARCH model. The GARCH model building followed the steps as set out in Tsay (2013):

- **Mean equation specification**. The PACF of the log returns in Figure 3.5 shows a significant spike at lag 2 which is an indication of a possible AR(2) process. And, the ACF of the log returns in Figure 3.4b shows a significant spike at lag 2 as well, which signifies a possibility of an MA(2) process. This visual assertion was also confirmed by the *auto.arima()* function of the **forecast** package, as the name suggests, the function automatically detects ARIMA model parameters from data. The chosen mean equation was an ARMA(2,2) model and it is shown in Table A.2.

- **Testing for ARCH effects**. The squared residual series a^2 of the ARMA(2,2) model were tested for serial correlation using Ljung-Box test. The null hypothesis of zero correlation is rejected at 0.05 level of significance as shown in Table 3.3. The second test was the Lagrange multiplier test for ARCH effects (ARCH LM test). The function *ArchTest()* of the **FinTS**[13]

[13](Tsay, 2013)

package implements the ARCH LM test. Again the results are shown in Table 3.3, the null hypothesis of no ARCH effects is rejected at 0.05 level of significance. Both tests confirmed the presence of ARCH effects in the log returns.

- **GARCH model specification.** The preceding two steps paved a way for modeling volatility using a GARCH. The RealGARCH was the variant chosen given the realised volatility proxy introduced in Section 3.2.3, and on the basis that RealGARCH takes both the returns as well as the volatility measure (proxy) as discussed in Section 3.2.2.1.

- **Model validity check.** Four ARMA-RealGARCH with ARMA(2,2) mean equations were evaluated as shown in Table 3.4. The order of RealGARCH was restricted to a maximum of (2,2), this was to keep it in lower orders as generally is the practice as alluded to by Tsay (2013). The best model fit of the four as indicated by AIC and BIC was ARMA(2,2)-RealGARCH(1,1). But, this was not the best performing model as measured by MAE and RMSE, the best performing model amongst the ones with ARMA(2,2) mean equation was ARMA(2,2)-RealGARCH(2,2). Further model checks outside the ARMA(2,2) mean equation group were carried, and it was found out that ARMA(2,1)-RealGARCH(2,1) was a better model by both model fit (AIC, BIC) and prediction accuracy (MAE,RMSE). The mean equation ARMA(2,1) is shown in Table A.2, and it shows a higher AIC than ARMA(2,2) which signifies inferior model fit. It was also tested for ARCH effects and the ARCH effects were confirmed by both Ljung Box test and Langrange multiplier tests.

  The fitted models were evaluated further using the **rugarch** package diagnostics which are shown in tables A.3 and A.4 of Appendix A.3.1. In the tables the rows with statistically non-significant variables are crossed out with lines. The two tables represent ARMA(2,2)-RealGARCH(1,1) and ARMA(2,1)-RealGARCH(2,1) which were the best fitted models according to the AIC's of the RealGARCH model group in Table 3.4. The outputs show the Ljung-Box tests on the standardised residuals of each of the RealGARCH models. The null hypothesis of no serial correlation is rejected in both models, the standardised residuals are therefore serially correlated. The Ljung-Box test on the squared residuals evaluates for adequately fitted ARMA mean equation. Table A.3 shows that the model ARMA(2,2)-RealGARCH(1,1) is adequately fitted up to lag 5 while Table A.4 shows inadequacy of the ARMA mean equation for the ARMA(2,1)-RealGARCH(2,1) model. This result is supported by the visual inspection of the PACF and ACF as well as by the *auto.arma()* function which suggested ARMA(2,2) as the superior mean model equation. Both models showed persistent ARCH effects according to the ARCH LM test results with ARMA(2,2)-RealGARCH(1,1) having ARCH effects only at lag 8 and ARMA(2,1)-RealGARCH(2,1) having ARCH effects at both lag 6 and lag 8.

  The **Sign Bias Test** is used to test leverage effects, with the null hypothesis that there is no leverage effects. Again, all the RealGARCH models showed presence of leverage effects with ARMA(2,2)-RealGARCH(1,1) showing leverage effects only on the joint statistic. These leverage effects results were surprising because RealGARCH has a 'leverage effect function' which is suppose to give it EGARCH-like properties. The presence of leverage effects suggested an alternative GARCH variant may be required. EGARCH and TGARCH are designed to capture leverage effects. EGARCH was chosen as an alternative model to explore. The model ARMA(2,2)-EGARCH(1,1) was fitted to the data, the resulting model statistics are shown in Table A.5. This model had the best AIC, -6.5427. There was no evidence of serial correlation in its standardised residuals, the ARMA(2,2) mean equation was adequate, and the ARCH process was adequate. The model also adequately captured the leverage effects according to the sign bias test. Theoretically, this is the best model for the log returns data but unfortunately it predicts latent volatility as opposed to realised volatility, it therefore could not serve as a basis for comparison to a nonparametric artificial neural network which is trained by realised volatility (that would not be comparing the proverbial apples to apples).

The ARMA(2,2)-RealGARCH(1,1) model had a better ARCH process, the ARCH effects were persistent only at higher lags. The leverage effects were generally captured by the model, only the joint statistic of the sign bias test showed leverage effects. No further transformation on the log returns data gave improvement on the RealGARCH model fitting, therefore, ARMA(2,2)-RealGARCH(1,1) was chosen as the basis of comparison.

Table 3.4: GARCH Models Comparison

		(2,2:1,1)	**(2,2:2,1)**	**(2,2:1,2)**	**(2,2:2,2)**	**(2,1:2,1)**
ARMA-realGARCH	**MAE**	0.1558108	0.1522862	0.1563295	0.151006	0.1498012
	RMSE	0.2320173	0.2284904	0.2314191	0.2265984	0.2258623
	AIC	-6.5304	-6.5297	-6.5296	-6.5287	-6.5312
	BIC	-6.5003	-6.4973	-6.4971	-6.4939	-6.5010
ARMA-eGARCH	**MAE**				N/A	
	RMSE				N/A	
	AIC				-6.5422	
	BIC				-6.5149	

3.2.4.2 Artificial Neural Network

3.2.4.2.1 Model Fitting

The *neuralnet()* function requires a linear model formula which shows the dependent and predictor variables. The volatility data was therefore transformed to conform to the autoregression equation, AR(2), of the form

$$y_t = \phi_0 + \phi_1 y_{t-1} + \phi_2 y_{t-2} + a_t, \tag{3.21}$$

where y_t is a realised volatility measure (RV_t) at time t, a_t is a white noise series with mean zero and variance σ^2, ϕ's are autoregressive coefficients. The parameter p in AR(p) is lag order of the autoregression equation and it was determined using Figure 3.4b, where there is significant autocorrelation on lag 2. This preprocessing step shows that the package is instructed to regress volatility value today as dependent to the volatility values of the previous two days (autoregression).

The package allows for the altering of the number of neurons in the hidden layer. Neurons between 3 and 10 in the hidden layer were fitted to the data and performance matrix taken to determine the optimal number of neurons, performance matrix is shown in Appendix A.4.1. The best prediction accuracy was found at six hidden layer neurons, and *neuralnet* only allows only one hidden layer. That meant that the optimal model was 2-6-1 MLP, the input layer with two input nodes as represented in Equation (3.15). Figure 3.6 shows the *neuralnet*'s graphical representation of the fitted model, with corresponding synaptic weights and the number of backpropagation iterations (Steps). Appendix A.4.1.1 shows the model output of the fitted ANN model.

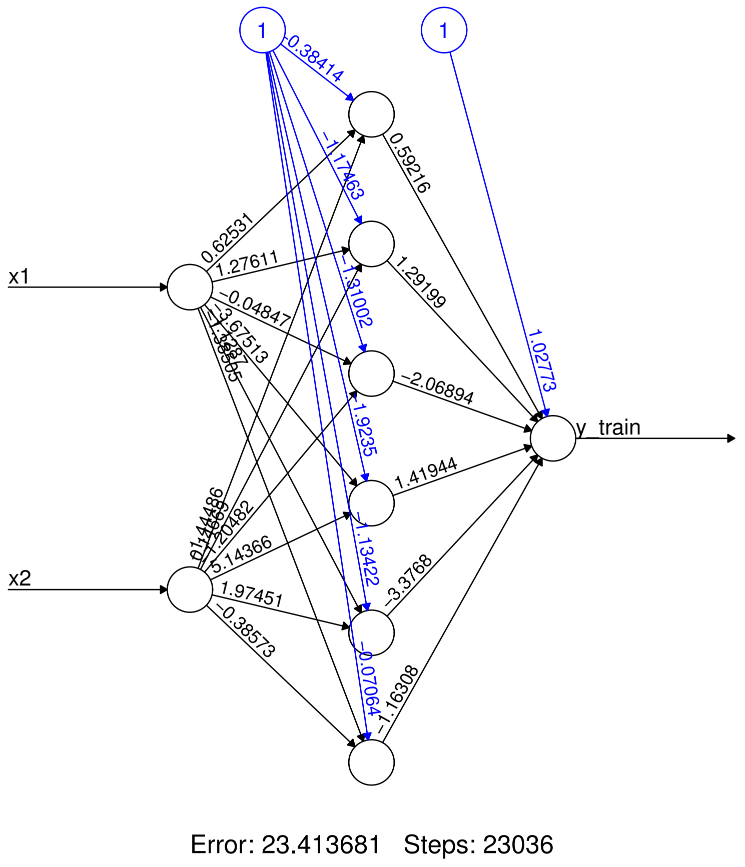


Figure 3.6: ANN Layers with Weights

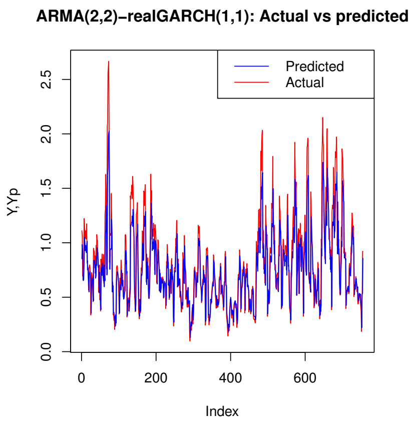
3.2.4.3 Results

Figure 3.7 shows plots of the actual values against the predicted values for the ANN and the ARMA(2,2)-RealGARCH(1,1) model. The red colour represents the actual values. The blue colour represents the predicted values which are graphically superimposed on the actual values. A dominance of red colour on the graphs indicates dispersion between actual and predicted values. Figure 3.7a appear to have more red color than figure 3.7b which indicates the ANN model fared better in tracking the true values.

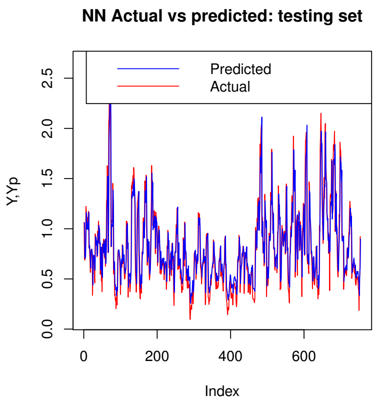
Table 3.5: Prediction Accuracy Results

	RMSE	MAE
ANN	0.2161168	0.14651724
ARMA(2,2)-RealGARCH(1,1)	0.2320173	0.1558108
ARMA(2,1)-RealGARCH(2,1)	0.2258623	0.1498012

Table 3.5 shows the numerical results of the comparison. The ANN model outperformed the Real-GARCH models in both RMSE and MAE. The differences are quite small when the ANN model is compared to ARMA(2,1)-RealGARCH(2,1) but this model is much more inadequate in all of the selected model validity measurements. In keeping up with the statistical modelling culture, a model with a more superior prediction accuracy was sacrificed for the one with a higher level of adequacy. As Davison and Hinkley (1997) summarised it, “The explicit recognition of uncertainty is central



(a) ARMA(2,2)-RealGARCH(1,1)



(b) Neural Network

Figure 3.7: Results Plots

to the statistical sciences". Bzdok et al. (2018) contrasts this notion of uncertainty by stating that machine learning focuses on finding patterns in unwieldy data using general-purpose algorithms. So, quantifying uncertainty is critical part of statistical modelling but not necessarily the case in machine learning.

Appendix A.5 contains the R code listing for the analysis in this case study.

3.2.5 Discussion and Conclusion

3.2.5.1 Discussion

The ANN model performed better than all the RealGARCH models evaluated in this case study. The result is consistent with the studies that were reviewed in Section 2.2.1 which show machine learning as having an edge over GARCH in prediction accuracy. But, GARCH has maturity and strong theoretical base in volatility modelling. In this research, there was more discovered about the data in the GARCH models than in the ANN model. For example, discovering the leverage effects gives an analyst extra information about the data and such information can provide guidance on what action should be taken. The action is not always about choosing a better prediction model, it could also be about correcting the data from source depending on the purpose of the data. So, significance tests also help with detecting data anomalies. It is difficult to imagine some of the volatility properties like stationarity, leverage effect and volatility clustering being unearthed outside the realm of inferential statistical modelling in one form or another. In this case study, even if the RealGARCH models had outperformed the ANN model, a statistical data modeller would probably be preoccupied about the persistence of the GARCH effects and the leverage effects in fitted models. Such preoccupation could lead to improvements in the data modelling or improve the process that generates the data (in cases where it is not from a stock exchange).

Section 3.2.3 touched on the challenges of obtaining 5-minute frequency intra-day data which would have been ideal for modelling realised volatility as it was mentioned in the discussion of volatility characteristics. The difficulties in obtaining high frequency intra-day data were also raised by Andersen and Bollerslev (1998), they questioned whether the volatility forecast precision that comes with such data is worth the trouble. This difficulty in obtaining the ALSI high frequency data led to a use of a probably less robust volatility proxy, which probably led to the less than ideal prediction accuracy by the RealGARCH models. RealGARCH models are data hungry models. In Section 1.1 a picture of enormous amounts of data that gets generated daily is drawn, although that is a fact, but data sparsity is still a reality. In many situations the data that is available is not the data that is required for a specified modelling problem. It helps to have a tool that will scrape through the data to find patterns regardless of data size and form, and the ANN model was that tool in this case study. And, this is the general attraction of the machine learning algorithms.

Interpretability is a subjective matter and largely depends on the audience. Figure 3.6 shows the fitted ANN model's 2-6-1 architecture with corresponding weights and Appendix A.4.1.1 shows the model's summary in Table A.7. The tables in Appendix A.3.1 show the fitted GARCH models (RealGARCH and EGARCH). The RealGARCH models shows a lot more information compared to the ANN model. The significant variables can be clearly identifiable, and other diagnostic metrics for goodness of fit are also shown. The summary output can be mapped back to Equation (3.7) after removing the insignificant variables. The ANN models summary on the other hand shows a series of matrices, lists containing other lists. Granted, this could be a frailty of the package **neuralnet** which makes interpretability poor. A few more R packages were explored during the data modelling and **neuralnet** was found to be relatively more parsimonious with minimum hardware requirements. And, it was seen to be more interpretable compared to others. Barring the variation in implementation across different packages, a GARCH model is quite more interpretable compared to an ANN model.

3.2.5.2 Conclusion

This case study confirmed the edge of machine learning in predictive modelling, in the same breath it confirmed the edge of statistical data modelling in uncovering causation. In many instances these are two sides of the same coin. People might be happy to have a model that will accurately predict where they are likely to get robbed, but they might also want to know the cause of spikes in robberies. Option traders would love to have a model that accurately predicts volatility for their option pricing, but risk managers would most likely want to know the cause of spikes and slumps in volatility. In fitting the RealGARCH, knowing that the cause of the poor model fit was the leverage effect informed the decision to try a different GARCH variant in EGARCH. It is clear from the empirical literature reviewed and the analysis in this case study that neither of the two paradigms is all-encompassing. The context and objective of the analysis should inform the choice of the modelling regime. There is a promise that comes from the hybrid models which are actively being researched, as seen in Donaldson and Kamstra (1997) and Yim (2002). If these hybrid models prove to have consistency and robustness, the dilemma of having to choose between prediction accuracy of the machine learning algorithms and causal inference of the statistical data models would be settled.

3.3 LDA versus SVM Case Study

3.3.1 Introduction

Classification is another form of predictive modelling. It is also an example of supervised learning where the model is given the true values on the training set (in sample) and expected to predict the true values in the validation set (out of sample). It differs from the prediction problem studied in the GARCH and ANN case study in that the response variable is qualitative (categorical) while in the prediction problem the response variable was quantitative. The categorical responses are usually referred to as classes and can vary from two (binary response) to many classes. There are many applications of classification models; calculating probability of default in credit scoring, spam email detection, diagnosis of presence or absence of a particular disease and many more. In statistics the two most popular classification techniques are logistic regression and LDA. LDA is known to perform better than logistic regression in cases where the number of classes are more than two (Hastie et al., 2013). For that reason, LDA was used as a data model based classification technique since a multi-class classification problem was also studied. There are other variants of LDA which attempt to address some of its known shortcomings like HDA which seeks to address the assumption of equal within-class covariance and quadratic discriminant analysis (QDA) that seeks to address non linear separation of data. These variants show the ongoing research in statistical modelling in contrast to the claim by Breiman (2001) that statistics is stuck with logistic regression and discriminant analysis. LDA itself is not only used for classification but it has also been applied in dimension reduction problems. In machine learning, ANNs are also used in classification problems and Lesniak et al. (2012) compares SVM to ANN in a classification problem. In fact, SVMs are widely seen as a form of ANNs but without the challenges of backpropagation and gradient descent which were mentioned in the GARCH/ANN case study. There are many other classification algorithms in machine learning[14]. The goal of this case study was to compare machine learning and statistical data modelling in classification using SVMs and LDA. The following subsections delve into the theory and the mechanics of SVMs and LDA.

3.3.2 Classification Theoretical Framework

3.3.2.1 Linear Discriminant Analysis (LDA)

One way of doing classification given a categorical dependent variable Y and independent predictor variable $X_1 \cdots X_p$(predictor variables are also called features), is to model it directly by finding $P(\mathbf{Y} = \mathbf{k}|\mathbf{X} = \mathbf{x})$. This simply calculates the probability of getting a particular category (k) or simply kth class of Y given particular feature x. Logistic regression achieves this by using a logistic function

$$p(y) = \frac{e^{\beta_0+\beta_1 X_1+\beta_2 X_2+\ldots+\beta_p X_p}}{1+e^{\beta_0+\beta_1 X_1+\beta_2 X_2+\ldots+\beta_p X_p}} \tag{3.22}$$

where p is the number of predictors (Hastie et al., 2013). Discriminant analysis flips this logic by approaching the probability from the opposite end, that is, $P(X = x|Y = y)$.

To illustrate the differences between the two approaches Ng (2018) makes an example of classification problem between elephants and dogs, dogs being one class and elephants being another class category. In this case a logistic regression model would find a straight line boundary based on animal features, and given a new animal, it will then check on which side of the boundary does it fall to make the appropriate classification. In discriminant analysis, a dog model is built based on features that makes up a dog, and an elephant model is built based on elephant features in the training set. When a new animal is introduced, it is first compared to both models separately and determination is made as to which model does the new animal has the most resemblance, then it is classified to the category of the model it resembles the most.

This probability $P(X = x|Y = y)$ is not calculated directly and Bayes' theorem is utilized. If $f_k(X) \equiv$

[14]Machine Learning practitioners do not really discriminate against statistical models, so logistic regression and LDA are usually listed in ML syllabus

$P(X = x|Y = y)$ is a density function of X for kth category observation, Bayes' theorem states that

$$P(Y = k|X = x) = \frac{\pi_k f_k(x)}{\sum_{l=1}^{K} \pi_l f_l(x)}. \tag{3.23}$$

In the equation above π_k is called a prior probability and is estimated from the training set as a proportion of observations that belong to the kth class. $P(Y = k|X = x)$ or just $p_k(X)$ is called a posterior probability, which is the probability that the observation belong to the kth class given the predictor value (Hastie et al., 2013). Unlike π_k, $f_k(x)$ is difficult to estimate from data but any probability density can be assumed. In this study we assume normal density which is represented by the following formula

$$f_k(x) = \frac{1}{\sqrt{2\pi}\sigma_k} e^{-\frac{1}{2\sigma_k^2}(x-\mu_k)^2}, X \sim N\left(\mu, \sigma^2\right) \tag{3.24}$$

and for $p = 1$ (uni-variate) and assuming variance is the same for the k classes, the posterior probability in equation (3.23) becomes

$$[H]p_k(x) = \frac{\pi_k \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2\sigma^2}(x-\mu_k)^2}}{\sum_{l=1}^{K} \pi_l \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2\sigma^2}(x-\mu_l)^2}}. \tag{3.25}$$

The classification rule is to assign an observation to a kth class for which $p_k(x)$ is the largest. Various studies has shown that the classifier in equation (3.25) is equivalent to

$$\delta_k(x) = x.\frac{\mu_k}{\sigma^2} - \frac{\mu_k^2}{2\sigma^2} + \log(\pi_k). \tag{3.26}$$

Again an observation is assigned to a kth class for which $\delta_k(x)$ is the largest (Hastie et al., 2013). The inputs μ_k and σ can be estimated from the training set as follows

$$\hat{\mu} = \frac{1}{n_k} \sum_{i:y_i=k} x_i \tag{3.27}$$

$$\hat{\sigma}^2 = \frac{1}{n-K} \sum_{K=1}^{K} \sum_{i:y_i=k} (x_i - \hat{\mu_k})^2. \tag{3.28}$$

The estimates yield

$$\hat{\delta}_k(x) = x.\frac{\hat{\mu}_k}{\hat{\sigma}^2} - \frac{\hat{\mu}_k^2}{2\hat{\sigma}^2} + \log(\pi_k). \tag{3.29}$$

This is a linear discriminant function (DF) of x which makes this classifier to be called **linear discriminant analysis** (LDA) but in other literature it is called Gaussian discriminant analysis because of the normal density assumption (Ng, 2018).

In the case when $p > 1$ (multivariate), the following equations are analogous to equations (3.24) and (3.26)

$$f(\mathbf{x}) = \frac{1}{(2\pi)^{\frac{p}{2}} |\mathbf{\Sigma}|^{\frac{1}{2}}} e^{\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T \mathbf{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right)} \tag{3.30}$$

$$\delta_k(\mathbf{x}) = \mathbf{x}^T \mathbf{\Sigma}^{-1} \boldsymbol{\mu}_k - \frac{1}{2} \boldsymbol{\mu}_k^T \mathbf{\Sigma}_k^{-1} + \log \pi_k \tag{3.31}$$

where X is a $p-$dimensional random variable and $\mathbf{\Sigma}$ is a $p \times p$ covariance matrix of X. The covariance matrix poses particular challenges as $p \to \infty$, say $p = 100,000$, then $\mathbf{\Sigma}$ is a $100,000 \times 100,000$ which can cause computation performance issues.

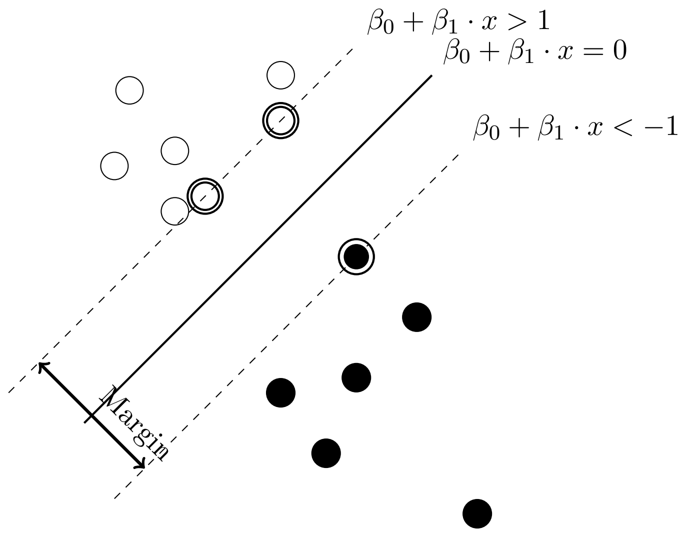

$\beta_0 + \beta_1 \cdot x > 1$
$\beta_0 + \beta_1 \cdot x = 0$
$\beta_0 + \beta_1 \cdot x < -1$
Margin

Figure 3.8: The maximal margin hypeplane.
Courtesy of `https://elbauldelprogramador.com/en/creating-trees-dependency-graphs-svms-in-tikz/`

LDA has been successfully applied in areas such as credit evaluation, bankruptcy prediction, marketing and many more. But LDA has also been criticised for having poor performance when the equal within class covariance assumption has been violated (Stapor, 2016). This led to the development of the HDA.

3.3.2.2 Support Vector Machines

The definition of a hyperplane in p dimensions is given by the following equation

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta_p X_p = 0. \tag{3.32}$$

This in fact results in a $p-1$-dimensional hyperplane. So in two dimensions the equation is $\beta_0 + \beta_1 \cdot x = 0$ which is a straight line (see Figure 3.8), in three dimensions it is a flat plane. When this equation does not hold, that is, this linear combination does not equal zero, that means X lies on either side of the hyperplane. If $\beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta_p X_p > 0$, X lies on one side and if $\beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta_p X_p < 0$, X lies on the other side. So the hyperplane divides the $p-$dimensional space into two halves (Hastie et al., 2013).

This is equivalent to having two classes $Y_i = 1$ and $y_i = -1$ with the following property

$$\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \beta_3 x_{i3} + ... + \beta_p x_{ip} > 0 \; \textit{if} \; y_i = 1 \tag{3.33}$$

and

$$\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \beta_3 x_{i3} + ... + \beta_p x_{ip} < 0 \; \textit{if} \; y_i = -1. \tag{3.34}$$

So, for a particular observation x^* with $f(x^*) = \beta_0 + \beta_1 x_1^* + \beta_2 x_2^* + \beta_3 x_3^* + ... + \beta_p x_p^*$, x^* will be classified to class -1 if $f(x^*) < 0$ and will be classified to class 1 if $f(x^*) \geq 0$ (Hastie et al., 2013).

There can be infinitely many hyperplanes that separate the training data into distinct groups with no overlaps. A hyperplane that is the furthest from the training observations is chosen as an optimal hyperplane and it is called *maximal margin hyperplane*. A *margin* is a smallest perpendicular distance from the hyperplane to each of the training observations, a hyperplane with the largest margin is the maximal margin hyperplane. An observation is then classified according to which side of the maximal margin hyperplane it lies, this is called *maximal margin classifier* (Hastie et al., 2013). Figure 3.8 illustrates the maximal margin hyperplane for the case $p = 2$.

In Figure 3.8, there are three observations that are of equal distance from the maximal margin hyperplane, if these observations were to move, the maximal margin hyperplane would change but a move on any other observations would not have such effect unless such move is such that the margin is changed. The three observations are called *support vectors* (2−dimensional vectors in this case).

The maximal margin hyperplane is a solution to the following maximization problem

$$\begin{aligned} \underset{\beta_0,\beta_1,\beta_2,...,\beta_p}{\text{maximize}} \quad & M \\ \text{subject to} \quad & \sum_{j=1}^{p} \beta_j^2 = 1, \\ & y_i\left(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \beta_3 x_{i3} + ... + \beta_p x_{ip}\right) \geq M \ \forall \ i = 1, 2, ...n. \end{aligned} \tag{3.35}$$

M represents the margin in the maximization problem, the objective is to find values $\beta_0, \beta_1, \beta_2, ..., \beta_p$ that maximize M.

Maximal margin classifier depends on the existence of a maximal margin hyperplane which neatly separates the groups. But this is not always possible, in many cases such hyperplane which achieves such neat separations does not exist because of the overlaps in the observations. Another classifier called *support vector classifier* (also called *soft margin classier*) relaxes the requirements of the maximal margin classifier by allowing observation mix up to some extent, which means, some observations from $y_i = -1$ appear in $y_i = 1$ and vice versa(Hastie et al., 2013). By allowing a possibility of misclassification, this soft margin introduces some robustness because it is not sensitive to movements of one support vector observation (which makes maximal margin susceptible to overffitng the training data). The maximization problem is also adjusted to make room for misclassification

$$\begin{aligned} \underset{\beta_0,\beta_1,\beta_2,...,\beta_p}{\text{maximize}} \quad & M \\ \text{subject to} \quad & \sum_{j=1}^{p} \beta_j^2 = 1, \\ & y_i\left(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + ... + \beta_p x_{ip}\right) \geq M(1 - \epsilon_i), \\ & \epsilon_i \geq 0, \sum_{i=1}^{n} \epsilon_i \leq C. \end{aligned} \tag{3.36}$$

The objective is as before, maximize M, and M is still the margin. The new variables $\epsilon_0, \epsilon_2, ..., \epsilon_n$ are referred to as *slack variable* and allow particular observations to be misclassified. If $\epsilon_i = 0$, the ith is correctly classified, and if $\epsilon_i > 0$, the observation is on the wrong side of the margin and finally if $\epsilon_i > 1$, the observation is on the wrong side of the hyperplane. C measures the level of tolerance of the misclassification, it bounds the ϵ_i's. If $C = 0$ it means there is no tolerance of misclassification and therefore the classifier becomes the maximal margin classifier (Hastie et al., 2013).

Sometimes non linear boundary is required for a classification problem. Expanding feature space, like taking combinations of features $X_1X_2, X_1X_3....X_{p-1}X_p$ or taking the features with their squares $X_1X_1^2, X_2X_2^2.....X_pX_p^2$. These feature space expansions might improve on the linear boundary classifier but they increase features exponentially and can be cumbersome and computationally expensive. A *support vector machine* is an extension of support vector classifier which solves the linear boundary problems by making use for *kernels*. Central to the idea of kernels is the idea of a *dot product* which is mathematically defined as $\langle a, b\rangle = \sum_{i=1}^{r} a_i b_i$. The linear support vector classifier is given as

$$f(x) = \beta_0 + \sum_{i=1}^{n} \alpha_i \langle x, x_i \rangle. \tag{3.37}$$

A linear kernel function K can be given as

$$K(x_i, x_{i'}) = \langle x_i, x_{i'} \rangle = \sum_{i=1}^{n} x_{ij} x_{i'j} \tag{3.38}$$

this gives the support vector classifier with linear decision boundary. The kernel function can be changed slightly to the following

$$K(x_i, x_{i'}) = \left(1 + \sum_{i=1}^{n} x_{ij} x_{i'j}\right)^d. \tag{3.39}$$

This is called a *polynomial kernel* of degree d, when used together with support vector classifier, the resulting classifier is called suport vector machine.

Studies have indicated that support vector machines have significant ability of dealing with high dimensional data because their classification error performance has no direct dependency on input data's dimensionality (Lesniak et al., 2012).

3.3.3 Data Preprocessing

The dataset is a *whole sale customer dataset* from **UCI machine learning repository**. The dataset contains 440 entries with 8 attributes. The first two columns of the data are categorical columns, Channel and Region. There were two channels; Horeca and Retail and three regions; Oporto, Lisbon and Other Region. The original data represented the categorical data using codes, 1="Retail" and 2 = "Horeca", and regions were 1 = "Lisbon", 2 = "Oporto" and 3 = "Other Region". Table 3.6 shows the first 6 rows of the data which is only a portion of the 440 data points.

Table 3.6: First 6 rows showing categorial codes

	Channel	Region	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicassen
1	2	3	12669	9656	7561	214	2674	1338
2	2	3	7057	9810	9568	1762	3293	1776
3	2	3	6353	8808	7684	2405	3516	7844
4	1	3	13265	1196	4221	6404	507	1788
5	2	3	22615	5410	7198	3915	1777	5185
6	2	3	9413	8259	5126	666	1795	1451

Two class predictions were carried out, first being the two-class problem with channel being the dependent variable, and the rest of the variables including region being the independent variables. The second class prediction problem had the region as the dependent variable and therefore channel forming part of the independent variables. In a nutshell, there were two classification problems; a two-class classification problem and multi-class classification problem. This was done in case one approach is more suited to multi-class classification and the other to two-class classification problems. The **hold-out validation** method was employed, where the data was randomly split with 80% of the data used to train the models and the remaining 20% (hold-out sample) used to test classification accuracy. The data for continuous variables were normalized using R's *scale()* function. Having categorical variables in the form of region and channel as independent variables meant that dummy variables needed to be created. Normally if the number of class levels is k, only $k-1$ dummy variables are required. For the two-class problem (Channel), three region dummy variables were created; "RegionLisbon", "RegionOporto", and "RegionOtherRegion". The "RegionOtherRegion" variable was dropped since if it is known that the region is neither "RegionLisbon" nor "RegionOporto", then "RegionOther-Region" is implied. If a record had "Oporto" as a region value, dummy variable "RegionOporto" would have a 1 and the dummy variable "RegionLisbon" would have a 0 and 0 would be implied for "RegionOtherRegion". The same logic applies to a record with "Lisbon" region value. The three-class problem had two dummy variables for channel; "ChannelHoreca" and "ChannelRetail". The dummy variable "ChannelHoreca" was dropped in this case since knowing that the channel is not "horeca" implies that it is "retail". Either of the dummy variable could have been dropped without altering the analysis results. Tables 3.7 and 3.8 show the results of the data transformation (this is only top 6 rows of 440).

Table 3.7: First 6 rows - Region Dummy Variables with Transformed Data - Two-class Problem

x.RegionLisbon	x.RegionOporto	x.Fresh	x.Milk	x.Grocery	x.Frozen	x.Detergents_Paper	x.Delicassen	y
0	0	0.02	0.52	-0.04	-0.60	-0.06	-0.04	Retail
0	0	-0.41	0.54	0.16	-0.25	0.07	0.20	Retail
0	0	-0.47	0.40	-0.03	-0.11	0.11	3.53	Retail
0	0	0.06	-0.61	-0.38	0.78	-0.48	0.20	Horeca
0	0	0.78	-0.05	-0.08	0.23	-0.23	2.07	Retail
0	0	-0.23	0.33	-0.29	-0.50	-0.23	0.02	Retail

Table 3.8: First 6 rows - Channel Dummy Variables with Transformed Data - Multi-class Problem

x.ChannelRetail	x.Fresh	x.Milk	x.Grocery	x.Frozen	x.Detergents_Paper	x.Delicassen	y
1	0.02	0.52	-0.04	-0.60	-0.06	-0.04	Other Region
1	-0.41	0.54	0.16	-0.25	0.07	0.20	Other Region
1	-0.47	0.40	-0.03	-0.11	0.11	3.53	Other Region
0	0.06	-0.61	-0.38	0.78	-0.48	0.20	Other Region
1	0.78	-0.05	-0.08	0.23	-0.23	2.07	Other Region
1	-0.23	0.33	-0.29	-0.50	-0.23	0.02	Other Region

The **data imbalance problem**[15] is an area of active research. In its strict definition, it arises when observations ratio of a majority class to a minority class is in the regions of 100:1, 1 000:1, and 10 000:1. Such over-representation of a majority class results in classification models failing to accurately predict the minority class. The models turn to be bias towards the majority class (He and Garcia, 2009). As stated in Agrawal et al. (2015), in real world classification problems the minority class is usually the class of interest. For example in predicting the presence or absence of a cancer, the focus is mainly on accurately predicting the presence although the overwhelming majority of cases are probably cases of absence. In situations where it is feasible, more data can be gathered which sometimes fixes the imbalance in the data[16]. In cases where that is not feasible, the correction of the imbalance in

[15]Also known as **class imbalance problem** and commonly referred to as **"imbalanced data problem"**.
[16]`https://machinelearningmastery.com/tactics-to-combat-imbalanced-classes-in-your-machine-learning-dataset/`

the data is usually through two random sampling techniques; oversampling and undersampling. In oversampling, the minority class is randomly augmented by synthetic observations which are sampled from existing observations. In undersampling, observations are randomly removed from the majority class to balance the data. He and Garcia (2009) and Agrawal et al. (2015) cite the drawbacks of the two balancing techniques; oversampling may results in overfitting because observations are replicated while undersampling may lead to a loss of essential information. He and Garcia (2009) go further to discuss the "state of the art" techniques which address these drawbacks. Figure 3.9 shows the class distribution of both the two-class (Channel) and the three-class (Region) problems.

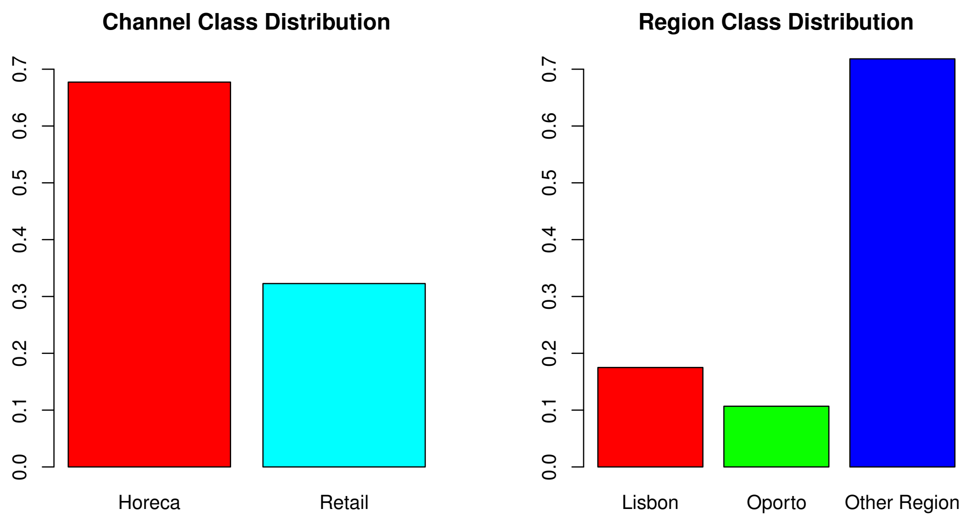


Figure 3.9: Class Distribution

The classes are unbalanced in both cases but the imbalances are nowhere near the ratios that define the data imbalance problem, although in the three-class problem the imbalance looks quite sizeable. The R package **scutr**[17] was used for transforming the data into balanced data where necessary. The package uses SMOTE and cluster-based undersampling technique (SCUT) which combines oversampling and undersampling. Synthetic minority oversampling technique (SMOTE) is the oversampling part of the technique.

3.3.4 Data Analysis and Results

3.3.4.1 Two-Class Classification

3.3.4.1.1 Analysis

Figure B.2 in appendix B.1.2 shows pairwise scatter plots of the predictor variables for two-class classification problem. The figure shows a marked linear relationship between Grocery and Paper-Detergents, Grocery and Milk, as well as Milk and Paper-Detergents. The marked relationship is evident in the correlation matrix of Table 3.9. The correlation coefficients between the pairs Grocery and Paper-Detergents was 92%,Grocery and Milk was 74% and Milk and Paper-Detergents was 66%. Evidently, these pairs would have been good candidates for dimension reduction if a

[17]`https://cran.r-project.org/web/packages/scutr/index.html`

technique like PCA was performed, but there was no dimension reduction exercise undertaken in this research since the data was no where near high-dimensional.

Table 3.9: Correlation Matrix for the Two Classes

	x.RegionLisbon	x.RegionOporto	x.Fresh	x.Milk	x.Grocery	x.Frozen	x.Detergents_Paper	x.Delicassen
x.RegionLisbon	1.00	-0.16	-0.03	-0.02	-0.03	-0.01	-0.02	-0.03
x.RegionOporto	-0.16	1.00	-0.06	-0.03	0.05	0.07	0.06	-0.04
x.Fresh	-0.03	-0.06	1.00	0.10	-0.01	0.35	-0.10	0.24
x.Milk	-0.02	-0.03	0.10	1.00	0.73	0.12	0.66	0.41
x.Grocery	-0.03	0.05	-0.01	0.73	1.00	-0.04	0.92	0.21
x.Frozen	-0.01	0.07	0.35	0.12	-0.04	1.00	-0.13	0.39
x.Detergents_Paper	-0.02	0.06	-0.10	0.66	0.92	-0.13	1.00	0.07
x.Delicassen	-0.03	-0.04	0.24	0.41	0.21	0.39	0.07	1.00

LDA assumptions for homogeneity of variance-covariance matrices and normality were tested. The results of Box M-test for homogeneity of covariance matrices is shown in Table 3.10. Based on the p-value < 0.05, the null hypothesis that the observed covariance matrices for the dependent variables are equal across groups was rejected. Because Box M-test is very sensitive and picks up the slightest deviations, another test which is less stringent called Levene's test was also performed and the output is shown in Table 3.11. The Levene's test is a univariate test for equality of variances between the groups for each variable. The equality of variances was also rejected for all the variables at 0.05 level of significance. This meant that the data did not satisfy the variance-covariance homogeneity assumption required for the LDA.

Table 3.10: Box M Test for Channel

	DF	X-squared	p-value
Channel	42	1016.5	2.2e-16

Table 3.11: Levene's Test for Homogeneity of Variance - Channel

	Milk	Fresh	Grocery	Frozen	Detergents_Paper	Delicassen
F-value	27.386	6.9219	63.231	16.023	111.93	0.243
$Pr(>F)$	2.59e-07	0.008815	1.579e-14	7.348e-05	2.2e-16	0.6223

The Shapiro-Wilk test for normality was carried out for each class and each variable and the summary of the results is shown in Table 3.12. At 0.05 level of significance the null hypothesis that the independent variables are normally distributed within each group is rejected as all the p-values are less than 0.05. Figure B.1 shows a Q-Q plot of the Horeca group and it can be seen that the points are generally deviating away from the straight line which is indicative of absence of normal distribution. The Q-Q plot for the Retail group was similar. It could be concluded then that the data did not satisfy the LDA assumptions.

The *partimat()* function from R's **klaR** package gives a quick lda classification between any two variables accompanied by classification errors as shown in figure B.2 of appendix B.1.2.

3.3.4.1.2 Model Fitting

The LDA model for the two classes was fitted on the training set using *lda()* function from R's **MASS** package and results are shown in in Table B.2 of appendix B.1.4.1. The results also show

Table 3.12: Shapiro-Wilk normality test - Channel

	Milk	Fresh	Grocery	Frozen	Detergents_Paper	Delicassen
Horeca						
W	0.78416	0.58498	0.78283	0.55875	0.64586	0.2889
p-value	2.2e-16	2.2e-16	2.2e-16	2.2e-16	2.2e-16	2.2e-16
Retail						
W	0.84045	0.66795	0.73141	0.74405	0.75578	0.68838
p-value	4.081e-11	2.2e-16	8.272e-15	1.924e-14	4.329e-14	5.695e-16

the group prior probabilities of 0.677 for "Horeca" and 0.323 for "Retail". The model output also shows group means as well as the DF (in the form of linear discriminant coefficients). Figure 3.10 shows the plots of the DF, illustrating how well the function separates the two groups. Although the two plots overlap a bit but it is clear that the function does separate the groups quite well (with "Retail" group right skewed and "Horeca" left skewed). For quantitative **model validation**, the discriminating power of a DF can be tested for significance using Wilk's lambda. Eigenvalues also give a good indication of such discriminating power. Schlegel (2016) illustrated how to calculate these statistics using R-programming. But, SPSS statistics software package outputs these and other statistics with minimal to no programming. So, SPSS was preferred over R for the significance testing of the DF and Figure B.3 shows an extract of the SPSS output. Bhaumik (2020) described Wilk's lambda as assessing discrimination power of the independent variables. When the statistic is significant, the null hypotheses of "No group separability" is rejected and the conclusion is that the DF is significant. The Wilk's lambda statistic in Figure B.3 is significant ($p - value < 0.01$), this is in line with the visual representation in Figure 3.10 [18].

In Bhaumik (2020), an eigenvalue is defined as the ratio of the between to within group sum of squares, and a higher eigenvalue indicates a strong discriminating power of the DF. The two group problem has only one eigenvalue (since one DF), and it is a relatively high ratio (0.792) which again signifies a high discriminating power of the DF.

SVM was also fitted on the training set using R's **e1071**[19] package, the model parameters can be found in Table B.4 of appendix B.1.5.1 . The package has a *tune()* function which helps in choosing the best model given a list of possible cost values as a well a list of gamma values. The *tune()* function also takes in the kernel type as an argument, in this research a linear kernel was chosen. The resulting SVM model had 76 support vectors and a cost of 1. Cost is "cost of constraints violations" and indicates the level of tolerance of error. A low cost value results in wider margin (Figure 3.8) and tolerates more misclassifications. A high cost value results in narrower margins and has less tolerance for misclassification.

3.3.4.1.3 Results

The hold-out sample was input into the LDA model and a confusion matrix for validating the classification accuracy was constructed, Table 3.13 shows the confusion matrix. The table shows a total of 59 "Horeca" entries were in the test data, 57 of them correctly predicted to be "Horeca" while 2 were misclassified as "Retail". A total of 28 "Retail" entries in the test data, 11 of them were misclassified as being "Horeca" while 17 were correctly classified. This gave $\frac{57+17}{57+11+2+17} = 0.85058 \Rightarrow 85.051\%$

[18]Note is this despite R and SPSS discriminant coefficients being different, the difference could be due to standardisation and scaling.

[19](Meyer et al., 2021)

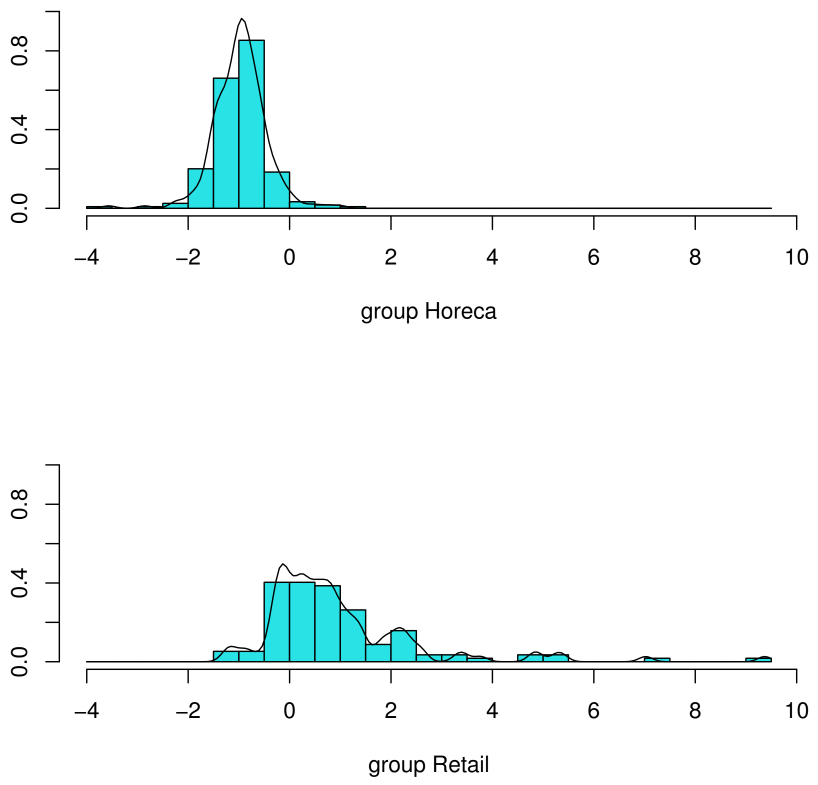


Figure 3.10: Discrimination Plots for the Two Group

prediction accuracy.

Table 3.13: Two-Class LDA confusion matrix

		Actual	
		Horeca	Retail
predicted	Horeca	57	11
	Retail	2	17

The hold-out sample was again input into the SVM model and another confusion matrix was drawn. Table 3.14 shows the confusion matrix. Following the same logic explained in the LDA case above, the classification accuracy of the SVM was 88.506%.

Table 3.14: Two-Class SVM confusion matrix

		Actual	
		Horeca	Retail
predicted	Horeca	54	5
	Retail	5	23

The LDA model predicted "Horeca" (majority class) 78.161% [20] of the time, this may indicate a bias to some extent towards the majority class since the majority class had a prior probability of only 0.677. The SVM model on the other hand predicted "Horeca" 67.816% of the time which was

[20] $\left(\frac{57+11}{57+11+2+17}\right)$

consistent with the prior probability. It should be noted that the LDA model did not fail to predict the minority class, the issue is that its prediction generally seems 'disproportionately' favourable to the majority class. This apparent bias requires other tools of performance measurement other than overall prediction accuracy, as mentioned in (He and Garcia, 2009), accuracy is sensitive to changes in the data and is not suitable for unbalanced data. He and Garcia (2009) discuss other measures of model performance, among them; *precision*, *recall*, and *F-Measure*. This research was confined to the three measures mentioned here (including *accuracy*), this is because they are intuitive and sufficient for the purposes of the aimed performance comparisons. The mathematical definitions of the three measures are shown in Appendix B.1.3. Positive and negative observations here follow the same convention that is followed in He and Garcia (2009), where positive is the minority class and negative is the majority class. *Precision* measures the ratio of the accurately predicted positive observations to the total number of predicted positive observations. It indicates the number of observations that are labelled correctly out of all the positive labelled observations. *Recall* measures the ratio of accurately predicted positive observations to the total number of the actual positive observations. Predicting almost all observations as positive results into an almost perfect *recall* value (almost no False Negative (FN)'s), and predicting almost all cases as negative results into an almost perfect precision *precision* value (almost no False Positive (FP)'s). *F-Measure* is a weighted mean of the two measures, it penalises extreme values and it reaches maximum value when both measures are a 100% each, and it is 0 when either of them is 0. So a higher *F-Measure* value is preferred. The functions *precision()*, *recall()*, and *F-meas()* from the R package **caret** were used to calculate the three measures. They all require that the actual and predicted datasets be specified, as well as the positive class label. Taking "Retail"(minority class) as a positive class, the LDA model had a *precision*, *recall*, and an *F-Measure* of 89.4737%, 60.7143% and 72.3404% respectively. The SVM model had 82.1429% for each of the three measures. Therefore, the SVM model was a better model for the minority class as measured by *F-Measure* and it had better overall accuracy.

Further model performance checks were carried on the balanced data. The balancing was done on the training dataset using *SCUT()* function from the **scutr** package. The function takes in the dataset to be balanced as well as a clustering function which can be either K-means, hierarchical or any user defined clustering methodology. The test dataset was left unchanged (unbalanced), this is to ensure that the models are tested on real data. The function works out the number of observations in balanced dataset as follows; if m is the number of observations in the unbalanced dataset and n is the number of classes, then m/n is the number of observations per class in the resulting dataset. Table 3.15 shows the class distribution before and after balancing.

Table 3.15: Balanced Data Class Distribution

	Horeca	Retail
Training Unbalanced Dataset	239	114
Training Balanced Dataset	**176**	**176**
Test Dataset	59	28

Table B.3 and Figure B.4 show the results of the LDA model fitting from R and SPSS respectively. The prior probability for each class is 0.5 which signifies a balanced dataset (the prior probabilities on lda() function defaults to class proportions). Again the null hypothesis of "No group separability" is rejected which indicates that the classes are still separable. Table 3.16 shows the resulting confusion matrix.

Table 3.16: Two-Class LDA confusion matrix - Balanced Data

		Actual	
		Retail	Horeca
predicted	Retail	27	25
	Horeca	1	34

The LDA model had a *precicion* of 51.9231%, a *recall* of 96.4286%, as well as an *F-Measure* of 67.5000%. The LDA balanced data model did not perform favourably compared to the unbalanced data model according to the *F-Measure*. The model's overall accuracy of 70.110% was lower compared to that of the model on the unbalanced dataset.

Table 3.17: Two-Class SVM confusion matrix - Balanced Data

		Actual	
		Retail	Horeca
predicted	Retail	27	15
	Horeca	1	44

An SVM model was also fitted to the balanced data for comparison, Table B.5 shows the model output and Table 3.17 its confusion matrix. The values for *precision*, *recall*, and *F-Measure* were 64.2857%, 96.4286%, and 77.1429%, respectively. Again the *F-Measure* value was no better than in the corresponding unbalanced data model. This meant neither of the two techniques benefited from data balancing. The SVM model again outperformed the LDA when compared by both *F-Measure* and accuracy.

It is worth noting that the balancing techniques applied do not produce the same balanced data points on every execution, each execution might have a unique dataset which can result in different performance metrics, this is because random sampling is applied in SCUT. But, the results in tables 3.17 and 3.16 were the most prominent between the different executions.

Since the original data was not normally distributed within the groups as shown through the significance tests, and the problem is a binary classification problem, a logistic regression model was also fitted to the data. The model output is shown in Table B.6 of Appendix B.1.6. The confusion matrix of the logistic regression model is shown in Table 3.18. The model achieved the same level of classification accuracy as the SVM model and therefore outperformed the LDA model.

Table 3.18: Two-Class Logistic Regression confusion matrix

		Actual	
		Horeca	Retail
predicted	Horeca	55	6
	Retail	4	22

3.3.4.2 Three-Class Classification

3.3.4.2.1 Analysis

Table 3.19 shows the correlation matrix of the three-class data, and the pairwise scatter plot is shown in figure B.6 of appendix B.2.2. The correlation coefficients of the continuous variables remained unchanged as expected. The dummy variables, "ChannelHoreca" and "ChannelRetail", showed perfect negative correlation as expected again since when one has a value 1 the other must have a value 0 and vice versa.

Table 3.19: Correlation Matrix for the Thee-Class classification

	x.ChannelRetail	x.Fresh	x.Milk	x.Grocery	x.Frozen	x.Detergents_Paper	x.Delicassen
x.ChannelRetail	1.00	-0.17	0.46	0.61	-0.20	0.64	0.06
x.Fresh	-0.17	1.00	0.10	-0.01	0.35	-0.10	0.24
x.Milk	0.46	0.10	1.00	0.73	0.12	0.66	0.41
x.Grocery	0.61	-0.01	0.73	1.00	-0.04	0.92	0.21
x.Frozen	-0.20	0.35	0.12	-0.04	1.00	-0.13	0.39
x.Detergents_Paper	0.64	-0.10	0.66	0.92	-0.13	1.00	0.07
x.Delicassen	0.06	0.24	0.41	0.21	0.39	0.07	1.00

LDA assumptions were also tested for this multi-class scenario. The Box-M test showed rejection of the null hypothesis of homogeneity of covariance matrices as shown in Table 3.20. The less stringent Levene's test showed that the null hypothesis of equal variances among the groups for the independent variables cannot be rejected as all the p-values were greater than 0.05. The output of the p-values for the Levene's test are in Table 3.21. The normality assumption was rejected for each group in all the independent variables as shown in Table 3.22. Figure B.5 shows the graphical evidence of the lack of normality (other "region" plots were similar).

Table 3.20: Box M Test for Region

	DF	X-squared	p-value
Region	21	374.91	2.2e-16

Table 3.21: Levene's Test for Homogeneity of Variance - Region

	Milk	Fresh	Grocery	Frozen	Detergents_Paper	Delicassen
F-value	0.1363	1.4408	0.103	1.1706	0.863	0.9968
$Pr(> F)$	0.8726	0.2379	0.9021	0.3112	0.4226	0.3699

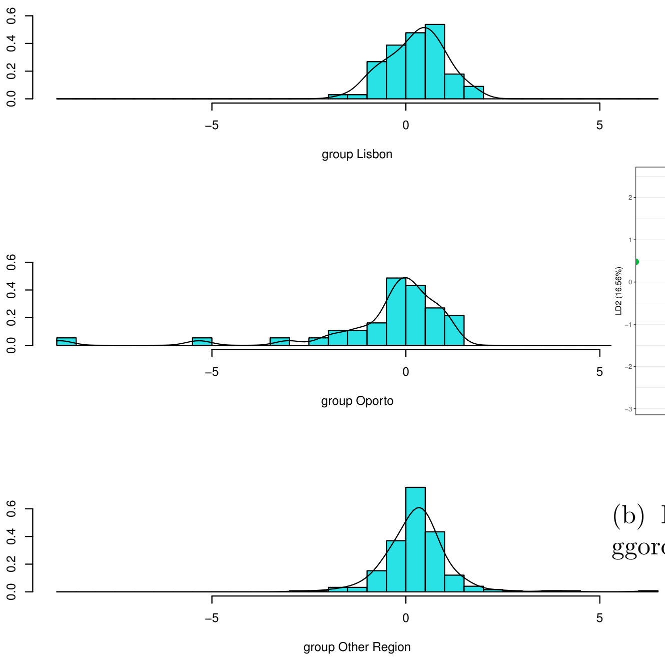
3.3.4.2.2 Model Fitting

Just like in the two-class classification problem, a randomly selected training set (80% of the data) was used to fit the model using the *lda()* function. This is consistent with the hold-out validation method. Table B.9 shows the details of the resulting model. In the three group problem there are two DF's, LD1 and LD2. The proportion of trace in the table indicates that LD1 accounts for 83.44% of the discrimination, and the rest is accounted for by LD2. The "Other Region" group has prior probability of 0.7054 with "Lisbon" and "Oporto" having 0.1898 and 0.1048 respectively. Figures 3.11a and 3.11b show the visual representation of the strength of discrimination. The histograms on Figure 3.11a seem to be centred about the same point, no clear pattern of separation between the groups. Figure 3.11b is a plot of LD2 against LD1, again the points are scattered about the same area with no visible pattern of separation. SPPS output of the three group model in Figure B.7 confirms the R visual representation of the discrimination, the DF's are not significant (p-values of 0.183 and 0.111). Therefore, the null hypothesis of the Wilk's lambda test "No group separability" cannot be rejected. However, Manly and Navarro-Alberto (2016) warned that the tests of significance

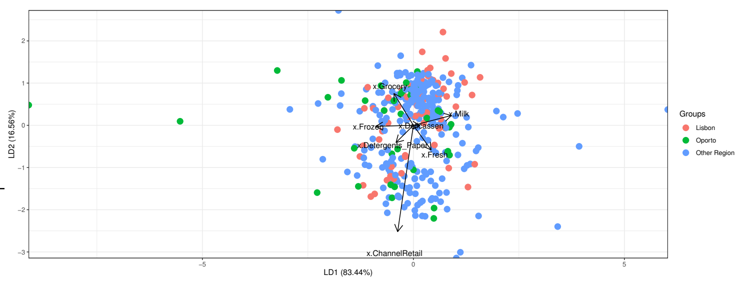
Table 3.22: Shapiro-Wilk normality test - Region

	Milk	Fresh	Grocery	Frozen	Detergents_Paper	Delicassen
Other Region						
W	0.7753	0.59282	0.66529	0.61061	0.60805	0.34567
p-value	2.2e-16	2.2e-16	2.2e-16	2.2e-16	2.2e-16	2.2e-16
Lisbon						
W	0.78718	0.79487	0.73995	0.7776	0.64912	0.78988
p-value	3.428e-09	5.516e-09	2.293e-10	1.923e-09	2.836e-12	4.044e-09
Oporto						
W	0.9086	0.733	0.63459	0.38114	0.57874	0.80593
p-value	0.00137	6.705e-08	1.426e-09	8.362e-13	2.158e-10	2.161e-06

of DF's should not be relied upon in deciding how many DF's should be kept to represent the actual group differences. The reason as they put it is that the j^{th} DF in the population may not be the j^{th} DF in the sample, owing to sample errors. They suggest that alternative ways of validating the nature of group differences should be explored. The eigenvalues in the figure indicate that DF 1 has a relatively stronger discrimination power (larger eigenvalue) compared to DF 2, which is in line with what was shown in R's proportion of trace output.



(a) Discrimination Plots for the Three Group using ggord R-core



(b) Discrimination Plots for the Three Group using ggord package

Figure 3.11: Discrimination Plots for the Three Group

3.3.4.2.3 Results

The hold-out sample was used to test the prediction accuracy of the fitted LDA model and a confusion matrix was constructed to validate the model's predictive strength. The confusion matrix of the LDA in Table 3.23 shows that the LDA model predicted 0 entries for "Lisbon" when in fact the actual is 10 occurrences, predicted 2 occurrences for "Orporto" when in fact there was 10 occurrences, and lastly predicted 65 occurrences for “OtherRegion” when in fact there was 67 occurrences. This translated to 74.713% classification accuracy.

Table 3.23: Three-Class LDA confusion matrix

		Actual		
		Lisbon	Oporto	Other Region
predicted	Lisbon	0	0	0
	Oporto	0	0	2
	Other Region	10	10	65

Similarly, the hold-out sample was used to test prediction accuracy of the SVM model. The confusion matrix for the SVM model is shown in Table 3.24. It shows misclassifications of “Lisbon” and “Oporto” but unlike the LDA model all occurrences of “OtherRegion” were correctly classified. This translated to 77.012% classification accuracy.

Table 3.24: Three-Class SVM confusion matrix

		actual		
		Lisbon	Oporto	Other Region
predicted	Lisbon	0	0	0
	Oporto	0	0	0
	Other Region	10	10	67

A closer look at the confusion matrices reveals that none of the minority classes (“Lisbon” and “Oporto”) were accurately predicted by both models, the LDA model inaccurately predicted 2 entries for “Oporto”. The SVM model predicted “Other Region” class 100% of the time while LDA predicted it 98%[21] of the time. It can reasonably be concluded that both models suffered from the **data imbalance problem**, this is because the majority class was disproportionately favoured and the models failed to predict the minority classes. This led to each of the minority classes having a 0 for *precision*, *recall*, as well as for *F-Measure*. This clearly was an anomaly in the prediction models despite them having relatively high overall prediction accuracies, this is called accuracy paradox (Uddin, 2019). The *SCUT()* function was again used to balance the data. Table 3.25 shows the class distribution for the resulting balanced dataset.

Table 3.25: Balanced Data Class Distribution - Region

	Lisbon	Oporto	Other Region
Training Unbalanced Dataset	62	38	253
Training Balanced Dataset	118	118	118
Testing Training Dataset	15	9	63

Table B.10 and Figure B.8 show the fitted LDA model. The prior probability for each of the three classes is 0.3333. The null hypothesis of “No group separability” is rejected, this means the classes are this time separable. The *ml_test()* function from the **mltest** R package was used to compute the model performance metrics because it can handle multi-class classification which is not the case with the functions in **caret** which are meant for binary classification (**caret** was used in the two-class problem). Table 3.26 is the confusion matrix for the LDA model while Table 3.27 is its performance measures, the latter table shows performance measures when each of the classes is considered as a

[21] $\frac{10+10+65}{10+10+67}$

positive class. Evidently, there is a vast improvement in the minority classes prediction by LDA as measured by *precision*, *recall*, and *F-Measure*. In a nutshell, the LDA model predicted accurately 6 out of the possible 9 for "Oporto" and 10 out of the possible 15 for "Lisbon". This is a high *recall* rate compared to the previous LDA model. The harmonising effect of the *F-Measure* can be seen when the majority class ("Other Region") is taken as a positive class. It has a *precision* of 91.6667% but an *F-Measure* of only 29.3333%, this is owed to the low *recall* of 17.4603%.

Table 3.26: Three-Class LDA confusion matrix - Balanced Data

		Actual		
		Oporto	Lisbon	Other Region
predicted	Oporto	6	5	19
	Lisbon	2	10	23
	Other Region	1	0	11

Table 3.27: Three-Class LDA Performance measures - Balanced Data

	Oporto	Lisbon	Other Region
recall	66.6667%	66.6667%	17.4603%
precision	20.0000%	22.2222%	91.6667%
F-Measure	30.7692%	33.3333%	29.3333%
accuracy	31.0345%		

Similarly, the results for the SVM model are shown in Table 3.28 and Table 3.29. Its performance metrics show quite lower numbers on the minority classes compared to the ones achieved by the LDA model, although it has a higher accuracy.

Table 3.28: Three-Class SVM confusion matrix - Balanced Data

		actual		
		Oporto	Lisbon	Other Region
predicted	Oporto	3	6	23
	Lisbo	2	4	17
	Other Region	4	5	23

Although LDA outperformed SVM on the balanced data when measured by *F-Measure* but the results are generally not flattering, the performance measurements are quite low for both techniques.

Other variants of discriminant analysis were also checked against the three-class scenario on unbalanced data as shown in Table B.11 of Appendix B.2.5. QDA achieved the same accuracy as LDA, multiple discriminant analysis (MDA) had 73.6% accuracy with HDA having the lowest accuracy of 70.1%. The HDA was not really expected to improve the three-class scenario as the equality of variance could not be rejected by Levene's test in Table 3.21.

3.3.4.2.4 Comparing prediction accuracy when normality assumptions are satisfied - Iris Dataset

The Iris dataset was first used in Fisher (1936), and is widely used in the statistics and machine learning training programs. The data is a multi-class with dependent variable, Species, having three groups "setosa", "versicolor", and "virginica". The data does satisfy normality assumption

Table 3.29: Three-Class SVM Performance measures - Balanced Data

	Oporto	Lisbon	Other Region
recall	33.3333%	26.6667%	36.5079%
precision	9.3750%	17.3913%	71.8750%
F-Measure	14.6342%	21.0526%	48.4211%
accuracy	34.4800%		

as shown in Figure B.9 and Table B.13. The interest in the data was to see how would LDA compare to SVM in a case where normality assumptions are satisfied. The homogeneity of covariance matrices was still not satisfied according to the Box M-test and only two variables satisfied the group variance equality assumption test in Table B.13.

Table B.15 in Appendix B.2.6.2 shows the LDA fitted model on Iris dataset. Table 3.30 is the confusion matrix which indicates 98% classification accuracy. The classification accuracy of the SVM model on the Iris dataset is shown in Table 3.31 and is 96.667%.

The LDA model performed better than the SVM default model.

SVM allows for the tuning of the model parameters such as the kernel, gamma and cost value. After tuning the model by changing the cost from 0.1 to 100, the tuned SVM model achieved 100% classification accuracy. However, this was not always reproducible as at times the fitting seemed to go on non-stop which is usually an indication that the algorithm fails to converge. This could be due to physical memory limitations.

The code listing for the analysis in this case study is in Appendix B.3.1.

Table 3.30: Three-Class LDA confusion matrix - Iris Dataset

		Actual		
		setosa	versicolor	virginica
predicted	setosa	50	0	0
	versicolor	0	48	1
	virginica	0	2	49

Table 3.31: Three-Class SVM confusion matrix - Iris Dataset

		Actual		
		setosa	versicolor	virginica
predicted	setosa	49	0	0
	versicolor	0	49	3
	virginica	1	1	47

3.3.5 Discussion and Conclusion

3.3.5.1 Discussion

The Wilk's lambda test examines the null hypothesis of "No group separability". On the unbalanced data, the null hypothesis was rejected in the two group classification but could not be rejected in the three group classification for both DF's . This was an indication that the DF's were not doing a good job in separating the three groups. But, Manly and Navarro-Alberto (2016) warn against over reliance on such statistical significance tests citing their frailties, so both DF's were kept in the model. Moreover, a rework of the thee-class LDA model would have been required if the two DF's were to be dropped since they were the only DF's in the model. The next step was testing for prediction accuracy on the hold-out sample, the results showed that the SVM model outperformed the LDA model in the two-class scenario and the LDA model was again outperformed in the three-class scenario. The LDA performance on the customer data was expected to be suboptimal given that the accompanying assumptions were not satisfied. The three-class problem had a lower prediction accuracy in both LDA and SVM compared to the two-class problem. This is an interesting fact because two-class LDA model's DF was significant and the two DF's of the three-class scenario were not. This suggests that Wilk's lambda did give an accurate indication of poor separability in the three-class problem. This is a valuable information whether one is using LDA or SVM for class prediction. It also suggests that the two techniques can be used to work hand in hand. The LDA can be used to gauge the level of separability using the Wilk's lambda test, and the SVM can be used for its superior class prediction accuracy. As was discussed in Section 2.2.2, Vinay et al. (2015) did use the two techniques in combination with LDA being used for dimension reduction and SVM for classification.

Balancing the data did not benefit the two-class problem for both modelling techniques, in fact, the models fitted on balanced data fared poorly compared to the original models. This showed that class balancing may not always be the solution to the 'perceived' biases when data is unbalanced. The accuracy measurements on the initial three-class modelling hid a crucial anomaly about the fitted models, namely, the data imbalance problem. There was not a single accurately predicted minority class observation although the imbalance on the proportions did not match the definition of the data imbalance problem. Synthetic data balancing was warranted for the three-class problem due to the lack of feasibility of gathering more data. The LDA performed better on the minority classes according to *F-Measure* but was poorer on overall accuracy. Such results are open to debate and interpretation. The acceptable measure would be guided by the objectives of the classification problem. As mentioned before, if the objectives are for detection of presence or absence of a particular disease (e.g. cancer), the overall accuracy is no use if it comes at the expense of mislabelling the presence cases as absence cases. In marketing, one likely needs a model that is as accurate as possible, although it is not a life-and-death issue, but it wastes large amounts of money and effort to market products to wrong groups. Regardless of which performance metric was under consideration, the metrics showed poor performance (all be below 35%) by both models on the balanced three-class problem. This was probably an indication of more underlying issues which required further interventions. But, the analysis and the results were sufficient for the basis of comparison of the two modelling techniques.

The two-class analysis on unbalanced data was extended to include logistic regression to ascertain whether the results were due to normality assumptions. The problem was binary classification and the data not normally distributed, conditions in which logistic regression usually triumphs over LDA. So, had the logistic regression model performed poorer than the LDA model that would have suggested some other underlying cause of the performance of LDA against SVM. The logistic regression accuracy was at par with SVM and therefore also outperformed the LDA model. The multi-class analysis was extended to the Iris dataset which satisfies some of the LDA assumptions. The LDA outperformed SVM on the initial (default) configuration which showed there was a benefit in using LDA when the assumptions are met. Further tuning and changing on model parameters of the SVM saw the SVM outperforming LDA although these introduced instability in the SVM model.

Interpretability of the models was judged based on the model summary outputs from the respective packages used, appendices B.1.5.1 and B.2.3.1 are outputs of the two and three classes SVM models respectively, while appendices B.1.4.1 and B.2.4.1 are outputs for the LDA models. The LDA model outputs give more insights into the data in terms of discriminant coefficients, and groups prior probabilities and group means. This information can be easily mapped to equations (3.29) or (3.31) in order to find a posterior probability of any arbitrary new observation with same features as the original data. The outputs of the SVM models show the number of support vectors, SVM type and kernel type used. It is difficult to interpret the SVM model compared to the LDA model based on the model information output.

The results are consistent with the surveyed studies in Section 2.2.2 where SVM performance was slightly better or below that of LDA depending on the kernels used in the SVM. There was a room to explore the various kernels for the SVM in this research, as it was the case in some of the previous studies surveyed. The linear kernel gave better results than the polynomial kernel, the equations for the two kernels were shown in equations (3.38) and (3.39) . Varying the kernels can lead to heavy fluctuations in performance, and can add extra complexity which sometimes may not necessarily provide better performance as was also evident in Shao et al. (2015).

3.3.5.2 Conclusion

The SVM model outperformed the LDA model in both the two-class and three-class scenarios when measured by *accuracy.* But, it fared poorly when measured by *F-Measure* on the minority classes for the three-class problem. The case study again highlighted the necessity and usefulness of statistical significance testing. The gain in performance of LDA when used in the data which obeyed normality assumptions was apparent, the opposite was also apparent when the assumptions were violated. Through significance testing other questions about the data could be answered like homogeneity of variance-covariance matrices which play a major role in volatility modelling. Cox (2001) also accurately states that statisticians often have to respond to issues of causality and not just predictive accuracy. This point was made even more clearer by the accuracy paradox which was caused by the data imbalance problem on the three-class problem. There are a lot of causal questions that can be answered through various statistical significance tests. It was clear again that significance testing is not something that someone does while thinking what to do next as claimed by Mallows (2006), but an essential part of data modelling. Another takeaway from the case study was that a data analyst should not retrofit data to a modelling technique but choose the best technique based on the data characteristics. It was clear that LDA performs better when its assumptions hold but SVM (even logistic regression) should be applied when such assumptions are violated in classification problems.

3.4 K-Means versus Gaussian Mixture Models Case Study

3.4.1 Introduction

Cluster analysis is a form of unsupervised learning, again following the analogy in Hastie et al. (2001), it is learning without a teacher. It tries to discover groups of homogeneous data points in the data, but unlike in classification, the true group memberships have no prior labels, hence it is said to be unsupervised. The principle is that the data points in the same group should be as similar as possible while data points from different groups should be as a different as possible(Grün, 2018).

Clustering mainly deals with structure detection in a data set, whether it be for data summarisation and simplification or for discovering latent heterogeneous groups to reveal patterns and behaviour. Since its nature is exploratory, the problem to be solved is usually not well defined. Grün (2018) mentioned uncovering of *interesting* and *useful* patterns in the data as the objective but acknowledges the difficulty of defining what is *interesting* and what is *useful*. The author further suggest that a true cluster only depends on the context, and that given a data set there is no distinct clustering solution but different aims would arrive at different clusters, therefore, a clustering solution should be accompanied by clear statement of the aims. The domain experts are usually roped in defining clustering requirements but as mentioned in Grün (2018), these experts are normally unable to make meaningful contribution beforehand, only when some clustering has been performed.

Grün (2018) discussed clustering in terms of two broad categories, heuristic clustering and model based clustering. Her study looked at various heuristic clustering methods as they relate to Gaussian mixture models. The goal of this case study was to compare K-means clustering algorithm to GMM in uncovering clusters of unlabelled retail data. The following subsections go into details of the theoretical definitions of the models as well as experiment design.

3.4.2 Clustering Theoretical Framework

3.4.2.1 GMM Clustering

Mixture model cluster analysis is one of many forms of model based approach to cluster analysis. A probability distribution is assumed to be the process that generated the data in each of the partitions (clusters) of the observed data. The objective then is to separate data into clusters by estimating their model parameters. The data as a whole is assumed to have been drawn from a model of a mixture of k distributions with each distribution representing a cluster (Franzén, 2008).

In Lui (2010) a mixture model is defined as follows:

$$g(x) = \sum_{j=1}^{k} \pi_j f_j(x) \tag{3.40}$$

where $0 \leq \pi_j \leq 1, \sum_{j=1}^{k} \pi_j = 1, j = 1, ..., k, k > 1$. The $g(.)$ is called a mixture density function, X is said to have a mixture distribution where $\pi_1, ..., \pi_k$ are mixing coefficients or weights and $f_1(x), ..., f_k(x)$ are component densities of the mixture. If $f_i(.)$ is parametric with parameters θ_j not known, then the model becomes

$$g(x|\Psi) = \sum_{j=1}^{K} \pi_j f_j(x|\theta_j) \tag{3.41}$$

where (Ψ) is a distinct collection of all parameters occurring in the mixture (Lui, 2010). The main objective of the mixture model clustering is to estimate (Ψ).

In a GMM the data is assumed to be multi-modal and generated from a mixture of normal distributions. EM algorithm is a form of maximum likelihood estimation which is used when a closed form solution cannot be found using the normal MLE. Do and Batzoglou (2008) go into a step by step illustration of the EM with a numerical example using coin tosses, while Hastie et al. (2001) offer comprehensive theoretical underpinnings of the GMM and EM. The following notes are an extract from Bishop (2013) and provide a somewhat concise rational for the EM algorithm in a multivariate setting. The appendices (C.1.1), (C.1.2) and (C.1.3) are extracts from the notes of Bonakdarpour (2016), and compliment Bishop (2013) but in a univariate setting.

The Gaussian mixture distribution for K mixture components can be written as

$$p(\mathbf{x}) = \sum_{k=1}^{K} \pi_k N(\mathbf{x}|\mu_k, \mathbf{\Sigma}_k). \tag{3.42}$$

The marginal distribution of $\mathbf{z}$ is expressed as

$$p(z_k = 1) = \pi_k. \tag{3.43}$$

Let the vector $\mathbf{z}$ be a K-dimensional binary random variable having one-hot encoding (1-of-K) such that if one element Z_k has a value of 1, all the other elements have value of 0. That translates to $z_k \in \{0, 1\}$ and $\sum_k z_k = 1$. The variable $\mathbf{z}$ is not observed, and therefore it is called a latent variable.

Because of the one-hot encoding of $\mathbf{z}$, its distribution can be written as

$$p(\mathbf{z}) = \prod_{k=1}^{K} \pi_k^{z_k}, \tag{3.44}$$

and the conditional distribution of $\mathbf{x}$ given specific value of $\mathbf{z}$

$$p(\mathbf{z}|z_k = 1) = N(\mathbf{x}|\boldsymbol{\mu}_k, \mathbf{\Sigma}_k), \tag{3.45}$$

and $\mathbf{x}$ given $\mathbf{z}$

$$p(\boldsymbol{x}|\mathbf{z}) = \prod_{k=1}^{K} N(\mathbf{x}|\boldsymbol{\mu}_k, \mathbf{\Sigma}_k)^{z_k}. \tag{3.46}$$

Using equations (3.44) and (3.46) the marginal distribution of $\mathbf{x}$ is obtained by summing the joint distribution over all possible states of $\mathbf{z}$

$$p(\mathbf{x}) = \sum_{z} p(\mathbf{z})p(\mathbf{x}|\boldsymbol{z}) = \sum_{k=1}^{K} \pi_k N(\mathbf{x}|\mu_k, \mathbf{\Sigma}_k). \tag{3.47}$$

The representation of the marginal distribution of $\mathbf{x}$ as $p(\mathbf{x})$ ensures that for every $\mathbf{x}_n$ there is a corresponding latent variable $\mathbf{z}_n$, for $\mathbf{x}_1, \ldots, \mathbf{x}_N$.

Another important expression in the formulation of the EM algorithm is of $\gamma(z_k)$ which represents $p(z_k = 1|\mathbf{x})$ and it is derived by using Baye's theorem,

$$\begin{aligned} \gamma(z_k) \equiv p(z_k = 1|\mathbf{x}) &= \frac{p(z_k = 1)p(\mathbf{x}|z_k = 1)}{\sum_{j=1}^{K} p(z_j = 1)p(\mathbf{x}|z_j = 1)} && (3.48) \\ &= \frac{\pi_k N(\mathbf{x}|\boldsymbol{\mu}_k, \mathbf{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j N(\mathbf{x}|\boldsymbol{\mu}_j, \mathbf{\Sigma}_j)} && (3.49) \end{aligned}$$

π_k is viewed as the prior probability of $z_k = 1$, and the quantity $\gamma(z_k)$ as the posterior probability when $\mathbf{x}$ is observed (Bishop, 2013).

For a Gaussian mixture, the observations $\mathbf{x}_1, \ldots, \mathbf{x}_N$ can be represented with an $N \times D$ matrix $\mathbf{X}$ with the n^{th} row being x_n^T with the corresponding $N \times K$ matrix $\mathbf{Z}$ having rows $\mathbf{z}_n^T$. Assuming the observations are independently drawn from the distribution, likelihood function is expressed as

$$ln\ p(\mathbf{X}|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{n=1}^{N} ln\{\sum_{k=1}^{K} \pi_k N(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\} \tag{3.50}$$

(Bishop, 2013). Apart from the summation inside the log problem in equation (3.50), Bishop (2013) discusses further other problems associated with the likelihood estimation in Gaussian mixture, such as singularity (or singular covariance matrix) and identifiability. Appendices (C.1.1) and (C.1.2) show maximum likelihood estimation in uni-variate setting, with the former being the closed form single Gaussian and the latter being Gaussian mixture model likelihood estimation. Appendix (C.1.2) sheds a bit more light on how equation (3.50) comes about.

In appendix (C.1.3) univariate likelihood estimates $\hat{\mu_k}$, $\hat{\sigma^2}$ and $\hat{\pi_k}$ are derived for a GMM, the derivations are analogous for the estimation of the multivariate counterparts in equation (3.50). Setting the derivative of $ln\ p(\mathbf{X}|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma})$ with respect to $\boldsymbol{\mu}_k$ equal to 0, the estimate works out as

$$\boldsymbol{\mu}_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk})\mathbf{x}_n \tag{3.51}$$

with

$$N_k = \sum_{n=1}^{N} \gamma(z_{nk}), \tag{3.52}$$

and N_k represents the number points allocated to the k^{th} mixture component. Again setting the derivative of $ln\ p(\mathbf{X}|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma})$ with respect to $\boldsymbol{\Sigma}_k$ equal to 0,

$$\boldsymbol{\Sigma}_k = \frac{1}{N_k}\gamma(z_{nk})(\mathbf{x}_n - \boldsymbol{\mu}_k)(\mathbf{x}_n - \boldsymbol{\mu}_k)^T, \tag{3.53}$$

and finally, π_k works out to

$$\pi_k = \frac{N_k}{N}. \tag{3.54}$$

The equations (3.53),(3.51) and (3.54) are not closed form expressions as they still depend on $\gamma(z_{nk})$ which in turn depends on unknown variables. But it is clearer now if $\gamma(z_{nk})$ is known, the parameters can be estimated and in turn if the parameters are known $\gamma(z_{nk})$ can be derived. EM algorithm uses these variables in the following iterative steps:

1. Make initial guesses of $\boldsymbol{\mu}_k$, $\boldsymbol{\Sigma}_k$ and π_k , compute the log-likelihood using these guess parameters.
2. **Expectation-Step**: Compute the posterior probabilities $\gamma(z_{nk})$ by equation (3.49) using the values of $\boldsymbol{\mu}_k$,$\boldsymbol{\Sigma}_k$ and π_k obtained in step 1.
3. **Maximization-Step**: Having $\gamma(z_{nk})$, estimate $\boldsymbol{\mu}_k$, $\boldsymbol{\Sigma}_k$ and π_k using equations (3.53),(3.51) and (3.54).
4. Compute the log-likelihood using these new parameters. Repeat steps 2 and 3 until convergence, viz, until the change in the log-likelihood estimates is negligible.

Maximisation step is a soft assignment (soft clustering) of each observation to each mixture component according to the mixture component's weight (probabilistic assignment). Each observation has a probability of belonging to each mixture component, with one having higher probability than the others. This notion is an important distinction between the model based clustering and heuristic clustering and it comes through in next section on K-means. As mentioned earlier, the above is a somewhat simplified rational for EM. Both Bonakdarpour (2016) and Hastie et al. (2001) went further into more mathematically rigorous rational showing why the algorithm converges. They delve a bit deeper into matrix algebra. Bishop (2013) also discussed the weaknesses and alternatives to the EM algorithm.

3.4.2.2 K-means Clustering

In K-means clustering there is a known number of required clusters, K. The clusters are distinct and non-overlapping. Each observation belongs to exactly one cluster. The main objective is that the within cluster variation is as small as possible. This within cluster variation is usually measured as a squared Euclidean distance (Hastie et al., 2013). The idea is that the within cluster data point distances should be small compared to data point distances outside the cluster (Bishop, 2013). To put it formally, assume a data set $\mathbf{x}_1, \ldots, \mathbf{x}_N$ with N observations of a $p-$dimensional variable $\mathbf{X}$. A centroid, $\boldsymbol{\mu}_k$, is a $p-$dimensional vector where $k = 1, \ldots, K$ correspond to the k^{th} cluster, also referred to as a cluster centre. The main objective of K-means is to minimise the distance between each data point and its closest cluster centre. The variable $r_{nk} \in \{0, 1\}$ is used for cluster assignments, with $k = 1, \ldots, K$, and $r_{nk} = 1$ if $\mathbf{x}_n$ is assigned to cluster k, and $r_{nj} = 0$ for $j \neq k$. The Euclidean distance is given as

$$J = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} ||\mathbf{x}_n - \boldsymbol{\mu}_k||^2. \tag{3.55}$$

J is referred to as dissimilarity measure and the objective is to minimise J. Such optimisation problem is not solved directly since there are many ways in which n observations can be split into K subgroups, K^n ways to be precise (Hastie et al., 2013). There are various K-means algorithms, the most common being the one that uses the Euclidean squared distance as described above, and it is called Hartigan-Wong algorithm (Kassambara, 2017). K-means algorithm's tries to find values for r_{nk} and $\boldsymbol{\mu}_k$ such that the Euclidean distance in (3.55) is minimised.

This is achieved by iteratively keeping r_{nk} fixed, and solve for $\boldsymbol{\mu}_k$ and vice versa. This is repeated until there is no change in the value or some threshold number of iterations is reached.

Assigning r_{nk} a value of 1 for any value of k that gives the minimium value of $||\mathbf{x}_n - \boldsymbol{\mu}_k||^2$, or simply allocating the n^{th} data point to the closest centroid can be expressed as

$$r_{nk} = \begin{cases} 1 & \textit{if } k = argmin_j ||\mathbf{x}_n - \boldsymbol{\mu}_j||^2 \\ 0 & \textit{otherwise.} \end{cases} \tag{3.56}$$

Fixing r_{nk} and minimising for $\boldsymbol{\mu}_k$ by setting the derivative of J equal to zero,

$$2 \sum_{n=1}^{N} r_{nk} (\mathbf{x} - \boldsymbol{\mu}_k) = 0 \tag{3.57}$$

and therefore $\boldsymbol{\mu}_k$ works out as

$$\boldsymbol{\mu}_k = \frac{\sum_n r_{nk}\mathbf{x}_n}{\sum_n r_{nk}}, \tag{3.58}$$

where $\sum_n r_{nk}$ is the number of points assigned to cluster k and $\boldsymbol{\mu}_k$ is the mean of all data points x_n assigned to cluster k. The iteration steps are described as follows

3.4.2.2.1 The K-means algorithm Summary of the iteration steps as presented in Kassambara (2017):

1. Choose the number of clusters k.
2. Randomly choose k objects (data points) from the data set to serve as cluster centers (centroids).
3. **Cluster assignment step**: Assign each of the remaining objects to the closest centroid, closest according to the Euclidean distance between the object the centroid as in equation (3.56).
4. **Centroid update step**: Compute the new cluster centroids for each cluster,i.e, calculate p-vector means of the data points in a cluster as in equation (3.58).
5. Repeat steps (3) and (4) until convergence, that is, until the centroids stop changing.

3.4.2.3 Similarities between EM and K-means algorithms

K-means' cluster assignment step resembles EM's expectation step while K-means' centroid update step resembles EM's maximisation step. These differ in that K-means algorithm does hard assignments, namely $r_{nk} \in \{0, 1\}$, while EM makes soft assignments based on posterior probabilities. In fact, it is shown by both Bishop (2013) and Hastie et al. (2001) that K-means is a particular limit of EM for Gaussian mixtures. This is achieved by setting $\boldsymbol{\Sigma}_k = \sigma^2\boldsymbol{I}$ where σ^2 is a constant variance and $\boldsymbol{I}$ an identity matrix. That implies that the variance is constant throughout all the mixture components and thus need not be estimated. It turns out, based on tail of behaviour of Gaussian distribution, if $\sigma^2 \to 0$, the posterior probabilities $\gamma(z_{nk})$ in equation (3.49), turn to r_{nk} ($\gamma(z_{nk}) \to r_{nk}$) which results in a hard assignment in the EM algorithm as it is in the K-means algorithm. Based on the fact K-means assume constant variability, equal weights as well as spherical clusters, Grün (2018) makes an argument that the general perception that heuristic clustering impose less assumptions than the model based counterparts is actually inaccurate.

3.4.3 Data Preprocessing

The **Online Retail Data Set** is available on UCI Machine Learning repository website. The dataset is a transactional data for transactions which took place between 1st December 2010 and 9th December 2011 for an online retail business based in the United Kingdom. The dataset contains 541909 rows with 8 columns (number of attributes), and spans retail sales in 39 countries. The data was broken into customer segments using a popular marketing technique called RFM analysis, the acronym stands for Recency, Frequency and Monetary. Recency refers to how recent has a customer visited the store, Frequency is how frequent has the customer visited the store, and Monetary refers how much monetary value has the customer brought to the store. The process of breaking data into segments is called **segmentation analysis**. Its main objective in this case was to pinpoint high and low value customers for targeted marketing. It can be noted that although there is a striking similarity between segmentation analysis and clustering, there is a subtle difference. Segmentation analysis deals with predefined group boundaries, for an example, a grouping may be according to a customer attribute like age; young adult, middle age and older adult. The separation may be objective, informed by historical behaviour or subjective, based on perceived behaviour. In cluster analysis on the other hand, the boundaries are not know in advance, the aim of the analysis is to discover the boundaries using an objective criteria like a data model or an algorithmic model. In the segmentation analysis by customer age example, the marketing strategy would miscategorise the so called "old souls", which refers to individuals who behave like beyond their age. Also, some individuals who experience the so called "mid-life crisis" are known to sometimes have buying patterns which are beneath their age. Ideally, an objective clustering model should be able to pick up such exceptions. Of course, the example is a bit trivialised in order to drive the point home. Segmentation analysis is bit more advanced and objective than illustrated here.

Coffey (2016) presented an RFM analysis of the Online Retail Data Set using R. For this preprocessing step that analysis served as a guideline. The analysis was restricted to one year worth of data and only United Kingdom purchases were considered for analysis. Observations with missing values on key attributes like customer IDs and invoice numbers were removed. This further reduced the data under consideration to 3863 observations. Recency was calculated as the number of days that elapsed since the last purchase by a customer as indicated by "InvoiceDate". The reference date for recency calculation was 9^{th} December 2011. Frequency was calculated as the number of purchases by a customer as indicated by the number of invoices in that year for that customer. Lastly, the monetary value was measured as the total money spent by a customer in that year. A zero on the monetary value indicates that the item was sold and later returned to the store by the customer. Log transformation was applied as the variables were positively skewed. Since some of the monetary value had zero values, 0.1 was added to each value to avoid log of zero. Lastly, the three variables were standardised as z-scores using R's *scale()* function. The format of the original data is shown in Table 3.32. Numerous R utility functions were used for the transformation which resulted in the format shown in Table 3.33. The z-score variables **recency.z**, **frequency.z** and **monetary.z** were used in all the subsequent analysis.

Table 3.32: Top 6 data entries of original data

InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	12/1/2010 8:26	2.55	17850	United Kingdom
536365	71053	WHITE METAL LANTERN	6	12/1/2010 8:26	3.39	17850	United Kingdom
536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	12/1/2010 8:26	2.75	17850	United Kingdom
536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	12/1/2010 8:26	3.39	17850	United Kingdom
536365	84029E	RED WOOLLY HOTTIE WHITE HEART	6	12/1/2010 8:26	3.39	17850	United Kingdom
536365	22752	SET 7 BABUSHKA NESTING BOXES	2	12/1/2010 8:26	7.65	17850	United Kingdom

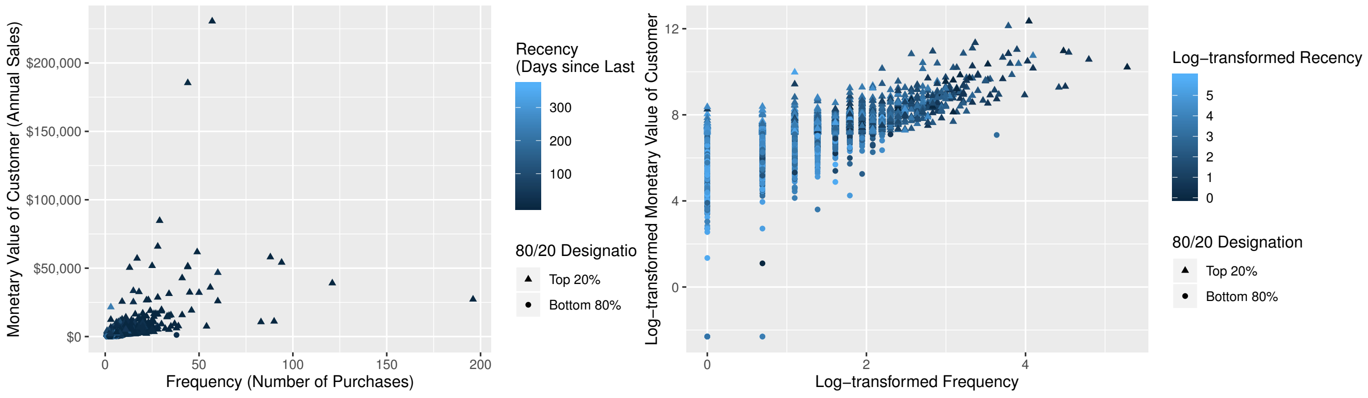
Table 3.33: Top 5 data entries transformed variables

CustomerID	recency	frequency	monetary	recency.log	frequency.log	monetary.log	recency.z	frequency.z	monetary.z
12346	326	1	0	5.786897381	0	-2.302585093	1.476973681	-1.041919353	-6.433370109
12747	3	10	3837.45	1.098612289	2.302585093	8.252589421	-1.950577446	1.5373188	1.306143017
12748	1	196	27214.9	0	5.278114659	10.21152357	-2.753760212	4.870354962	2.742518754
12749	4	5	3868.2	1.386294361	1.609437912	8.260570413	-1.740256403	0.76089075	1.311995028
12820	4	4	942.34	1.386294361	1.386294361	6.848472257	-1.740256403	0.510936747	0.27658323

3.4.4 Data Analysis And Results

3.4.4.1 Analysis

The dataset features are plotted in Figure 3.12. Coffey (2016) alludes to the pareto principle which states that “80% of results comes from 20% of causes”. In this dataset it worked out as 80% of sales came from the top 29% of customers. The monetary value and frequency were chosen as a key measurement, with the requirement being to track high-value/high-frequency and low-value/low-frequency customers. Figure 3.12 shows the plots of these variables, Figure 3.12a shows the data in the original scale which is hardly interpretable as the data seems condensed together in one area with outliers (top 20%) and Figure 3.12b shows the log transformed variables which give some clarity. It is clearer in Figure 3.12b that the high-value/high-frequency customers have lower recency, which means they have been to the store on more recent days preceding 9th December 2011.



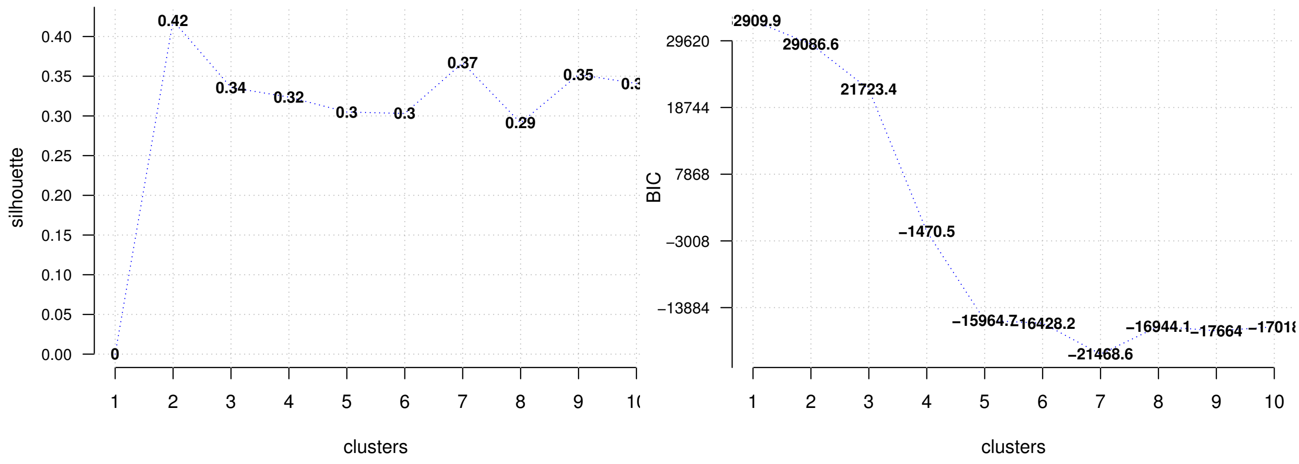
(a) Days Since Last Purchase (b) Log Transformed Monetary Value

Figure 3.12: Features Plots

3.4.4.2 Determining The Number Of Clusters

Determining the number of clusters is not an exact science. As stated in Everitt et al. (2011), some procedures are often subjective and largely informed by user’s prior expectations. Everitt et al. (2011) discusses several approaches to choosing the number of clusters one of which is the silhouette plot. The silhouette coefficient ranges from -1 to +1. The value compares the separation of an object from its cluster against the heterogeneity of the cluster (Everitt et al., 2011). It basically measures how close is an object in a cluster to objects in the adjacent cluster. In other words, silhouette coefficient measures how similar an observation is to its assigned cluster compared to other clusters. A value close to +1 signifies that the object is “well-classified”. Likewise, a value close to -1 signifies that the object is “miss-classified”. A value of zero shows a lack of clarity of whether the object is correct in the assigned cluster or should have been assigned to an adjacent cluster. This means the highest possible silhouette coefficient is preferred in choosing the number of clusters. Figure 3.13a shows the silhouette average values against possible number of clusters. The values were plotted using **ClusterR** package. The highest silhouette average corresponded to two clusters, but according to Fraley and Raftery (1998), the number of clusters should be higher than the dimensions (three dimensions in this case) to avoid having a non-singular variance-covariance matrix. The minimum needed here was four clusters. The non-singular variance-covariance matrix requirement is important in the model based clustering setting, EM algorithm even fails if the covariance matrix is singular (Fraley and Raftery, 1998). On top of that technical reason for deviating from the two clusters as suggested by the silhouette plot, a more intuitive reason is that having only two clusters is not a meaningful subgrouping of the given data. The next highest silhouette average corresponded to seven clusters with a silhouette average of 0.37. The silhouette average values were below 0.5 and Everitt et al. (2011) suggest that reasonable classification values should be above 0.5. A value below 0.2 would have meant a substantial lack of structure in the clusters.

The cluster number determination above is distance based and is more suitable for a heuristic clustering method like K-means than the model based GMM clustering. For model based clustering, BIC is employed. BIC is used to choose the number of components in a mixture as well as the underlying densities of the components (Fraley and Raftery, 1998). Fraley and Raftery (1998) cover further technical details of BIC. In a nutshell, BIC is an estimate function of a posterior probability of a model being the true model, under Bayesian conditions, a lower BIC means a model is more likely to be a true model. Other texts, like Fraley and Raftery (1998) and **mclust**[22] R package invert the sign for the BIC which means a larger BIC is preferred in such cases. The inversion of the sign is for cosmetic reasons like a better interpretability of the plots. The **ClusterR** package was used to plot the BIC values against possible number of clusters. The lowest BIC corresponded to seven clusters as shown in Figure 3.13b.



(a) Silhouette Plot for K-means clusters (b) BIC Plot for GMM-EM clusters

Figure 3.13: Plots for determining no. of clusters

NbClust package has 39 different indices for determining the number of clusters, the indices include the silhouette average. The package runs through all of the indices and produce an output of proposed number of clusters as suggested by the various indices. A decision is taken according to how many clusters do the majority of indices propose. The majority of indices proposed seven clusters for the data in this case study. The output is shown in Table C.1 of Appendix C.1.4. The package also outputs the graphical determination of Hubert index and D index and their second differences, plotted against possible number of clusters. The significant peak in the second differences plot signifies an optimal number of clusters. The second differences plot had significant peaks on seven clusters as shown in Figure C.1 of Appendix C.1.4.

[22](Scrucca, 2021)

3.4.4.3 Model Fitting

The function *Mclust()* from the **mclust** R package was used to fit the GMM. The function takes the preprocessed data and an optional number of clusters as arguments (it works out the number of clusters if not provided). The input for the number of clusters was seven as was worked out in the previous section. The function uses the EM algorithm, and produces a wide range of mixture models based on volume, shape and orientation. It chooses a mixture model that maximises the BIC as the optimal model. The sign is inverted hence the maximisation of the BIC is preferred. The resulting seven components mixture is variable, equal and variable (VEV). This means the clusters have an ellipsoidal distribution with different volume, same shape and different orientation. The resulting GMM model output is shown in Appendix C.1.5. Everitt et al. (2011) explains in detail the rational behind the various model names available in **mclust** package.

Table 3.34: K-means Clusters

Cluster	Monetary	Frequecy	Recency	No. of Observations
1	223.47	1	233	846
2	838.70	3	10	538
3	0.00	1	110	20
4	773.44	3	86	816
5	2140.07	6	30	605
6	266.35	1	39	700
7	4706.14	13	5	338

Table 3.35: GMM-EM Clusters

Cluster	Monetary	Frequecy	Recency	No. of Observations
1	201.16	1	250	613
2	1353.56	7	22	566
3	2101.98	4	42	142
4	1375.71	5	24	637
5	1730.65	4	26	418
6	303.16	1	53	719
7	523.11	2	60	705

The *kmeans()* function from R core was used to fit K-means and the resulting model output is shown in Table C.5 of Appendix C.2. The function takes the number of centroids K, which was seven as worked out in the last section, and the preprocessed data as arguments. The clusters are shown in Table 3.34 and Table 3.35. The numerical values in Table 3.34 and Table 3.35 are medians in the original scale rather the means. This is because the data exhibits outliers and skewness and therefore a median is a better measure of central tendency.

3.4.4.4 Results

This section presents the comparison in the form of cluster stability using bootstrapping, compactness using distance measures and cluster visualisation. The focus is on cluster validity as opposed to accuracy metrics because cluster analysis deals with unlabelled data and is exploratory in nature. This is in contrast to the previous two case studies where the goal was prediction accuracy and the data was labelled. The hold out sample in those two use cases enabled for quantification and comparison of performance metrics but that is not possible with unlabelled data.

3.4.4.4.1 Model validation using nonparametric bootstrapping

Statistical significance testing for model validation in cluster analysis is a disputable notion. It is strongly argued in Institute (2017) that significance tests for testing cluster differences are not valid. This is because the aim of clustering methods is to make the distances between the clusters as large as possible (maximum separation) and this violates the usual assumptions of the statistical significance tests. However, there are ongoing R projects trying to make significance testing possible in cluster analysis. These are in the form of packages like **sigclust**, **mixR** and **poLCA** to mention a few. Huang et al. (2015) is a vignette for **sigclust**, **sigclust** tests for statistical significance of Gaussian mixtures but is limited to only two mixtures (two clusters). Xu (2016) is a vignette for the **CancerSubtypes** package, the package contains a function *sigclustTest()* which is built on top of **sigclust** package. The function extends the number of clusters but it requires an extra parameter which is a vector of codes (some form of genetic coding) which would not be possible for the customer data (financial data). The package **mixR**[23] works for single variable data and the package **poLCA**[24] is meant for latent class analysis on categorical observations. All of the above cluster statistical significance testing packages were not explored in this research because their assumptions could not be met by the customer data used for this case study.

Nonparametric bootstrapping is one of the alternative forms of cluster validation that can be considered given the challenges of statistical significance testing mentioned in the preceding paragraph. Nonparametric here means the distribution is unknown and where it is known, the distribution parameters are not specified. Hinkley (1988) describes the ethos of bootstrapping as the simulation of applicable statistical procedure with least model assumptions. One of the main goals of cluster validation is establishing the presence of cluster stability, or lack thereof. In essence, a cluster should not vanish if the data is slightly altered, otherwise such cluster would be considered unstable (Hennig, 2007). The basic idea behind nonparametric bootstrapping is that multiple samples are drawn from observed data (resampling). The sample sizes of the bootstrap samples are the same as the size of the observed data (original sample). The data points (observations) from the data are usually drawn with replacement, which means that some observations can be repeated in each of the bootstrap samples. This makes it possible to draw many samples from the data (replications).

Scrucca et al. (2016) go into detail of bootstrap CI's implemented in the package **mclust**. In this implementation, each sample is drawn and partitioned such that the partitions resemble the clusters (Gaussian mixtures) of the original sample. Then various statistics are estimated from the bootstrap clusters, the estimates are the same as the ones taken from the original clusters ($M\hat{L}E$'s). CI's of the various estimates are constructed from the bootstrap distribution. If the CI's contain the corresponding $M\hat{L}E$, then it can be concluded that the corresponding clusters valid and not spurious. There are two functions in **mclust** which are used in bootstrapping problems, namely *mclustBootstrapLRT()* and *MclustBootstrap()*. The function *mclustBootstrapLRT()* is used to determine the optimal number of Gaussian mixtures, it achieves that by resampling and computing likelihood ratio test[25]. It takes in the data and the model name (the model name was 'VEV' in this case). The null hypothesis in such test is $H_0 : G = G_0$, against the alternative $H_1 : G = G_0 + 1$, where G_0 is the number of mixtures. The function performs the test sequentially starting from $G_0 = 1$, incrementing by one until the p-

[23](Yu, 2021)

[24](Linzer, 2014)

[25]Section 8.2.3.3 on Heckert et al. (2002) provides more details of likelihood ratio test (LRT).

value is not statistically significant. If the p-value is not significant when $G_0 = 1$, that means the data is homogeneous and there is no scope for GMM (but it does not rule out other forms of cluster analysis). Table C.3 shows the results of the *mclustBootstrapLRT()*, the table shows 7 as the number of optimal clusters, this is equal to the number of clusters that were computed in Section 3.4.4.2. The function *MclustBootstrap()* is used for bootstrap inference and it implements the computation of bootstrap CI's. It takes the actual fitted model object, the number of replications and resampling type (with options being; nonparametric bootstrap, parametric bootstrap, weighted likelihood bootstrap, and jacknife). Table C.4 shows the results of the nonparametric bootstrap CI's computation. The function computes intervals for the means, variance and mixing proportions. The table only shows the results of the means and mixing proportions at 95% level of confidence. The estimates of the means and mixing proportions (MLE's) in Table C.2 fall inside the corresponding CI's in Table C.4. This is an indication of valid clusters for the GMM. Figures C.3 illustrates the bootstrap CI's for the mixing proportions in the form of histograms. The dotted lines are the corresponding $M\hat{L}E$'s, it can be seen that all the dotted lines are inside the histograms.

The cluster validation for the k-means was also performed using nonparametric bootstrapping. The function *clusterboot()* from **fpc**[26] package performs bootstrapping on Jaccard coefficient. This coefficient measures the degree of similarity between two sets (clusters in this case study). The coefficient ranges from 0 to 1 (or 0% - 100%), and the higher its value the higher is the degree of similarity. Basically, the Jaccard coefficients of the original clusters to the most similar clusters on the bootstrap resamples are calculated and averaged, the averages (means) then represent the degree of similarity. Clusters with higher degree of similarity are deemed to be more stable. Hennig (2007) goes into more technical details of Jaccard coefficient bootstrapping. Table C.6 shows the resulting means of Jaccard coefficient. The lowest mean value is 0.8658098 on cluster 4, which indicates a high degree of stability in all the clusters. The table also shows other statistics. The value of 'dissolved cluster' shows how many times the clusters from bootstrapping were dissolved, a low value is preferred. A cluster is dissolved if the mean Jaccard coefficient from bootstrap samples is less than or equal to $\frac{1}{2}$ (Hennig, 2007). Again cluster 4 is the only cluster that had corresponding clusters dissolved, even though this happened only twice. This dissolution corresponds to cluster 4 having the lowest Jaccard bootstrap mean. But, overall the bootstrapping results indicate valid and stable clusters for the K-means.

A direct comparison of model stability between the two clustering techniques is not possible since GMM is model based and K-means is heuristic. And, the validation used different bootstrapping paradigms with bootstrap CI's being used for GMM and Jaccard coefficient bootstrapping being used for K-means. Bootstrapping helped establish that the clusters in the data were not due chance but statistically sound clusters.

3.4.4.4.2 Cluster distance measures

The *cluster.stats()* function from **fpc** package computes some of the well known cluster validation indices like Dunn Index (DI) and the silhouette coefficient. The function takes a minimum of two arguments, the pre-processed data and the clustering matrix.

Dunn's index tests for compactness of the clusters, compact clusters is when the variance between cluster members is small, and the distance between the cluster means is adequately large. A higher

[26](Hennig, 2020)

Table 3.36: Cluster Distance Measures

	K-means (7)	GMM-EM(7)
Av silhouette index	0.3202	0.0733
Dunn's Index	1.1120	0.2302
Within SS	2314.8220	5712.7730

value of Dunn's index shows better clustering. A higher value of the silhouette index shows that the observation is well matched to its cluster compared to other clusters. Another metric noted from the output of *cluster.stats()* function is the within cluster sum of squares which also tests compactness of the clusters, a lower value indicates compact clusters. Table 3.36 shows the validation metrics for the two models, K-means outperformed GMM in all three metrics. Section 2.2.3 reflected on a study by Baid et al. (2016) where K-means, GMM and Fuzzy C-means were compared. Fuzzy C-means have similarities to both GMM and K-means. Similar to K-means in that they are both heuristic and follow almost similar training algorithm, and similar to GMM in that they both do soft clustering. The case study was extended by fitting the fuzzy C-mean model to the data. The results are shown in Appendix C.3.1. It had a silhouette average of 0.2980, a Dunn's index of 1.1475 and a within sum of squares value of 2666.221. This meant that the fuzzy C-mean model outperformed the GMM but fell below the performance of K-means. This is in contrast to the findings of Baid et al. (2016) in which fuzzy C-means outperformed the two models.

3.4.4.4.3 Visualisation

Visualisation is another useful tool in cluster validation. Figures 3.14 and 3.15 show the visual representation of the clusters for both K-means and GMM. There does not seem to be clear partitions in both visual representations. This goes along with what was seen in the silhouette plot in Figure 3.13a. The plot values were all below 0.5 which indicated a lack of distinct clusters. Figure 3.14 shows that K-means did a better job separating hig-value/high-frequency in cluster 7, and low-value/low-frequency in cluster 3. There is a lot of overlapping between the other remaining clusters 1,2 and 4. Table 3.34 also shows that K-means correctly partitioned the zero valued monetary group as a separate cluster, which is a group that would have bought and returned items. Such a cluster gives an important signal on the data. The marketing team could dig further into the items to find out whether there was a pattern on the returns. For example, if the returns were on the same item or brand, that could indicate compromised quality or defects. The GMM model missed this signal on the data. Figure C.2 of Appendix C.1.5 shows the rest of the pairwise cluster classification plots from **mclust** package.

The code listing for the cluster analysis in this case study is in Appendix C.4.

3.4.5 Discussion and Conclusion

3.4.5.1 Discussion

Nonparametric bootstrapping gave confidence that partitioning the data using the two clustering techniques considered in this case study gives valid and stable clusters. Although that was an essential technical question to answer, it did not give further information in terms of cluster 'performance'. Distance measures were looked to gauge if any performance metrics could be extracted from the

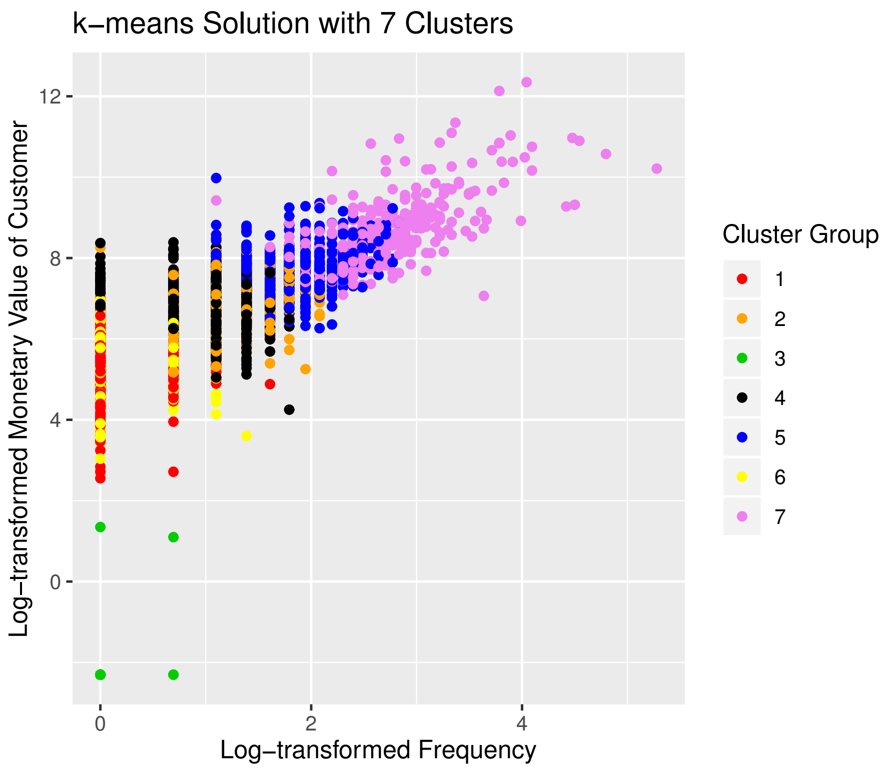


Figure 3.14: K-means clusters

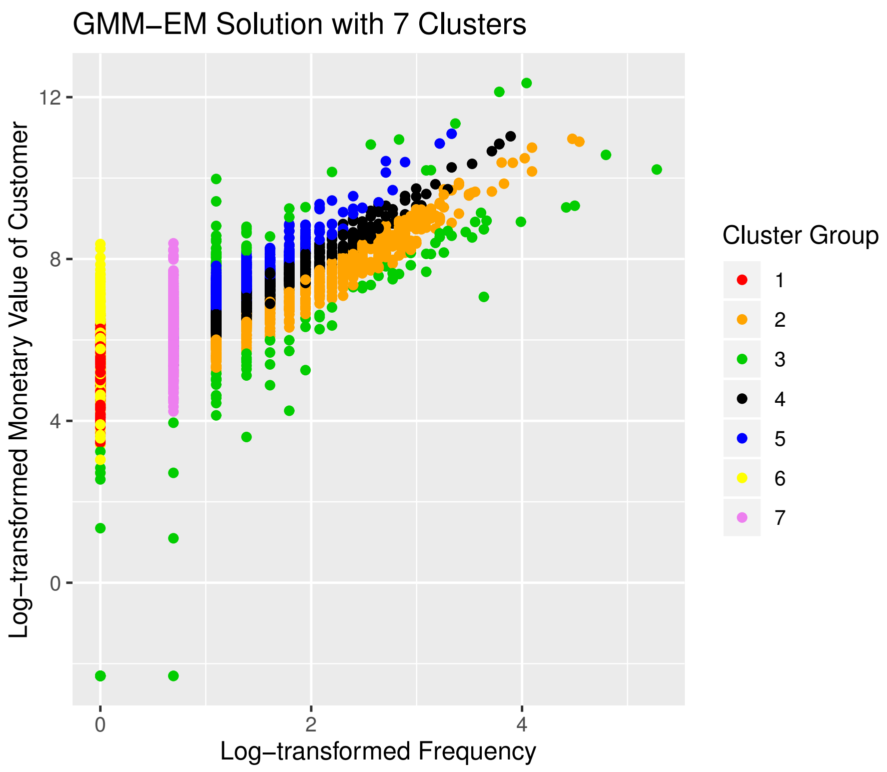


Figure 3.15: GMM-EM clusters

clustering solutions.

There is a lack of consensus on whether it is correct to use silhouette coefficient in validating a Gaussian mixture which is not a centroid based algorithm, and even generally, whether it is correct to use any distance based measurement in validating a soft clustering model like GMM. For a likelihood model like GMM, a good performance metric would have been a BIC, but K-means is a heuristic algorithm with no likelihood function, so a BIC would not have been an ideal indicator [27]. There are other techniques where clustering solutions are better compared, like synthesisation and simulation carried out in Baid et al. (2016) and pre-labelling of data as performed in Bi et al. (2005). These techniques do give better comparison metrics but also do change the problem to a classification problem, and the classification problem was dealt with in the classification case study.

The data visualisation of the clusters seems to have captured the essence of performance differences adequately. It would have been a tough task to convince a domain expert (or business executive) had the performance metrics given a different story. K-means algorithm was a somewhat better clustering solution for the given problem compared to GMM. As it was noted, both models performed relatively poorly in creating distinct partitions(clusters), and this could be due to numerous reasons. One could be that the data was not suitable for the clustering problem, that is, there was no way that the data could be separated into more distinct partitions that was already achieved. But establishing such fact with certainty requires quite deeper and intimate knowledge of the process that produced the data. Also, probably having more knowledge of the domain of application, marketing in this case, would have resulted in a better fine tuned pre-processing step in preparing the data for the clustering problem. Usually, domain knowledge experts are roped in and participate in an iterative process in order to produce the best possible clustering solution as mentioned in Grün (2018).

The choice of a clustering model can also be questionable given the lacklustre performance of the two clustering models. The bootstrapping model validation gave a level of comfort that the data was separable given the cluster stability achieved for both models. But, there are still many more clustering techniques that were not tried in the study. For instance, there are numerous variants of K-means, each trying to improve a certain aspect of the algorithm. Baid et al. (2016) explored fuzzy C-means and Zhang et al. (1999) looked at the performance of KHM as discussed earlier. A fuzzy C-means model was also fitted to the data in this case study and did not improve on the performance of K-means but outperformed GMM. K-means algorithm was used in this research because it is a de facto heuristic clustering technique in the machine learning world and it is relatively easy to follow its theoretical basis. And, there are many other clustering techniques which are founded on K-means. K-means also has a wide range and well tested software tools and packages compared to its variants which still have to mature in that respect. The algorithm has enjoyed success in a wide range of applications, Coates and Ng (2012) go through some ingredients and tricks to get the best out of K-means with a focus on image processing. Similar arguments could be made for the GMM, other model based techniques could have been explored but the mathematics in the GMM is quite tractable and not too demanding on computation resources and generally performs well. It was stated earlier that clustering is unsupervised learning and is exploratory in nature and usually has no accuracy metrics. But the collaboration with domain experts should give clarity to cluster analysis goals which should lead to better model choice for meaningful clusters.

3.4.5.2 Conclusion

The K-means model performed relatively better than the GMM model although both models did not do a good job in creating distinct partitions. Better in this context is confined to the definition of the problem, which was to partition frequency and monetary value with the aim of discovering high-

[27]Although there are some who calculate BIC for K-means.

value/high-frequency customers as well as low-value/low-frequency customers and everything else in between. K-means managed to uncover an *interesting* and *useful* pattern in the data by creating a cluster consisting of only zero monetary value customers which are customers that bought and returned items. Clustering is exploratory in nature therefore the initial clustering solution is usually not necessarily the final solution. The preceding section mentioned the role that domain experts play in an iterative process to refine the clustering solution. In instances where it becomes clear that the clusters would be spherical in nature with no overlaps, and having a constant variance, a K-means algorithms could be fitted. If the nature of the clusters are likely to have overlaps with varying variance between clusters and oblong shapes, a GMM model could be fitted. There are other clustering strategies that could be employed other than the two mentioned in this research. The big takeaway from this part of the research was that there is no one size fits all model when it comes data analysis. Different environments produce different kinds of problems which require different kinds of solutions.

Chapter 4

Conclusion and Recommendations

The aim of the study was to empirically compare statistical data modelling and machine learning in terms of rationale, model parsimony as well as predictive accuracy. Each of the three case studies focused on different data analysis problems in the form of prediction, classification and clustering. Prediction comparison was performed using a GARCH model for statistical data modelling, and an ANN model was used for machine learning. For classification an LDA model was compared to an SVM model, and for clustering a GMM model was compared to a K-means model. It was noted that classification is also a form of predictive modelling, but the term *prediction* was exclusively reserved for the prediction of a continuous quantitative value.

Machine learning algorithms generally performed marginally better than statistical data models in the case studies but were more difficult to interpret. The findings in this research can be summarised by the following three bullet points:

- There is so much that rides on statistical significance testing.
- There are some statistical analysis techniques which are required for even machine learning algorithms.
- Understanding the domain problem is essential in choosing an appropriate model.

To proceed with volatility modelling in the prediction case study, a few assumptions needed to be tested to ensure EMH was obeyed. The tests included stationarity and serial correlation, and these were statistical significance tests. Testing for EMH was a prerequisite whether the model in question was a machine learning algorithm or a statistical data model. This points to a central role that significance testing plays in econometrics. This is in contrast to the remarks of Marton Wilk in Mallows (2006). Clearly, significance testing is not something that is done to pass time when it comes to EMH. Wilk's lambda test of no group separability was performed in the classification case study. For the two-class problem, the null hypothesis was rejected which meant there was evidence of group separability. This group separability showed in both the LDA and the SVM models as the correct classification percentages were relatively higher in the confusion matrix. When the null hypothesis was accepted in the three-class problem, the classification percentages dropped for both models. This indicated that before applying even a machine learning algorithm like SVM, there is something to gain in executing such a significance test. Yes, significance testing may have been misused in some corners as alluded to by Johnson (1999). This is to be expected as many disciplines outside statistics have long embraced statistical methods for their data analysis and research initiatives. This is indicative of the success that statistics as a subject has enjoyed over the years. It is a tool used in the arts, natural sciences and social sciences and in many walks of life. In this respect, statistics is a victim of its own success. It is up to the statisticians to clarify these misconceptions by making the subject more accessible, and take it out of the university departments. Almost every second blog on the internet is dedicated to some form of machine learning algorithm, such visibility gives machine learning disproportionate voice amongst captains of industries and to the man on the street. To compete in that space, there should be a concerted effort to disseminate statistical thinking to society. If someone searches for "regression analysis" in the internet, the information that should come up should be the ordinary least squares regression instead of a gradient descent regression or some other machine learning regression. Statisticians should be at the forefront of statistical thinking in all spheres of life instead of relying on people who are not trained in statistics explaining things that only statisticians can know. Postgraduate students could be encouraged in this regard, by having the quality of their internet presence examined as part of their formal study and not only focus on journal articles. This can also be a requirement for academic advancements such a professorships and so on. As Chambers (1993) warned, if the statisticians remain aloof, society will lose on the mental qualities that they provide. Unfortunately the loss will not only be to society but statistics itself will lose on the race for relevance and also as Hand (2000) noted, lose on much needed research funding.

In a tweet[1], François Collet, an AI researcher and author wrote :"Neural networks are a sad misnomer. They are neither neural nor even networks. They are chains of differentiable, parametrized geometric functions, trained in gradient descent (with gradients obtained via chain rule). A small set of highschool-level ideas put together". Of course this was a tongue in cheek comment which somewhat had a lot of truth in it. The idea here is that most machine learning algorithms are a series of simple mathematical ideas that are well marketed, and they are generally effective and if used responsibly, they can benefit society. Statistical significance testing is not state of the art statistical idea either. But, it was shown in both the prediction and classification problems in this research that such an elementary statistical concept plays a big role in data modelling. How many analysts are currently missing out on such elementary and yet powerful statistical ideas in the data analysis arena? If these algorithms are simple mathematical ideas, then statisticians should step in to induce statistical ideas into the mix. Some of the prior studies explored in Section 2.2.1 like Donaldson and Kamstra (1997) and Yim (2002) show that there is an active research taking place around hybrid models, which make use of both statistical data models and machine learning algorithms. Statisticians should form part of such initiatives to give guidance and statistical way of thinking. Another effective way of disseminating statistical thinking is through rapid developments in statistical computing. Computing is central to 4IR. In the development of S programming language[2], John Chambers is quoted to having said the objective was, "to turn ideas into software, quickly and faithfully". That was back in 1976, that quote could not be more relevant now in this race for relevance in the face of 4IR. Statisticians need to take advantage of the popularity of R programming language[3] by peer reviewing packages and even better developing new packages. R programming was used for all the problems in this research.

The clustering solutions for both models needed further refining, a step that is usually performed with the input from domain experts (Grün, 2018). Almost all statistical problems in practice require input

[1]https://twitter.com/fchollet/status/951906139632840704
[2]https://en.wikipedia.org/wiki/S_(programming_language)
[3]https://en.wikipedia.org/wiki/R_(programming_language)

from those with domain knowledge, but since clustering is exploratory in nature, the domain knowledge becomes crucial for a clustering solution. For an example to progress with the clustering problem in this research, FRM analysis as illustrated by Coffey (2016) was required. In another domain, or had the problem been phrased differently in the same domain (marketing), probably a different pre-analysis step to shape the same data would have been required. It was also shown in the volatility prediction problem that in order to proceed with any modelling EMH needed to be satisfied. This is why there are many fields of application within statistics like biostatistics, econometrics, statistical physics and many others. These sub-fields or specialisations have contributed a great deal to the development of statistics as a subject. Sir R. A. Fisher mentioned in this research as one of the main drivers behind giving statistics a mathematical character was also a geneticist. His contribution to statistics came while working as a geneticist, introduced analysis of variance (ANOVA) in the design and analysis of experiments and had immense contribution in the development of maximum likelihood estimation. While domain specialization within statistics is a good thing, another view is that the subject should be abstracted from application, like in the tool-maker and carpenter analogy in Mallows (2006). It is an individual's decision to make in deciding whether to go a domain specialisation route or the route of abstraction from the domain so that they can apply statistical knowledge to a wide range of subjects. But, the bottom line is, thorough understanding of the domain problem at hand is necessary for modelling whether one is doing statistical data modelling or machine learning.

Just like Bzdok et al. (2018) concluded, the boundary line between statistical modelling and machine learning is becoming blurred with overlaps between these disciplines increasing by day. In this research bootstrapping was applied in both K-means and GMM. In fact, GMM itself is implemented through an algorithm (EM) even though it is a mixture of parametric models. This makes it difficult to draw a clear line of distinction between the two disciplines. Probably the following considerations might serve as guidelines in deciding which one to use in data analysis:

- If the quantitative measure of uncertainty is critical for the analysis, a statistical model would likely be a better choice.
- If the data generating process is well known and can be related to existing (biological) knowledge (Bzdok et al., 2018), a statistical model would probably be a better choice.
- If prediction accuracy is the ultimate goal, a machine learning algorithm will likely be a better choice.
- If the data is wide and short (more columns than rows)(Nongxa, 2017), a machine learning algorithm would likely be a better choice .
- If the data is from uncontrolled experimental design with complicated linear interactions (Bzdok et al., 2018), a machine learning algorithm would likely be a better choice .

Of course there are probably many more other considerations that might help in arriving at a correct decision. Prediction accuracy and quantification of uncertainty are not necessarily mutually exclusive, in many situations these would be both critical. Such situation would probably be served with hybrid models.

In this research the hold-out method was used for cross validation in the prediction and classification (supervised learning) use cases. Although the splitting of the data was randomised, this method is a bit limiting because the testing was done on one sample and this may introduce a bias. Bootstrapping was employed in cluster validation, but bootstrapping can also be employed in cross validation of performance in the prediction problems. In such validation, the hold-out sample is bootstrapped resulting in many more samples upon to test model adequacy. Tsamardinos et al. (2018) present their own flavour called Bootstrap Bias Corrected Cross Validation. Future work should focus on bootstrap cross validation in evaluating performance in this kind of comparative research.

Davison and Hinkley (1997) describe bootstrapping as harnessing the computer to obtain reliable standard errors, CI's and other measures. And, it was noted in this research that a more popular and "superior" Keras framework for neural networks was dropped in favour of a lightweight and probably "inferior" package due to memory and other infrastructure requirements. So, the underlying infrastructure whether one is bootstrapping or running an algorithm plays a role. Future studies could also focus on quantifying how much of a role does the underlying infrastructure play in optimising such computationally intensive strategies.

Bibliography

Agrawal, Astha, Herna L. Viktor and Eric Paquet (2015). "SCUT: Multi-class imbalanced data classification using SMOTE and cluster-based undersampling". In: *2015 7th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K)* 01, pp. 226–234.

Andersen, G. and Tim Bollerslev (1998). "Answering the Skeptics: Yes, Standard Volatility Models do Provide Accurate Forecasts". In: *International Economic Review* 39.4, pp. 885–905. ISSN: 00206598, 14682354. URL: `http://www.jstor.org/stable/2527343`.

Andersen, T., T. Bollerslev, F.X. Diebold and P. Labys (2003). "Modeling and Forecasting Realized Volatility". In: *Econometrica* 71, pp. 529–626.

Arnerić, Josip, Tea Poklepović and Zdravka Aljinović (2014). "GARCH based artificial neuralnetworks in forecasting conditional variance of stock returns." In: *Croatian Operational Research Review* 5(2), 329–43. URL: `https://www.researchgate.net/publication/333094907_Time_series_analysis_of_human_brucellosis_in_mainland_China_by_using_Elman_and_Jordan_recurrent_neural_networks#pag:8:mrect:(393.13,640.30,14.62,9.34)`.

Asai, Manabu, Michael McAleer and Jun Yu (2006). "Multivariate Stochastic Volatility: A Review". In: *Econometric Reviews* 25.2-3, pp. 145–175. DOI: `10.1080/07474930600713564`. eprint: `http://dx.doi.org/10.1080/07474930600713564`. URL: `http://dx.doi.org/10.1080/07474930600713564`.

Baid, U., S. Talbar and S. Talbar (2016). "Comparative Study of K-means, Gaussian Mixture Model, Fuzzy C-means algorithms for Brain Tumor Segmentation". In: *International Conference on Communication and Signal Processing 2016 (ICCASP 2016)*. Atlantis Press, pp. 583–588. ISBN: 978-94-6252-305-0. DOI: `https://doi.org/10.2991/iccasp-16.2017.85`. URL: `https://doi.org/10.2991/iccasp-16.2017.85`.

Bauwens, Luc, Sébastien Laurent and Jeroen V. K. Rombouts (2006). "Multivariate GARCH models: a survey". In: *Journal of Applied Econometrics* 21.1, pp. 79–109. ISSN: 1099-1255. DOI: `10.1002/jae.842`. URL: `http://dx.doi.org/10.1002/jae.842`.

Bhaumik, Sankar Kumar (Apr. 2020). *Lecture notes in Discriminant Analysis*. Department of Economic Studies & Policy Central University of South Bihar. URL: `https://www.cusb.ac.in/images/cusb-files/2020/el/eco/Discriminant_Analysis_Part%20I.pdf`.

Bi, Weixing, XuGang Wang, Zheng Tang and Hiroki Tamura (2005). "Avoiding the Local Minima Problem in Backpropagation Algorithm with Modified Error Function". In: *IEICE Transactions* 88-A, pp. 3645–3653.

Bishop, C.M. (2013). *Pattern Recognition and Machine Learning*. 1st ed. Information science and statistics. Springer (India) Private Limited. ISBN: 9788132209065. URL: `https://booksrun.com/textbooks/9788132209065-pattern-recognition-and-machine-learning-1st-edition`.

Bonakdarpour, Matt (2016). *Introduction to EM: Gaussian Mixture Models.* URL: https://stephens999.github.io/fiveMinuteStats/intro_to_em.html (visited on 09/05/2020).

Bouchard, Kevin and Sylvain Giroux (2015). "Smart Homes and the Challenges of Data". In: *Proceedings of the 8th ACM International Conference on PErvasive Technologies Related to Assistive Environments.* PETRA '15. Corfu, Greece: Association for Computing Machinery. ISBN: 9781450334525. DOI: 10.1145/2769493.2769519. URL: https://doi.org/10.1145/2769493.2769519.

Brailsford, Tim., Robert W. Faff and Monash University. (1993). *An evaluation of volatility forecasting techniques / by Timothy J. Brailsford and Robert W. Faff.* English. Dept. of Accounting, Finance, Faculty of Economics, Commerce and Management [Clayton, Vic.], 29 p. :

Breiman, Leo (Aug. 2001). "Statistical Modeling: The Two Cultures (with comments and a rejoinder by the author)". In: *Statistical Science* 16.3, pp. 199–231. DOI: 10.1214/ss/1009213726. URL: https://doi.org/10.1214/ss/1009213726.

Brooks, Corrinne (1998). "Predicting stock index volatility: can market volume help?" In: vol. 17, pp. 59 –80.

Buja, Andreas (2006). "[Tukey's Paper after 40 Years]: Discussion". In: *Technometrics* 48.3, pp. 327–330. ISSN: 00401706. URL: http://www.jstor.org/stable/25471202.

Bzdok, Danilo, Naomi Altman and Martin Krzywinski (2018). "Statistics versus machine learning". In: *Nature Methods* 15.4, pp. 233–234. ISSN: 1548-7105. DOI: 10.1038/nmeth.4642. URL: https://doi.org/10.1038/nmeth.4642.

Chambers, J. M. (1993). "Greater and Lesser Statistics: A Choice for Future Research". In: *Statistics and Computing* 3.4, pp. 182–184.

Charef, Fahima and Fethi Ayachi (2016). "A Comparison between Neural Networks and GARCH Models in Exchange Rate Forecasting". In: *International Journal of Academic Research in Accounting, Finance and Management Sciences* 6.1, pp. 94–99. URL: https://EconPapers.repec.org/RePEc:hur:ijaraf:v:6:y:2016:i:1:p:94-99.

Charrad, Malika, Nadia Ghazzali, Véronique Boiteau and Azam Niknafs (2014). "NbClust: An R Package for Determining the Relevant Number of Clusters in a Data Set". In: *Journal of Statistical Software, Articles* 61.6, pp. 1–36. DOI: 10.18637/jss.v061.i06. URL: https://www.jstatsoft.org/v061/i06.

Coates, A. and A.Y. Ng (2012). "Learning Feature Representations with K-Means". In: *Montavon G., Orr G.B., Müller KR. (eds) Neural Networks: Tricks of the Trade* 7700. DOI: https://doi.org/10.1007/978-3-642-35289-8_30.

Coffey, Kimberly (2016). *k-means Clustering for Customer Segmentation: A Practical Example.* URL: http://web.archive.org/web/20080207010024/http://www.808multimedia.com/winnt/kernel.htm (visited on 09/05/2020).

Cox, D. R. (2001). "[Statistical Modeling: The Two Cultures]: Comment." In: *Statistical Science* 16.3, 216–218. DOI: www.jstor.org/stable/2676682.

Daly, Kevin (Apr. 2008). "Financial Volatility: Issues and Measuring Techniques". In: *Physica A: Statistical Mechanics and its Applications* 387, pp. 2377–2393. DOI: 10.1016/j.physa.2008.01.009.

Daly, Kevin James (2011). "An Overview of the Determinants of Financial Volatility: An Explanation of Measuring Techniques". In: *Modern Applied Science* 5(5). DOI: 10.5539/mas.v5n5p46.

Davenport, Thomas H. and D.J. Patil (Oct. 2021). “Data Scientist: The Sexiest Job of the 21st Century”. In: *Havard Business Review.* URL: `https://hbr.org/2012/10/data-scientist-the-sexiest-job-of-the-21st-century` (visited on 27/01/2021).

Davison, A. C. and D. V. Hinkley (1997). *Bootstrap Methods and their Application.* Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press. DOI: `10.1017/CBO9780511802843`.

Demuth, Howard B., Mark H. Beale, Orlando De Jess and Martin T. Hagan (2014). *Neural Network Design.* 2nd ed. USA: Martin Hagan. ISBN: 0971732116, 9780971732117.

Do, Chuong B. and Serafim Batzoglou (2008). “What is the expectation maximization algorithm?” In: *Nature Biotechnology* 26.8, pp. 897–899. ISSN: 1546-1696. DOI: `10.1038/nbt1406`. URL: `https://doi.org/10.1038/nbt1406`.

Donaldson, R.Glen and Mark Kamstra (1997). “An artificial neural network-GARCH model for international stock return volatility”. In: *Journal of Empirical Finance* 4.1, pp. 17 –46. ISSN: 0927-5398. DOI: `https://doi.org/10.1016/S0927-5398(96)00011-4`. URL: `http://www.sciencedirect.com/science/article/pii/S0927539896000114`.

Donoho, David (2017). “50 Years of Data Science”. In: *Journal of Computational and Graphical Statistics* 26.4, pp. 745–766. DOI: `10.1080/10618600.2017.1384734`. eprint: `https://doi.org/10.1080/10618600.2017.1384734`. URL: `https://doi.org/10.1080/10618600.2017.1384734`.

Duignan, Brian (2018). “Occam’s razor (https://www.britannica.com/topic/Occams-razor)”. In: URL: `https://www.britannica.com/topic/Occams-razor/`.

Efron, Bradley (2006). “[Tukey’s Paper after 40 Years]: Discussion”. In: *Technometrics* 48.3, pp. 330–332. ISSN: 00401706. URL: `http://www.jstor.org/stable/25471203`.

Efron, D. R. (2001). “[Statistical Modeling: The Two Cultures]: Comment.” In: *Statistical Science* 16.3, 218–219. DOI: `https://doi.org/10.1214/ss/1009213726`.

Engle, R. F. (1982). “Autoregressive conditional heteroscedasticity with estimates of the variance of United Kingdom inflations”. In: *Econometrica* 50, 987–1007.

Everitt, B.S., S. Landau, M. Leese and D. Stahl (2011). *Cluster Analysis.* Wiley Series in Probability and Statistics. Wiley. ISBN: 9780470978443. URL: `https://books.google.co.za/books?id=w3bE1kqd-48C`.

Figlewski, Stephen (1997). *Financial Markets, Institutions & Instruments.* Vol. 6. Blackwell Publishers.

Fisher, Rory A. (1936). “The Use Of Multiple Measurements In Taxonomic Problems”. In: *Annals of Human Genetics* 7, pp. 179–188.

Forsberg, Lars and Tim Bollerslev (2002). “Bridging the Gap Between The Distribution Of Realized (ECU) Volatility and ARCH Modelling (Of Euro): The GARCH-NIG Model”. In: *Journal Of Applied Econometrics* 17, 535–548. DOI: `10.1002/jae.685`.

Fraley, C. and A. E. Raftery (1998). “How Many Clusters? Which Clustering Method? Answers Via Model-Based Cluster Analysis”. In: *The Computer Journal* 41.8, pp. 578–588.

Franses, Philip Hans and Dick van Dijk (1996). “Forecasting stock market volatility using (non-linear) Garch models”. In: vol. 15, pp. 229–235.

Franzén, Jessica (May 2008). “Bayesian Cluster Analysis : Some Extensions to Non-standard Situations”. PhD dissertation. PhD thesis. Statistiska institutionen, Stockholm.

Gao, P., R. Zhang and X. Yang (2020). “The Application of Stock Index Price Prediction with Neural Network.” In: *Mathematical and Computational Applications.* 25(3). DOI: `https://doi.org/10.3390/mca25030053`.

Ghalanos, Alexios (2019). *Introduction to the rugarch package.* R package version 1.4-1.

Ghouse, Ghulam, Saud Khan, Ahmed and Kashif Habeeb (Jan. 2019). "Information Transmission Among Equity Markets: A Comparison Between ARDL and GARCH Model". In: *Munich Personal RePEc Archive, MPRA* 97925. URL: https://mpra.ub.uni-muenchen.de/97925/.

Grün, Bettina (2018). *Model-based Clustering.* arXiv: 1807.01987 [stat.ME].

Hand, David J. (2000). "Data Mining: New Challenges for Statisticians". In: *Social Science Computer Review* 18.4, pp. 442–449. DOI: 10.1177/089443930001800407. URL: https://doi.org/10.1177/089443930001800407.

Hansen, P. R., Z. Huang and H. H. Shek (2012). "Realized garch: a joint model for returns and realized measures of volatility". In: *Journal of Applied Econometrics* 27(6), 877–906.

Hansen, Peter, Asger Lunde and James Nason (2003). "Choosing the Best Volatility Models: The Model Confidence Set Approach*". In: *Oxford Bulletin of Economics and Statistics* 65.s1, pp. 839–861.

Hansen, Peter R. and Asger Lunde (2005). "A forecast comparison of volatility models: does anything beat a GARCH(1,1)?" In: *Journal of Applied Econometrics* 20.7, pp. 873–889. DOI: https://doi.org/10.1002/jae.800. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/jae.800. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/jae.800.

Hastie, Trevor, Gareth James, Daniela Witten and Robert Tibshirani (2013). *An introduction to statistical learning : with applications in R.* 1st ed. New York : Springer, © 2013. ISBN: 978-1-4614-7138-7. DOI: https://doi.org/10.1007/978-1-4614-7138-7.

Hastie, Trevor, Robert Tibshirani and Jerome Friedman (2001). *The Elements of Statistical Learning.* Springer Series in Statistics. New York, NY, USA: Springer New York Inc.

He, Haibo and Edwardo A. Garcia (2009). "Learning from Imbalanced Data". In: *IEEE Transactions on Knowledge and Data Engineering* 21.9, pp. 1263–1284. DOI: 10.1109/TKDE.2008.239.

Heckert, Nathanael, James Filliben, C Croarkin, B Hembree, William Guthrie, P Tobias and J Prinz (Nov. 2002). *Handbook 151: NIST/SEMATECH e-Handbook of Statistical Methods.* en. NIST Interagency/Internal Report (NISTIR), National Institute of Standards and Technology, Gaithersburg, MD. URL: https://www.itl.nist.gov/div898/handbook/.

Hennig, Christian (2007). "Cluster-wise assessment of cluster stability". In: *Computational Statistics & Data Analysis* 52.1, pp. 258–271. ISSN: 0167-9473. DOI: https://doi.org/10.1016/j.csda.2006.11.025. URL: https://www.sciencedirect.com/science/article/pii/S0167947306004622.

— (Dec. 2020). *Flexible Procedures for Clustering.* URL: https://cran.r-project.org/web/packages/fpc/index.html.

Hill, Chelsey and B. D. McCullough (2019). "On The Accuracy of GARCH Estimation in R Packages". In: *Econometric Research in Finance* 4.2, pp. 133–156.

Hinkley, David V. (1988). "Bootstrap Methods". In: *Journal of the Royal Statistical Society. Series B (Methodological)* 50(3), 321–337. URL: http://www.jstor.org/stable/2345698.

Hossain, Altaf and Mohammed Nasser (2011). "Comparison of the finite mixture of ARMA-GARCH, back propagation neural networks and support-vector machines in forecasting financial returns". In: *Journal of Applied Statistics* 38.3, pp. 533–551. DOI: 10.1080/02664760903521435. eprint: https://doi.org/10.1080/02664760903521435. URL: https://doi.org/10.1080/02664760903521435.

Hossain, Altaf, Faisal Zaman, M. Nasser and M. Mufakhkharul Islam (2009). "Comparison of GARCH, Neural Network and Support Vector Machine in Financial Time Series Prediction". In: PReMI '09, pp. 597–602. DOI: 10.1007/978-3-642-11164-8_97. URL: https://doi.org/10.1007/978-3-642-11164-8_97.

Huang, Hanwen, Yufeng Liu and J. S. Marron (Feb. 2015). *Statistical Significance of Clustering.* URL: https://cran.r-project.org/web/packages/sigclust/index.html.

Huber, Peter J. (2006). "[Tukey's Paper after 40 Years]: Discussion". In: *Technometrics* 48.3, pp. 332–334. ISSN: 00401706. URL: http://www.jstor.org/stable/25471204.

Institute, S.A.S. (Sept. 2017). *SAS/STAT® 14.3 User's Guide.* Vol. 14.3. Sas Inst. URL: https://documentation.sas.com/api/collections/pgmsascdc/9.4_3.3/docsets/statug/content/introclus.pdf?locale=en#nameddest=statug_introclus_sect010.

Johnson, D. H. (1999). "The Insignificance Of Significance Testing". In: *Journal Of Wildlife Management* 63(3), pp. 763–772.

Kassambara, A. (2017). *Practical Guide to Cluster Analysis in R: Unsupervised Machine Learning.* Multivariate Analysis. STHDA. ISBN: 9781542462709. URL: https://books.google.co.za/books?id=-q3snAAACAAJ.

Kayri, Murat (2016). "Predictive Abilities of Bayesian Regularization and Levenberg–Marquardt Algorithms in Artificial Neural Networks: A Comparative Empirical Study on Social Data". In: *Mathematical and Computational Applications* 21 (20).

Kettenring, Jon R. (2012). "Statistics Research at Bell Labs in the Regulated Monopoly Era". In: *International Statistical Review / Revue Internationale de Statistique* 80.2, pp. 205–218. ISSN: 03067734, 17515823. URL: http://0-www.jstor.org.oasis.unisa.ac.za/stable/23258953.

Laily, Vania Orva Nur, Budi Warsito and Di Asih I Maruddani (2018). "Comparison of ARCH / GARCH model and Elman Recurrent Neural Network on data return of closing price stock". In: *Journal of Physics* 1025 (2018) 012103. DOI: 10.1088/1742-6596/1025/1/012103.

Lesniak, J M, R Hupse, R Blanc, N Karssemeijer and G Székely (2012). "Comparative evaluation of support vector machine classification for computer aided detection of breast masses in mammography". In: *Physics in Medicine and Biology* 57.16, pp. 5295–5307. DOI: 10.1088/0031-9155/57/16/5295. URL: https://doi.org/10.1088%2F0031-9155%2F57%2F16%2F5295.

Lin, Hui and Ming Li (2021). *Introduction to Data Science.* URL: https://scientistcafe.com/ids/ (visited on 27/01/2022).

Linzer, Drew (Jan. 2014). "Polytomous variable Latent Class Analysis". In: *CRAN.* URL: https://cran.r-project.org/web/packages/poLCA/poLCA.pdf.

Lui, Zhihui (Aug. 2010). "BAYESIAN MIXTURE· MODELS". Master's Thesis. MA thesis. School of Graduate Studies,McMaster University.

Madhanagopal, R., R. C. Avinaash and Kumaravel Karthick (2012). "Comparison of Support Vector Machines and Linear Discriminant Analysis for Indian Industries". In.

Magidson, J. and J.K. Vermunt (2002). "Latent class models for clustering: a comparison with K-means". English. In: *Canadian Journal of Marketing Research* 20.1, pp. 36–43. ISSN: 0829-4836.

Mallows, Colin L. (2006). "Tukey's Paper After 40 Years". In: *Technometrics* 48.3, pp. 319–325. URL: https://doi.org/10.1198/004017006000000219.

Manly, B.F.J. and J.A. Navarro-Alberto (2016). *Multivariate Statistical Methods: A Primer, Fourth Edition (4th ed.)* 4th ed. Chapman and Hall/CRC, pp. 139–157. DOI: https://doi.org/10.1201/9781315382135.

Marr, Bernard (2018). *How Much Data Do We Create Every Day? The Mind-Blowing Stats Everyone Should Read.* https://www.forbes.com/sites/bernardmarr/2018/05/21/how-much-data-do-we-create-every-day-the-mind-blowing-stats-everyone-should-read/. (Visited on 05/06/2020).

Mazanec, J., M. Melisêk, M. Oravec and J. Pavlovičová (2008). "Support Vector Machines, PCA AND LDA IN Face Recognition". In: *Journal of ELECTRICAL ENGINEERING* 59(4), pp. 203–209.

Mazur, Matt (2015). *A Step by Step Backpropagation Example.* URL: `https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/` (visited on 27/05/2020).

Megahan, Justin (2016). "This is the difference between statistics and data science". In: URL: `https://mixpanel.com/blog/2016/03/30/this-is-the-difference-between-statistics-and-data-science/`.

Meilă, Marina and David Heckerman (2001). "An Experimental Comparison of Model-Based Clustering Methods". In: *Machine Learning* 42.1, pp. 9–29. ISSN: 1573-0565. DOI: `10.1023/A:1007648401407`. URL: `https://doi.org/10.1023/A:1007648401407`.

Meyer, David, Evgenia Dimitriadou, Kurt Hornik, Andreas Weingessel, Friedrich Leisch, Chih-Chung Chang and Chih-Chen Lin (Sept. 2021). *Misc Functions of the Department of Statistics, Probability Theory Group (Formerly: E1071), TU Wien.* URL: `https://cran.r-project.org/web/packages/e1071/index.html`.

Ng, Andrew (2018). "CS229 Lecture notes:Gaussian Discriminant Analysis. Naive Bayes.(PDF notes)". In: URL: `http://cs229.stanford.edu/syllabus.html` (visited on 09/05/2020).

Nikitidis, S., S. Zafeiriou and M. Pantic (2014). "Merging SVMs with Linear Discriminant Analysis: A Combined Model". In: *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1067–1074. DOI: `10.1109/CVPR.2014.140`.

Nongxa, Loyiso G. (Apr. 2017). "Mathematical and statistical foundations and challenges of (big) data sciences". en. In: *South African Journal of Science* 113, pp. 1 –4. ISSN: 0038-2353. URL: `http://www.scielo.org.za/scielo.php?script=sci_arttext&pid=S0038-23532017000200006&nrm=iso`.

Obi, Jude Chukwura (2017). "A Comparative Study of the Fisher's Discriminant Analysis and Support Vector Machines". In: *European Journal of Engineering Research and Science* 2, pp. 35–40.

Peña, J. M., J. Lozano and P. Larrañaga (1999). "An empirical comparison of four initialization methods for the K-Means algorithm". In: *Pattern Recognit. Lett.* 20, pp. 1027–1040.

Prasad, N., R. Singh and S. P. Lal (2013). "Comparison of Back Propagation and Resilient Propagation Algorithm for Spam Classification". In: *2013 Fifth International Conference on Computational Intelligence, Modelling and Simulation*, pp. 29–34. DOI: `10.1109/CIMSim.2013.14`.

Riedmiller, Martin (1994). "Rprop - Description and Implementation Details. Technical Report." In: *University of Karlsruhe.*

Santos, Frédéric, Pierre Guyomarc'h and Jaroslav Bruzek (2014). "Statistical sex determination from craniometrics: Comparison of linear discriminant analysis, logistic regression, and support vector machines". In: *Forensic Science International* 245, 204.e1 –204.e8. ISSN: 0379-0738. DOI: `https://doi.org/10.1016/j.forsciint.2014.10.010`. URL: `http://www.sciencedirect.com/science/article/pii/S0379073814004162`.

Schlegel, Aaron (2016). "Discriminant Analysis of Several Groups". In: *R bloggers.* URL: `https://www.r-bloggers.com/2016/12/discriminant-analysis-of-several-groups/`.

Scrucca, Luca (Nov. 2021). *A quick tour of mclust.* URL: `https://cran.r-project.org/web/packages/mclust/vignettes/mclust.html`.

Scrucca, Luca, Michael Fop, T. Brendan Murphy and Adrian E. Raftery (2016). “mclust 5: Clustering, Classification and Density Estimation Using Gaussian Finite Mixture Models”. eng. In: *The R journal* 8.1. 27818791[pmid], pp. 289–317. ISSN: 2073-4859. URL: `https://pubmed.ncbi.nlm.nih.gov/27818791`.

Shao, Xiaolong, Hui Li, Nan Wang and Qiang Zhang (2015). “Comparison of different classification methods for analyzing electronic nose data to characterize sesame oils and blends”. In: *Sensors (Basel, Switzerland)* 15.10, 26726—26742. ISSN: 1424-8220. DOI: `10.3390/s151026726`. URL: `https://europepmc.org/articles/PMC4634481`.

Stapor, Katarzyna (2016). “A critical comparison of discriminant analysis and svm-based approaches to credit scoring”. In.

Suhandy, D. and M. Yulia (2018). “Luwak Coffee Classification Using UV-Vis Spectroscopy Data: Comparison of Linear Discriminant Analysis and Support Vector Machine Methods”. In: *Aceh International Journal of Science and Technology* 7, pp. 115–121.

Sullivan, A. and W. S. Gilbert (1885). “The Mikado”. In: *The Mikado Act I Part V a.* URL: `https://en.wikisource.org/wiki/The_Mikado/Act_I/Part_Va/` (visited on 21/08/2020).

Thiesson, Bo, Chris Meek, David Maxwell Chickering and David Heckerman (1999). *Computationally Efficient Methods For Selecting Among Mixtures Of Graphical Models, With Discussion.* Bayesian Statistics 6: Proceedings of the Sixth Valencia International Meeting. Editors: Bernardo, J., Berger, J., Dawid, A., and Smith, A., pp. 631–656. URL: `https://www.microsoft.com/en-us/research/publication/computationally-efficient-methods-for-selecting-among-mixtures-of-graphical-models-with-discussion/`.

Tsamardinos, Ioannis, Elissavet Greasidou and Giorgos Borboudakis (2018). “Bootstrapping the out-of-sample predictions for efficient and accurate cross-validation”. In: *Machine Learning* 107.12, pp. 1895–1922. ISSN: 1573-0565. DOI: `10.1007/s10994-018-5714-4`. URL: `https://doi.org/10.1007/s10994-018-5714-4`.

Tsay, R.S. (2010). *Analysis of Financial Time Series.* 3rd ed. John Wiley & Sons, Hoboken.

Tsay, Ruey S. (2013). *An Introduction to Analysis of Financial Data with R.* 1st ed. Wiley Publishing. ISBN: 0470890819.

Tukey, John W. (Mar. 1962). “The Future of Data Analysis”. In: *Ann. Math. Statist.* 33.1, pp. 1–67. DOI: `10.1214/aoms/1177704711`. URL: `https://doi.org/10.1214/aoms/1177704711`.

Uddin, Muhammad Fahim (2019). “Addressing Accuracy Paradox Using Enhanched Weighted Performance Metric in Machine Learning”. In: *2019 Sixth HCT Information Technology Trends (ITT)*, pp. 319–324. DOI: `10.1109/ITT48889.2019.9075071`.

Vinay, A., Vinay S. Shekhar, K.N. Balasubramanya Murthy and S. Natarajan (2015). “Performance Study of LDA and KFA for Gabor Based Face Recognition System”. In: *Procedia Computer Science* 57. 3rd International Conference on Recent Trends in Computing 2015 (ICRTC-2015), pp. 960 – 969. ISSN: 1877-0509. DOI: `https://doi.org/10.1016/j.procs.2015.07.493`. URL: `http://www.sciencedirect.com/science/article/pii/S1877050915020220`.

Wang, Weijie and Yanmin Lu (2018). “Analysis of the Mean Absolute Error (MAE) and the Root Mean Square Error (RMSE) in Assessing Rounding Model”. In: 324, p. 012049. DOI: `10.1088/1757-899x/324/1/012049`. URL: `https://doi.org/10.1088/1757-899x/324/1/012049`.

Wang, Zhiqiang, Catherine Da Cunha, Mathieu Ritou and Benoît Furet (2018). “Comparison of K-means and GMM methods for contextual clustering in HSM”. In: *Procedia Manufacturing* 28, pp. 154–159.

Wielki, Janusz (2016). “The Role of Internet of Things as a Key Technology Enabling the Fourth Industrial Revolution”. In: *SIMPRO 2016 Conference Proceedings*. Opole, Poland: Petrosani: Universitas Publishing House.

Wu, Binghui and Tingting Duan (2017). “A Performance Comparison of Neural Networks in Forecasting Stock Price Trend”. In: *International Journal of Computational Intelligence Systems* 10 (1), pp. 336–346. ISSN: 1875-6883. DOI: `https://doi.org/10.2991/ijcis.2017.10.1.23`. URL: `https://doi.org/10.2991/ijcis.2017.10.1.23`.

Xu, Taosheng (June 2016). “Cancer subtypes identification, validation and visualization based on multiple genomic data sets”. In: *Bioconductor*. URL: `https://bioconductor.org/packages/release/bioc/html/CancerSubtypes.html`.

Yap, B. W. and C. H. Sim (2011). “Comparisons of various types of normality tests”. In: *Journal of Statistical Computation and Simulation* 81.12, pp. 2141–2155. DOI: `10.1080/00949655.2010.520163`. eprint: `https://doi.org/10.1080/00949655.2010.520163`. URL: `https://doi.org/10.1080/00949655.2010.520163`.

Yim, Juliana (June 2002). “A Comparison of Neural Networks with Time Series Models for Forecasting Returns on a Stock Market Index”. In: pp. 25–35. DOI: `10.1007/3-540-48035-8_4`.

Yu, Youjiao (May 2021). “Finite Mixture Modeling for Raw and Binned Data”. In: *CRAN*. URL: `https://cran.r-project.org/web/packages/mixR/mixR.pdf`.

Zhang, B., M. Hsu and U. Dayal (1999). “K-harmonic means – a data clustering algorithm”. In: *Hewlett-Packard Labs* Technical Report HPL-1999-124.

Zhong, Xiao and David Enke (2019). “Predicting the daily return direction of the stock market using hybrid machine learning algorithms”. In: *Financial Innovation* 5.1, p. 24. ISSN: 2199-4730. DOI: `10.1186/s40854-019-0138-0`. URL: `https://doi.org/10.1186/s40854-019-0138-0`.

Appendix A

Prediction Case Study Appendices

A.1 Analysis Results

Table A.1: Ljung Box Test

	Lag Order	X-squared	p-value
ALSI Prices	13	32028	2.2e-16
Log Returns	13	18.461	0.1408

A.2 Mean equation models

Table A.2: ARMA models of the log return series

	ARIMA(2,0,2) with non-zero mean				
Coefficients:					
	ar1	ar2	ma1	ma2	mean
	0.2033	0.5857	-0.2125	-0.6423	4e-04
s.e.	0.3141	0.2922	0.3044	0.2901	1e-04
$sigma^2 \approx 0.0001011$		log likelihood=7993.12			
AIC=-15974.25		AICc=-15974.21		BIC=-15939.27	
	ARIMA(2,0,1) with non-zero mean				
Coefficients:					
	ar1	ar2	ma1	mean	
	0.8892	-0.0295	-0.9019	4e-04	
s.e.	0.0598	0.0220	0.0568	1e-04	
$sigma^2 \approx 0.0001013$		log likelihood=7988.63			
AIC=-15973.77		AICc=-15973.75		BIC=-15944.63	

A.3 GARCH Model Fitting

A.3.1 Fitted RealGARCH Models

Table A.3: ARMA(2,2)-RealGARCH(1,1)

Conditional Variance Dynamics

GARCH Model : RealGARCH(2,1)
Mean Model : ARFIMA(1,0,1)
Distribution : norm

Optimal Parameters

	Estimate	Std. Error	t value	p-value
~~**mu**~~	~~**-0.000356**~~	~~**0.000348**~~	~~**-1.0249e+00**~~	~~**0.30540**~~
~~**mu**~~	~~**-0.000343**~~	~~**0.000374**~~	~~**-0.916430**~~	~~**0.35944**~~
~~**ar1**~~	~~**0.089206**~~	~~**0.658303**~~	~~**0.135509**~~	~~**0.89221**~~
~~**ar2**~~	~~**0.639833**~~	~~**0.557354**~~	1. ~~**147983**~~	~~**0.25098**~~
~~**ma1**~~	~~**0.053388**~~	~~**0.661829**~~	0. ~~**080667**~~	~~**0.93571**~~
~~**ma2**~~	~~**-0.538225**~~	~~**0.**~~	~~**-1.146050**~~	~~**0.25178**~~
omega	-9.000000	0.180421	-49.883288	0.00000
alpha1	0.825307	0.021646	38.126790	0.00000
beta1	0.004935	0.019534	0.252631	0.80055
~~**eta11**~~	~~**-0.004545**~~	~~**0.005916**~~	~~**-0.768278**~~	~~**0.44232**~~
eta21	0.098825	0.003769	26.223372	0.00000
delta	1.023950	0.025928	39.492349	0.00000
lambda	0.223681	0.003246	68.902819	0.00000
xi	9.225523	0.242812	37.994462	0.00000

LogLikelihood : 8228.213

Information Criteria

Akaike	-6.5304
Bayes	-6.5003
Shibata	-6.5304
Hannan-Quinn	-6.5194

Weighted Ljung-Box Test on Standardized Residuals

	statistic	p-value
Lag[1]	17.38	3.056e-05
Lag[2*(p+q)+(p+q)-1][8]	59.12	0.000e+00
Lag[4*(p+q)+(p+q)-1][14]	65.41	0.000e+00

Weighted Ljung-Box Test on Standardized Squared Residuals

	statistic	p-value
Lag[1]	0.0688	0.793098
Lag[2*(p+q)+(p+q)-1][5]	1.6678	0.698634
Lag[4*(p+q)+(p+q)-1][9]	14.7697	0.004065

Weighted ARCH LM Tests

	Statistic	Shape	Scale	P-Value
ARCH Lag[3]	0.9286	0.500	2.000	0.33523
ARCH Lag[5]	2.5355	1.440	1.667	0.36460
ARCH Lag[8]	12.4681	2.315	1.543	0.00479

Sign Bias Test

	t-value	p-value
Sign Bias	0.2245	0.8223623
Negative Sign Bias	1.9131	0.0558529
Positive Sign Bias	1.7303	0.0837031
Joint Effect	17.3865	0.0005885

Table A.4: ARMA(2,1)-RealGARCH(2,1)

Conditional Variance Dynamics				
GARCH Model : RealGARCH(2,1)				
Mean Model : ARFIMA(2,0,1)				
Distribution : norm				

Optimal Parameters				
	Estimate	Std. Error	t value	p-value
~~**mu**~~	~~**-0.000324**~~	~~**0.000365**~~	~~**-0.88693**~~	~~**0.375117**~~
ar1	0.783942	0.051570	15.20145	0.000000
~~**ar2**~~	~~**0.035152**~~	~~**0.022402**~~	~~**1.56913**~~	~~**0.116619**~~
ma1	-0.661830	0.047314	-13.98808	0.000000
omega	-9.000000	2.923095	-3.07893	0.002077
alpha1	0.826682	0.022172	37.28482	0.000000
~~**alpha2**~~	~~**0.000000**~~	~~**0.280192**~~	~~**0.00000**~~	~~**1.000000**~~
~~**beta1**~~	~~**0.004633**~~	~~**0.323516**~~	~~**0.01432**~~	~~**0.988575**~~
~~**eta11**~~	~~**-0.004047**~~	~~**0.005910**~~	~~**-0.68473**~~	~~**0.493513**~~
eta21	0.098955	0.003769	26.25830	0.000000
delta	1.022756	0.026588	38.46701	0.000000
lambda	0.223307	0.003241	68.89355	0.000000
xi	9.211925	0.249117	36.97830	0.000000
LogLikelihood : 8229.218				

Information Criteria
Akaike -6.5312
Bayes -6.5010
Shibata -6.5312
Hannan-Quinn -6.5202

Weighted Ljung-Box Test on Standardized Residuals		
	statistic	p-value
Lag[1]	10.51	0.001189
Lag[2*(p+q)+(p+q)-1][8]	59.51	0.000000
Lag[4*(p+q)+(p+q)-1][14]	66.05	0.000000

Weighted Ljung-Box Test on Standardized Squared Residuals		
	statistic	p-value
Lag[1]	0.01574	9.002e-01
Lag[2*(p+q)+(p+q)-1][8]	9.95862	3.551e-02
Lag[4*(p+q)+(p+q)-1][14]	33.12502	1.290e-06

Weighted ARCH LM Tests				
	statistic	Shape	Scale	p-value
ARCH Lag[4]	0.6777	0.500	2.000	4.104e-01
ARCH Lag[6]	8.3063	1.461	1.711	1.981e-02
ARCH Lag[8]	21.1899	2.368	1.583	4.869e-05

Sign Bias Test		
	t-value	p-value
Sign Bias	0.263	0.792582
Negative Sign Bias	1.703	0.088706
Positive Sign Bias	1.716	0.086295
Joint Effect	15.745	0.001279

Table A.5: ARMA(2,2)-eGARCH(2,2)

Conditional Variance Dynamics

GARCH Model : eGARCH(2,2)
Mean Model : ARFIMA(2,0,2)
Distribution : norm

Optimal Parameters

	Estimate	Std. Error	t value	p-value
~~mu~~	~~0.000193~~	~~0.000180~~	~~1.07232~~	~~0.283574~~
ar1	-0.096020	0.039756	-2.41521	0.015726
~~ar2~~	~~0.119362~~	~~0.078732~~	~~1.51605~~	~~0.129507~~
ma1	0.113163	0.039798	2.84342	0.004463
ma2	-0.184576	0.078276	-2.35802	0.018373
omega	-0.233986	0.012600	-18.57000	0.000000
alpha1	-0.245109	0.026466	-9.26113	0.000000
alpha2	0.099027	0.027319	3.62483	0.000289
beta1	0.596626	0.000163	3664.48882	0.000000
beta2	0.378245	0.001725	219.30596	0.000000
~~gamma1~~	~~0.012897~~	~~0.039418~~	~~0.32719~~	~~0.743527~~
gamma2	0.144076	0.039519	3.64571	0.000267

LogLikelihood : 8242.709

Information Criteria

Akaike	-6.5427
Bayes	-6.5149
Shibata	-6.5427
Hannan-Quinn	-6.5326

Weighted Ljung-Box Test on Standardized Residuals

	statistic	p-value
Lag[1]	0.00859	0.9262
Lag[2*(p+q)+(p+q)-1][8]	2.31959	1.0000
Lag[4*(p+q)+(p+q)-1][14]	6.19328	0.9624

Weighted Ljung-Box Test on Standardized Squared Residuals

	statistic	p-value
Lag[1]	0.09706	0.7554
Lag[2*(p+q)+(p+q)-1][8]	7.50949	0.2604
Lag[4*(p+q)+(p+q)-1][14]	13.26248	0.1784

Weighted ARCH LM Tests

	statistic	Shape	Scale	p-value
ARCH Lag[4]	0.6638	0.500	2.000	0.41521
ARCH Lag[6]	4.9378	1.473	1.746	0.12502
ARCH Lag[8]	10.7427	2.402	1.619	0.01834

Sign Bias Test

	t-value	p-value
Sign Bias	0.8023	0.4224
Negative Sign Bias	1.0068	0.3141
Positive Sign Bias	0.1961	0.8445
Joint Effect	1.1102	0.7746

A.4 ANN model Fitting

A.4.1 No of hidden layer neurons vs Performance

Table A.6: No of hidden layer neurons vs Performance

Hidden layer neurons	3	4	5	6	7	8	9	10
MAE	0.14504355	0.14660644	0.1451221	0.14651724	0.14463381	0.14534083	0.14467150	0.14377992
MSE	0.04811708	0.04769322	0.0475964	0.04670646	0.04688395	0.04833844	0.04694797	0.04782738

A.4.1.1 ANN Fitted Model Output

Table A.7: Ouput Of ANN fitted model from **neuralnet** R package

	Length	Class	Mode
call	5	-none-	call
response	1758	-none-	numeric
covariate	3516	-none-	numeric
model.list	2	-none-	list
err.fct	1	-none-	function
act.fct	1	-none-	function
linear.output	1	-none-	logical
data	3	data.frame	list
exclude	0	-none-	NULL
net.result	1	-none-	list
weights	1	-none-	list
generalized.weights	1	-none-	list
startweights	1	-none-	list
result.matrix	28	-none-	numeric

A.5 GARCH and ANN Code Listings

A.5.1 GARCH Code Listing

```
library(rugarch)
library(xts)
library(forecast)
library(fUnitRoots)
library(FinTS)
library(fBasics)
library(nortest)
library(tsoutliers)
data <-read.csv(file="stock_prices3.csv", header=TRUE, sep=",")
manual_vol = read.csv(file="manual_vol3.csv", header=TRUE, sep=",")
date = as.Date(data$index)
xts_prices = xts(data$value,as.Date(data$index))
jse_data = xts(data$value,date)
returns = na.omit(diff(log(jse_data), 1))
vol = rollapplyr(returns, 5, sd, fill = NA)*100
#xts_vol = xts(manual_vol$value,as.Date(manual_vol$index))
basicStats(xts_prices)
basicStats(returns)

adfTest(coredata(xts_prices))
adfTest(coredata(returns))
#Shapiro-Wilk Normality Test
shapiro.test(coredata(returns))
shapiro.test(coredata(xts_prices))
#Anderson-Darling test for Normality
ad.test(coredata(returns))
ad.test(coredata(xts_prices))
#Jarque-Bera Test for Normality
jarque.bera.test(xts_prices)
jarque.bera.test(returns)
Box.test(returns, type="Ljung-Box", lag = 13)
Box.test(xts_prices, type="Ljung-Box", lag = 13)
xts_vol = xts(coredata(vol),as.Date(index(vol)))
value = coredata(xts_vol)
index = index(xts_vol)
xts_vol = xts(value,as.Date(index))
N = nrow(xts_vol)
n = round(N *0.7, digits = 0)
train_vol = xts_vol[1:n, ]
test_vol = xts_vol[(n+1):N, ]
#length(test)
N = nrow(returns)
n = round(N *0.7, digits = 0)
train_returns = returns[1:n, ]
test_returns = returns[(n+1):N, ]
xts_vol[is.na(xts_vol)] <- 1.895584
test_data = coredata(test)
```

```
auto.arima(returns)
model=ugarchspec (
  mean.model = list(armaOrder = c(2,2), include.mean = TRUE), variance.model = list(model =
      ↪ 'realGARCH', garchOrder = c(2,1))
)
count = 0
MAE_ = 0
RMSE_ = 0
for(i in 1:10){

N = nrow(xts_vol)
n = round(N *0.7, digits = 0)
train = xts_vol[1:n, ]
test = xts_vol[(n+1):N, ]
#length(test)
test_data = coredata(test)
# par(mfrow=c(1,2))
 plot(as.vector(t(as.matrix(test_data))), col = 'red', type='l', main = "realGARCH:␣Actual␣
     ↪ vs␣predicted:␣testing␣set", ylab = "Y,Yp")
 lines(as.vector(t(as.matrix(modelroll@forecast$density$RVolForecast))), type = 'l', col = '
     ↪ blue')
 legend("topright", c("Predicted", "Actual"), col = c("blue","red"), lty = c(1,1), lwd = c(1
     ↪ ,1))

modelroll <- ugarchroll (
  spec=model, data=returns, n.ahead = 1, forecast.length = 755, n.start = NULL, refit.every
      ↪ = 50, refit.window = c("recursive"),
  window.size = NULL, solver = "hybrid", fit.control = list(scale=1), solver.control = list
      ↪ (), calculate.VaR = FALSE, VaR.alpha = c(0.01,0.05),
  cluster = NULL, keep.coef = TRUE,realizedVol = xts_vol
)

MAE_ <- MAE_ + mean(abs(test_vol-modelroll@forecast$density$RVolForecast))
RMSE_ <- RMSE_ + sqrt(mean((test_vol - modelroll@forecast$density$RVolForecast)^2))
out <- paste0(MAE_,",",RMSE_,",", i, ".") # Some output
print(out)
}
MAE = MAE_/10
RMSE = RMSE_ /10
A <- matrix(c(MAE,RMSE),nrow=2,ncol = 1,byrow = TRUE)
dimnames(A) <- list( c("MAE", "RMSE"), c("realGARCH"))
A
```

A.5.2 ANN R Code Listing

```
library(tidyquant) # Loads tidyverse, tidquant, financial pkgs, xts/zoo
library(roll)
require(zoo)
library(neuralnet)
rm(list=ls())
require(rnn)
set.seed(10)
data <-read.csv(file="stock_prices3.csv", header=TRUE, sep=",")
#used for only extracting datetimes after rollarpply()
manual_vol = read.csv(file="manual_vol3.csv", header=TRUE, sep=",")
dates = as.Date(data$index,format = "%Y-%m-%d")
value = data$value
merged = cbind.data.frame(dates,value)
xts_data = xts(merged$value,merged$dates)
r = na.omit(diff(log(xts_data), 1))
# 5 day rolling standard deviation
vol = rollapplyr(r, 5, sd, fill = NA)*100
xts_vol = xts(vol,as.Date(manual_vol$index))
xts_vol = na.omit(xts_vol)
vol2 <-volatility(xts_data,n=5, calc = "close", N = 252,mean0=TRUE)
av_vol = mean(coredata(vol2[10:N]))
vol2 = na.fill(vol2,fill = av_vol)
vol_values <- coredata(vol2)
dates <- index(vol2)
N=length(xts_vol[,1])
n = round(N *0.7, digits = 0)
xts_vol_actual = xts_vol[(n+1):N, ]
Series = xts_vol
plot(xts_vol)
#Lagged dataset
lag_transform <- function(x, k= 1,j=2){

  lagged = c(rep(NA, k), x[1:(length(x)-k)])
  lagged2 = c(rep(NA, j), x[1:(length(x)-j)])
  DF = as.data.frame(cbind(lagged2,lagged, x))
  colnames(DF) <- c(paste0('x-',j), paste0('x-', k), 'x')
  DF[is.na(DF)] <- 0
  return(DF)
}
supervised = lag_transform(xts_vol, 1)
supervised[supervised==0] <- av_vol
supervised<-supervised[complete.cases(supervised),]
## split into train and test sets
supervised = as.data.frame(supervised)
type(supervised)
N = nrow(supervised)
N = nrow(Series)
n = round(N *0.7, digits = 0)
train = supervised[1:n, ]
```

```
test = supervised[(n+1):N, ]
train<-train[complete.cases(train),]
test<-test[complete.cases(test),]
write.table(test, "test_complete.txt", sep=",")
length(train)
length(test)
scale_data = function(train, test, feature_range = c(0, 1)) {
  x = train
  fr_min = feature_range[1]
  fr_max = feature_range[2]
  std_train = ((x - min(x) ) / (max(x) - min(x) ))
  std_test = ((test - min(x) ) / (max(x) - min(x) ))
  scaled_train = std_train *(fr_max -fr_min) + fr_min
  scaled_test = std_test *(fr_max -fr_min) + fr_min
  return( list(scaled_train = as.vector(scaled_train), scaled_test = as.vector(scaled_test)
      ↪ ,scaler= c(min =min(x), max = max(x))) )

}
Scaled = scale_data(train, test, c(-1, 1))
train_data <- Scaled$scaled_train
names_x_train <- names(train_data)[(names(train_data) %in% c("x-1","x"))]
train_x_variables <- train_data[, names_x_train]
names_y_train <- names(train_data)[(names(train_data) %in% c("x-2"))]
train_y_variables <- train_data[, names_y_train]
y_train = train_y_variables
x_train=train_x_variables
write.table(y_train, "xy_train.txt", sep=",")
write.table(x_train, "xx_train.txt", sep=",")
test_data <- Scaled$scaled_test
names_x_test <- names(test_data)[(names(test_data) %in% c("x-1","x"))]
test_x_variables <- test_data[, names_x_train]
names_y_test <- names(test_data)[(names(test_data) %in% c("x-2"))]
test_y_variables <- test_data[, names_y_test]
y_test = test_y_variables
x_test = test_x_variables
length(y_test)

invert_scaling = function(scaled, scaler, feature_range = c(0, 1)){
  min = scaler[1]
  max = scaler[2]
  t = length(scaled)
  mins = feature_range[1]
  maxs = feature_range[2]
  inverted_dfs = numeric(t)
  for( i in 1:t){
    X = (scaled[i]- mins)/(maxs - mins)
    rawValues = X *(max - min) + min
    inverted_dfs[i] <- rawValues
  }
  return(inverted_dfs)
}
# Reshape the input to 3-dim
dim(x_train) <- c(length(x_train), 1, 1)
# specify required arguments
X_shape2 = dim(x_train)[2]
X_shape3 = dim(x_train)[3]
batch_size = 1 # must be a common factor of both the train and test samples
units = 1 # can adjust this, in model tuninig phase

dim(y_train)
dim(x_train)
dim(y_train) <- c(length(y_train), 1, 1)
dframe = cbind.data.frame(y_train,x_train)
x_test = as.data.frame(x_test)
write.table(dframe, "NN_dframe.txt", sep=",")
names(dframe)[names(dframe) == "1"] <- "x1"
names(dframe)[names(dframe) == "2"] <- "x2"
NN = neuralnet(y_train ~ x1+x2 , dframe, hidden = 6 , linear.output = T )
plot(NN)
gwplot(NN)
length(x_test)
scaler = Scaled$scaler
predict_testNN = compute(NN, x_test)
plot(as.vector(t(as.matrix(y_test))), col = 'red', type='l', main = "Neural␣Net:␣Actual␣vs␣
    ↪ predicted:␣testing␣set", ylab = "Y,Yp")
lines(as.vector(t(as.matrix(predict_testNN$net.result))), type = 'l', col = 'blue')
legend("topright", c("Predicted", "Real"), col = c("blue","red"), lty = c(1,1), lwd = c(1,1)
    ↪ )
#x_test[1]
predict_testNN$net.result[1]
L = length(predict_testNN$net.result)
predictions = numeric(L)
for(i in 1:L){
  X = predict_testNN$net.result[i]
  dim(X) = c(1,1,1)
  yhat = X
    yhat = invert_scaling(yhat, scaler, c(-1, 1))
    predictions[i] <- yhat
}
L = length(y_test)
actual_values = numeric(L)
for(i in 1:L){
  X = y_test[i]
  dim(X) = c(1,1,1)
  yhat = X
  yhat = invert_scaling(yhat, scaler, c(-1, 1))
  actual_values[i] <- yhat
}
MAE.NN<-mean(abs(xts_vol_actual - predictions))
RMSE.NN <- sqrt(mean( (xts_vol_actual- predictions)^2))
```

```r
A <- matrix(c(MAE.NN,RMSE.NN),nrow=2,ncol = 1,byrow = TRUE)
dimnames(A) <- list( c("MAE", "MSE"), c("Neural␣Network"))
A
par(mfrow=c(1,2))
# Plot predicted vs actual. Testing set only.
plot(as.vector(t(as.matrix(xts_vol_actual))), col = 'red', type='l', main = "NN␣Actual␣vs␣
    ↪ predicted:␣testing␣set", ylab = "Y,Yp")
lines(as.vector(t(as.matrix(predictions))), type = 'l', col = 'blue')
legend("topright", c("Predicted", "Actual"), col = c("blue","red"), lty = c(1,1), lwd = c(1,
    ↪ 1))
```

Appendix B

Classification Case Study Appendices

B.1 Two-Classes Classification

B.1.1 QQ Plot for Normality Assumption chekcs -Channel

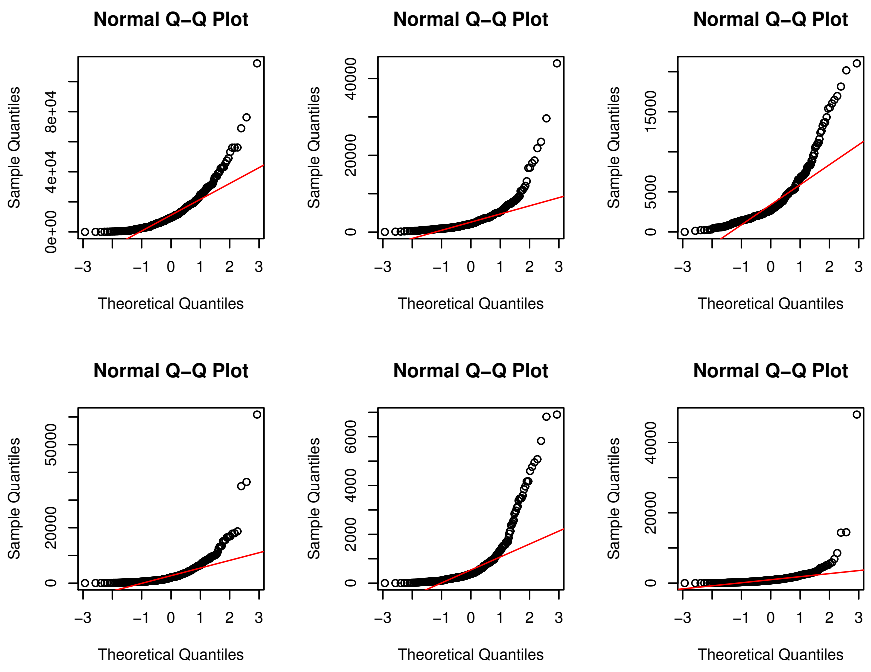


Figure B.1: QQ Plots Normal Check - Channel-Horeca

B.1.2 LDA Pairwise Plots Two Classes

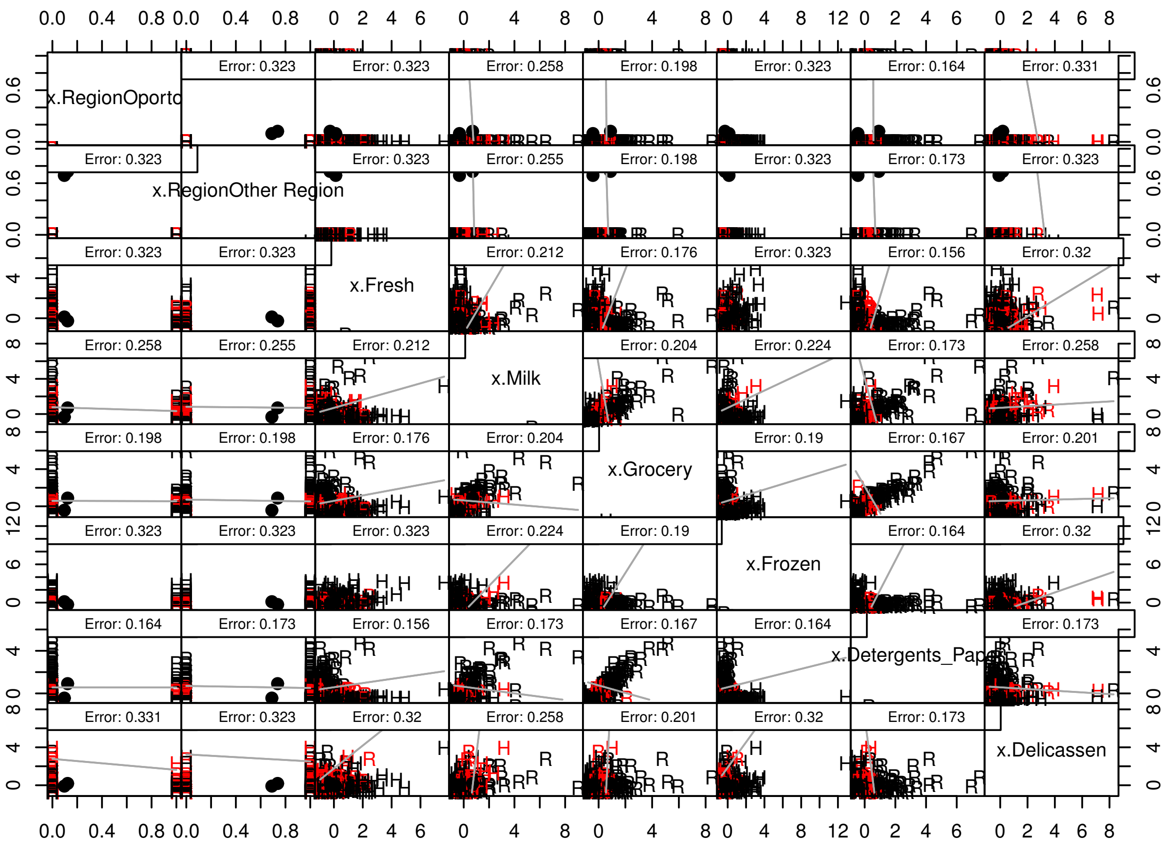


Figure B.2: LDA Pairwise Plots - Channel

B.1.3 F-Measure Calculation - Two Classes

Table B.1: Two-Class confusion matrix

		Actual	
		Positive Class	**Negative Class**
predicted	**Positive Class**	True Positive (TP)	FP
	Negative Class	FN	True Negative (TN)

$$Precision = \frac{TP}{TP + FP} \tag{B.1}$$

$$Recall = \frac{TP}{TP + FN} \tag{B.2}$$

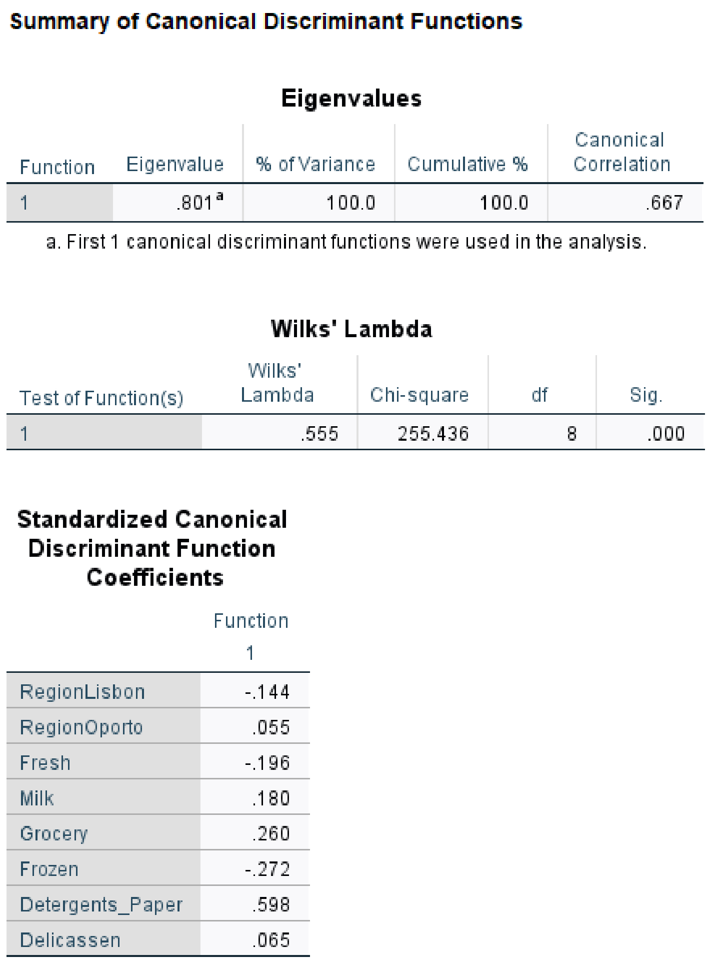
$$F\text{-}Measure = \frac{(1+\beta)^2 \times Precision \times Recall}{\beta^2 \times Recall + Precision}, (\ usually\ \beta = 1). \tag{B.3}$$

B.1.4 LDA Fitted Model - Two Classes

B.1.4.1 Unbalanced Data

Table B.2: LDA model : Two Classes

Prior probabilities of groups								
Horeca	Retail							
0.6770538	0.3229462							
Optimal Parameters								
	RegionLisbon	RegionOporto	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicassen
Horeca	0.2133891	0.09623431	0.1263303	-0.3335389	-0.4237690	0.1423064	-0.4375262	-0.08894369
Retail	0.1403509	0.12280702	-0.2648503	0.6992614	0.8884281	-0.2983442	0.9172698	0.18646967
Coefficients of linear discriminants								
		LD1						
RegionLisbon		-0.36825516						
RegionOporto		0.09834757						
Fresh		-0.26159407						
Milk		0.22051382						
Grocery		0.36398008						
Frozen		-0.28076916						
Detergents_Paper		0.69631048						
Delicassen		0.14703554						
Confusion Matrix								
		truth						
		Horeca	Retail					
predict	Horeca	57	11					
	Retail	2	17					
Classification Accuracy			85.0575%					



Summary of Canonical Discriminant Functions

Eigenvalues

Function	Eigenvalue	% of Variance	Cumulative %	Canonical Correlation
1	.801[a]	100.0	100.0	.667

a. First 1 canonical discriminant functions were used in the analysis.

Wilks' Lambda

Test of Function(s)	Wilks' Lambda	Chi-square	df	Sig.
1	.555	255.436	8	.000

Standardized Canonical Discriminant Function Coefficients

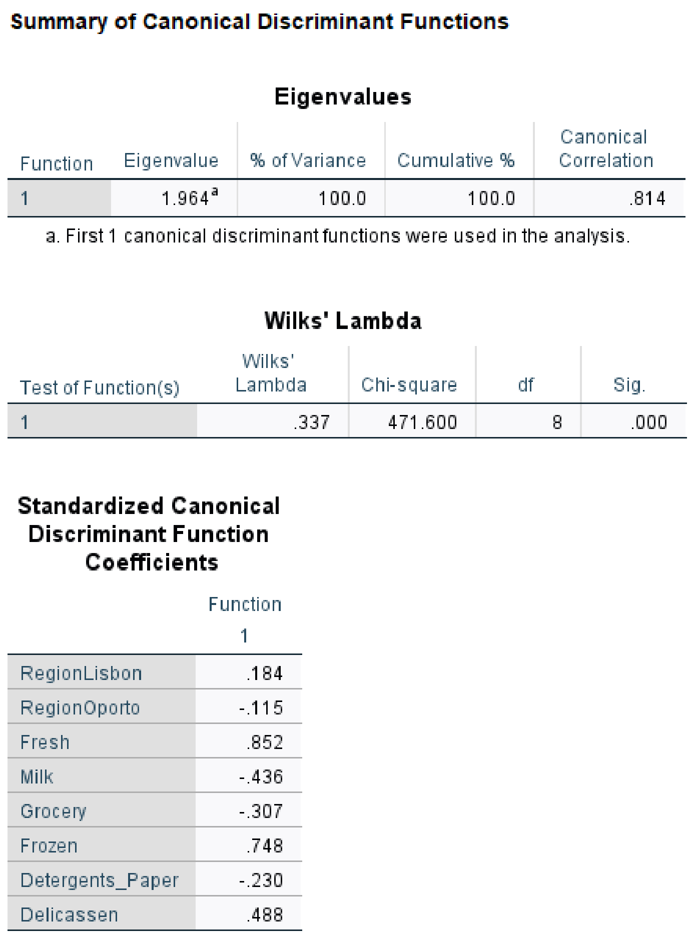
	Function
	1
RegionLisbon	-.144
RegionOporto	.055
Fresh	-.196
Milk	.180
Grocery	.260
Frozen	-.272
Detergents_Paper	.598
Delicassen	.065

Figure B.3: SPSS Significance Output

B.1.4.2 Balanced Data

Table B.3: LDA model : Two Classes

Prior probabilities of groups								
Horeca	Retail							
0.5	0.5							
Optimal Parameters								
	RegionLisbon	RegionOporto	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicassen
Horeca	0.09123295	0.1316801	-0.2127816	0.6478214	0.8727041	-0.2722088	0.9337072	0.02552776
Retail	0.02840909	0.2159091	2.5627823	1.9351193	0.4403386	4.5962050	-0.2760471	4.30064598
Coefficients of linear discriminants								
		LD1						
RegionLisbon		0.27164456						
RegionOporto		-0.24187559						
Fresh		0.38625587						
Milk		-0.22872166						
Grocery		-0.55457519						
Frozen		0.25607799						
Detergents_Paper		0.06904033						
Delicassen		0.12189443						
Confusion Matrix								
		truth						
		Horeca	Retail					
predict	Horeca	44	12					
	Retail	2	32					
Classification Accuracy			86.36364%					



Summary of Canonical Discriminant Functions

Eigenvalues

Function	Eigenvalue	% of Variance	Cumulative %	Canonical Correlation
1	1.964[a]	100.0	100.0	.814

a. First 1 canonical discriminant functions were used in the analysis.

Wilks' Lambda

Test of Function(s)	Wilks' Lambda	Chi-square	df	Sig.
1	.337	471.600	8	.000

Standardized Canonical Discriminant Function Coefficients

	Function
	1
RegionLisbon	.184
RegionOporto	-.115
Fresh	.852
Milk	-.436
Grocery	-.307
Frozen	.748
Detergents_Paper	-.230
Delicassen	.488

Figure B.4: SPSS Significance Output Two Groups - Balanced Data

B.1.5 SVM Fitted Model - Two Classes

B.1.5.1 Unbalanced Data

Table B.4: SVM model : Two Classes

SVM-Type	C-classification		
SVM-Kernel	linear		
cost	1		
gamma	1		
Number of Support Vectors		76	
Confusion Matrix			
		truth	
		Horeca	Retail
predict	Horeca	57	11
	Retail	2	17
Classification Accuracy		85.0575%	

B.1.5.2 Balanced Data

Table B.5: SVM model : Two Classes

SVM-Type	C-classification		
SVM-Kernel	linear		
cost	1		
gamma	1		
Number of Support Vectors		208	
Confusion Matrix			
		truth	
		Horeca	Retail
predict	Horeca	44	9
	Retail	0	35
Classification Accuracy		89.77273%	

B.1.6 Logistic Regression Fitted Model - Two Classes

Table B.6: Logistic Regression On Customer - Channel

Coefficients								
(Intercept)	x.RegionLisbon	x.RegionOporto	x.Fresh	x.Milk	x.Grocery	x.Frozen	x.Detergents_Paper	x.Delicassen
-0.26270	-1.58911	0.21536	-0.03791	1.09172	0.98655	-1.48200	4.50174	-0.06565
Degrees of Freedom (Total)		352						
Residual		344						
Null Deviance		444.1						
Residual Deviance		140.6						
AIC		158.6						
Confusion Matrix								
		truth						
		Horeca	Retail					
predict	Horeca	55	6					
	Retail	4	22					
Classification Accuracy			88.5058%					

B.2 Multi-Class Classification

B.2.1 QQ Plot for Normality Assumption chekcs -Region

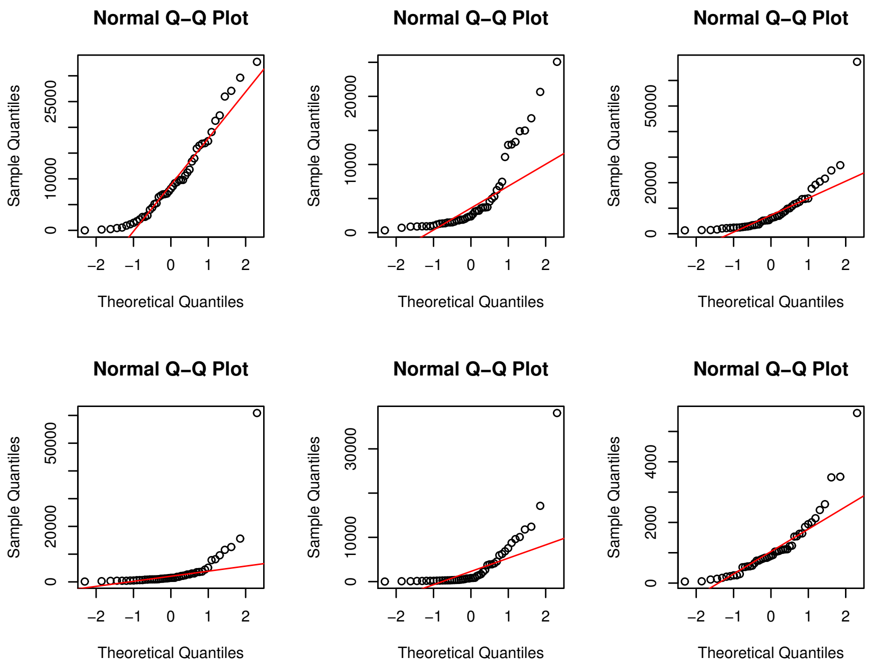


Figure B.5: QQ Plots Normal Check - Region-Oporto

B.2.2 Scatter Plots - Three Classes

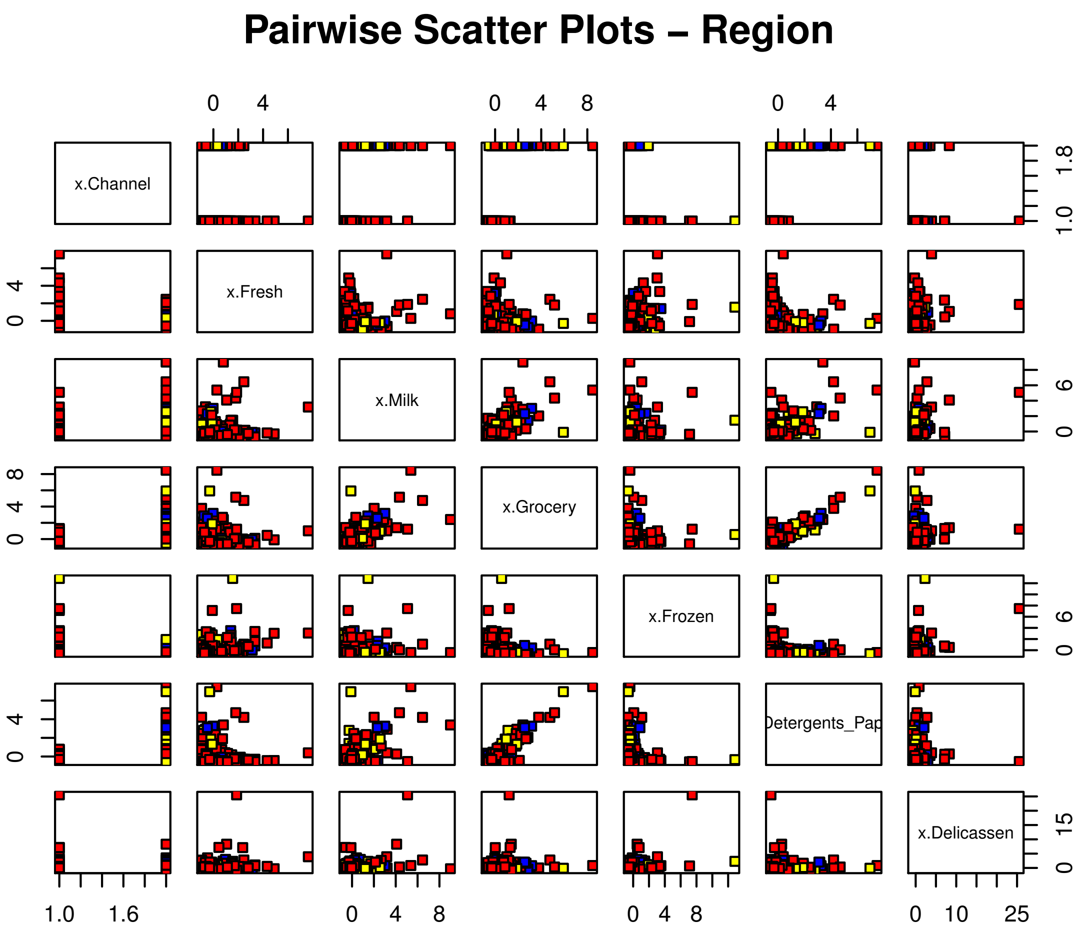


Figure B.6: Pairwise Scatter Plots - Region

B.2.3 SVM Fitted Model and Results - Three Classes

B.2.3.1 Unbalanced Data

Table B.7: SVM model : Three Classes

SVM-Type	C-classification			
SVM-Kernel	linear			
cost	0.1			
gamma	1			
Number of Support Vectors		272		
Confusion Matrix				
		truth		
		Lisbon	Oporto	Other Region
predict	Lisbon	0	0	0
	Oporto	0	0	0
	Other Region	10	10	67
Classification	**Accuracy**	77.0115%		

B.2.3.2 Balanced Data

Table B.8: SVM model : Three Classes - Balanced Data

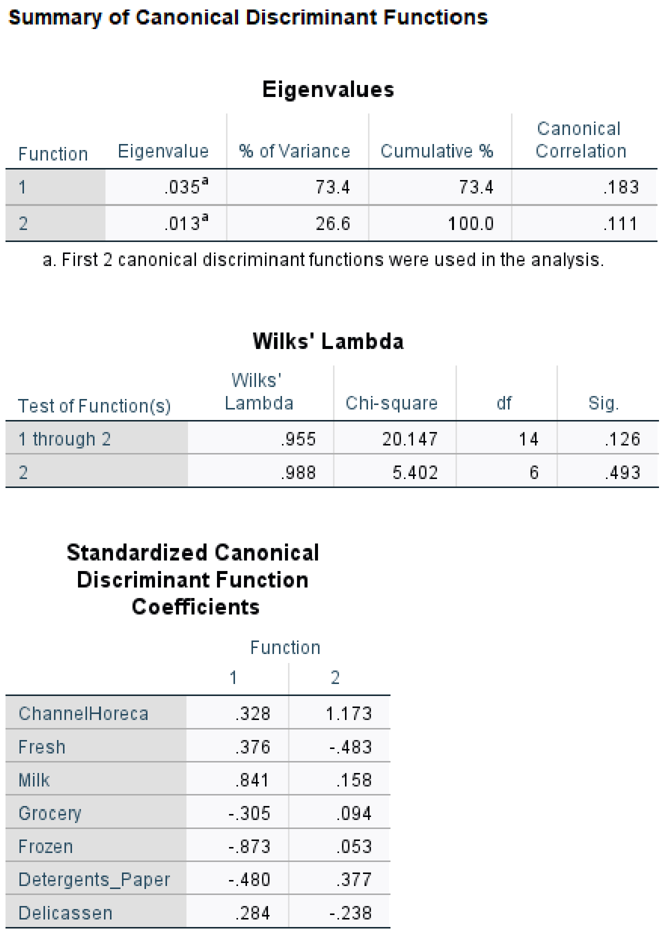
SVM-Type	C-classification			
SVM-Kernel	linear			
cost	0.1			
gamma	1			
Number of Support Vectors		281		
Confusion Matrix				
		truth		
		Lisbon	Oporto	Other Region
predict	Lisbon	24	15	3
	Oporto	3	12	1
	Other Region	2	2	25
Classification	**Accuracy**	70.11494%		

B.2.4 LDA Fitted Model and Results - Three Classes

B.2.4.1 Unbalanced Data

Table B.9: LDA model : Three Classes

Prior probabilities of groups							
Lisbon	Oporto	Other Region					
0.1898017	0.1048159	0.7053824					
Group means							
	ChannelRetail	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicassen
Lisbon	0.2388060	-0.05379723	-0.01524373	-0.052904436	-0.02149026	-0.067429761	-0.01531713
Oporto	0.3783784	-0.21133870	-0.14770483	0.154498000	0.36383501	0.175349964	-0.13196877
Other Region	0.3373494	0.04587930	0.02604984	-0.008722204	-0.04828132	-0.007912268	0.02373130
Coefficients of linear discriminants							
		LD1	LD2				
ChannelRetail		-0.3726638	-2.520312316				
Fresh		0.4196827	-0.583305870				
Milk		0.8995023	0.232552739				
Grocery		-0.4602017	0.753856344				
Frozen		-0.8894969	-0.024462167				
Detergents_Paper		-0.4111027	-0.406639843				
Delicassen		0.1523283	-0.006113117				
Proportion of trace							
LD1	LD2						
0.8344	0.1656						
Confusion Matrix							
		truth					
		Lisbon	Oporto	Other Region			
predict	Lisbon	0	0	0			
	Oporto	0	0	2			
	Other Region	10	10	65			
Classification Accuracy		74.7126%					



Summary of Canonical Discriminant Functions

Eigenvalues

Function	Eigenvalue	% of Variance	Cumulative %	Canonical Correlation
1	.035[a]	73.4	73.4	.183
2	.013[a]	26.6	100.0	.111

a. First 2 canonical discriminant functions were used in the analysis.

Wilks' Lambda

Test of Function(s)	Wilks' Lambda	Chi-square	df	Sig.
1 through 2	.955	20.147	14	.126
2	.988	5.402	6	.493

Standardized Canonical Discriminant Function Coefficients

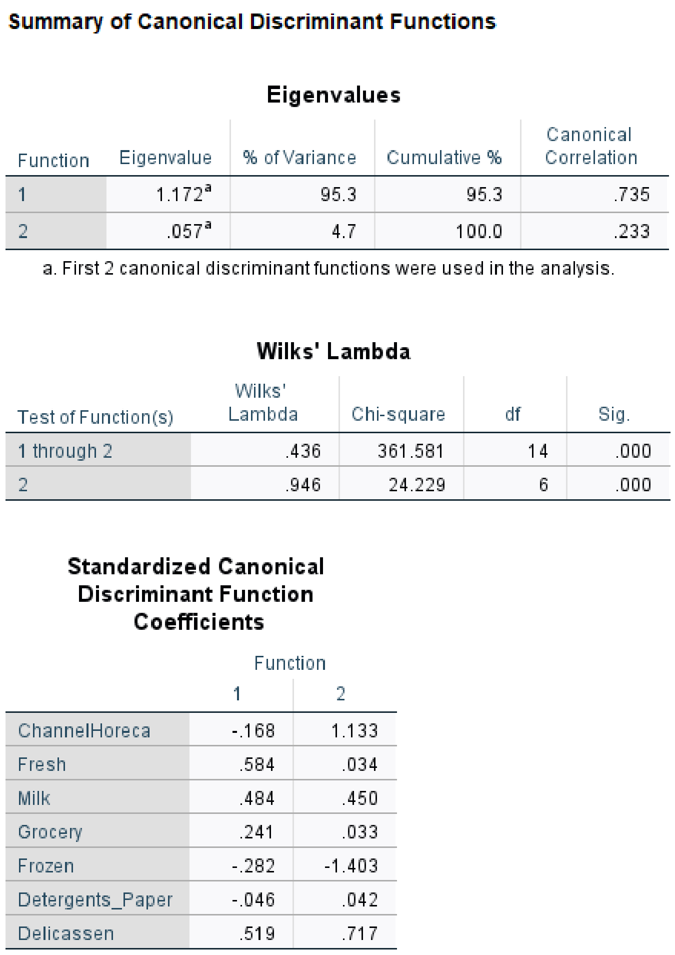
	Function	
	1	2
ChannelHoreca	.328	1.173
Fresh	.376	-.483
Milk	.841	.158
Grocery	-.305	.094
Frozen	-.873	.053
Detergents_Paper	-.480	.377
Delicassen	.284	-.238

Figure B.7: SPSS Significance Output for the three groups

B.2.4.2 Balanced Data

Table B.10: LDA model : Three Classes

Prior probabilities of groups							
Lisbon	Oporto	Other Region					
0.1898017	0.1048159	0.7053824					
Group means							
	ChannelRetail	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicassen
Lisbon	0.2427660	-0.05380345	-0.04845701	-0.0684995	-0.002396647	-0.07090886	-0.06473997
Oporto	0.4348894	-0.21994460	-0.03799474	0.1983732	0.034471140	0.25942821	-0.17456673
Other Region	0.4576271	0.5423729	1.14716986	2.76412498	1.9039890	1.334710842	1.38287658
Coefficients of linear discriminants							
		LD1	LD2				
ChannelRetail		0.10921496	-1.187434742				
Fresh		0.31953979	-0.005880814				
Milk		0.19234359	0.408461590				
Grocery		0.32721060	0.150904852				
Frozen		-0.32851322	-0.572730419				
Detergents_Paper		-0.07527566	-0.292854795				
Delicassen		0.24785495	0.051421068				
Proportion of trace							
LD1	LD2						
0.9511	0.0489						
Confusion Matrix							
		truth					
		Lisbon	Oporto	Other Region			
predict	Lisbon	24	15	3			
	Oporto	5	14	1			
	Other Region	0	0	25			
Classification Accuracy		72.41379%					



Summary of Canonical Discriminant Functions

Eigenvalues

Function	Eigenvalue	% of Variance	Cumulative %	Canonical Correlation
1	1.172[a]	95.3	95.3	.735
2	.057[a]	4.7	100.0	.233

a. First 2 canonical discriminant functions were used in the analysis.

Wilks' Lambda

Test of Function(s)	Wilks' Lambda	Chi-square	df	Sig.
1 through 2	.436	361.581	14	.000
2	.946	24.229	6	.000

Standardized Canonical Discriminant Function Coefficients

	Function	
	1	2
ChannelHoreca	-.168	1.133
Fresh	.584	.034
Milk	.484	.450
Grocery	.241	.033
Frozen	-.282	-1.403
Detergents_Paper	-.046	.042
Delicassen	.519	.717

Figure B.8: SPSS Significance Output for the three groups - Balanced Data

B.2.5 Other Discriminant Analysis Models - Three Classes

Table B.11: Other Discriminant Analysis Models Results

	QDA	MDA	HDA
Classification Accuracy	74.7126%	73.5632%	70.1150%

B.2.6 Multi-Class Classification Using Iris Dataset

B.2.6.1 LDA Assumptions Tests

Table B.12: Box M Test for Iris-Species

	DF	X-squared	p-value
Iris-Species	20	140.94	2.2e-16

Table B.13: Levene's Test for Homogeneity of Variance - Iris-Species

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
F-value	6.3527	0.5902	19.48	19.48
$Pr(> F)$	0.002259	0.5555	3.129e-08	3.129e-08

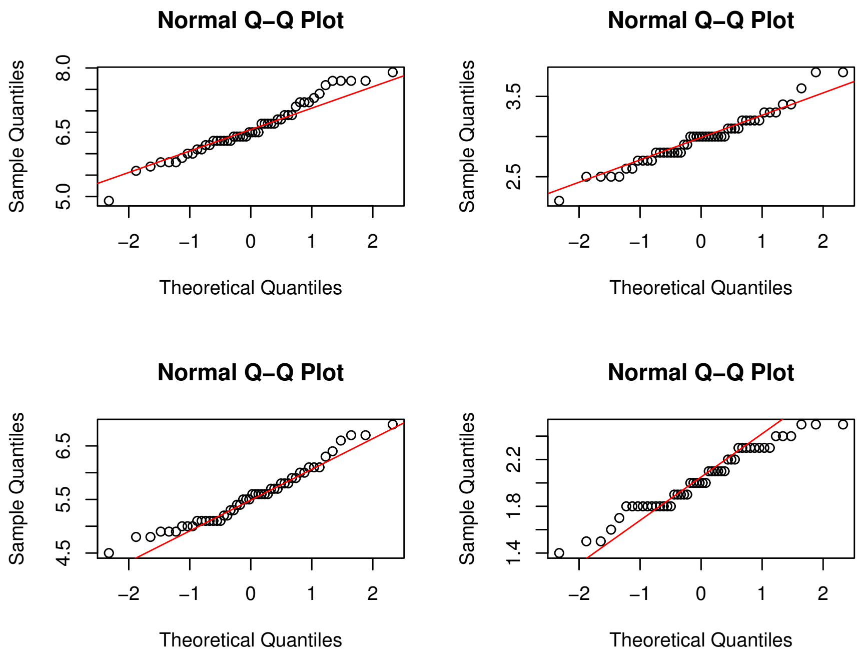


Figure B.9: QQ Plots Normal Check - Species-Virginica

Table B.14: Shapiro-Wilk normality test - Species

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
Setosa				
W	0.9777	0.97172	0.95498	0.79976
p-value	0.4595	0.2715	0.05481	8.659e-07
Versicolor				
W	0.97784	0.97413	0.966	0.94763
p-value	0.4647	0.338	0.1585	0.02728
Virginica				
W	0.97118	0.96739	0.96219	0.95977
p-value	0.2583	0.1809	0.1098	0.08695

B.2.6.2 Fitted Models - Iris Data

Table B.15: LDA Multi-Class Model - Iris Data

Prior probabilities of groups				
setosa	versicolor	virginica		
0.3333333	0.3333333	0.3333333		
Group means				
	x.Sepal.Length	x.Sepal.Width	x.Petal.Length	x.Petal.Width
setosa	-0.9989259	0.9122085	-1.2997229	-1.2401798
versicolor	0.1285026	-0.5927640	0.2765068	0.1761694
virginica	0.9189148	-0.1261768	1.0043441	1.0941735
Coefficients of linear discriminants				
		LD1	LD2	
x.Sepal.Length		0.6841420	0.01988153	
x.Sepal.Width		0.6708982	0.94636612	
x.Petal.Length		-3.9074192	-1.65427384	
x.Petal.Width		-2.1430429	2.16494837	
Proportion of trace				
LD1	LD2			
0.8344	0.1656			
Confusion Matrix				
		truth		
		setosa	versicolor	virginica
predict	setosa	50	0	0
	versicolor	0	48	1
	virginica	10	2	49
Classification Accuracy		98.00%		

Table B.16: SVM Multi-Class Model - Iris Data

SVM-Type	C-classification			
SVM-Kernel	linear			
cost	0.1			
gamma	1			
Number of Support Vectors		123		
Confusion Matrix				
		truth		
		setosa	versicolor	virginica
predict	setosa	49	0	0
	versicolor	0	49	3
	virginica	0	0	47
Classification	**Accuracy**	96.6667%		

Table B.17: Tuned SVM Multi-Class Model - Iris Data

SVM-Type	C-classification			
SVM-Kernel	linear			
cost	100			
gamma	1			
Number of Support Vectors		55		
Confusion Matrix				
		truth		
		setosa	versicolor	virginica
predict	setosa	50	0	0
	versicolor	0	50	0
	virginica	0	0	50
Classification	**Accuracy**	100%		

B.3 LDA and SVM Code Listing

B.3.1 Unbalanced Data

```
set.seed(10)
# Attach Packages
library(tidyverse) # data manipulation and visualization
library(kernlab) # SVM methodology
library(e1071) # SVM methodology
library(ISLR) # contains example data set "Khan"
library(RColorBrewer) # customized coloring of plots
library(caret)
library(dummies)
library(corrplot)
#########################################################################
#TWO CLASSES - PRE-PROCESSING
#########################################################################
data_orig <-read.csv(file="Wholesale_customers_data.csv", header=TRUE, sep=",")
head(data_orig)
print(xtable(head(data_orig), type = "latex"), file = "filenameHead0.tex")
data <-read.csv(file="Wholesale_customers_data_Dummy.csv", header=TRUE, sep=",")
training.samplesCustormer <- data$Channel %>%
  createDataPartition(p = 0.8, list = FALSE)
train.dataCustomer <- data[training.samplesCustormer, ]
test.dataCustomer <- data[-training.samplesCustormer, ]
train.data = train.dataCustomer[-1]
train.channel_label = as.factor(train.dataCustomer$Channel)
test.data = test.dataCustomer[-1]
test.channel_label = as.factor(test.dataCustomer$Channel)
preproc.param <- train.data %>%
  preProcess(method = c("center", "scale"))
# Transform the data using the estimated parameters
train.transformedCustomer <- preproc.param %>% predict(train.data)
test.transformedCustomer <- preproc.param %>% predict(test.data)
train.dataframe = data.frame(x=train.transformedCustomer,y=train.channel_label)
test.dataframe = data.frame(x=test.transformedCustomer,y=test.channel_label)
all_data_channel = rbind(train.dataframe,test.dataframe)
#names(all_data_channel)
pairs(all_data_channel[c("x.Region","x.Fresh","x.Milk","x.Grocery","x.Frozen","x.Detergents_
    ↪ Paper","x.Delicassen")], main="Pairwise␣Scatter␣Plots␣-␣Channel", pch=22,
     bg=c("red", "yellow")[unclass(all_data_channel$y)])
train.transformedCustomer.dummy_full = dummy.data.frame(train.transformedCustomer,drop =
    ↪ FALSE)
train.transformedCustomer.dummy = train.transformedCustomer.dummy_full[-3]
test.transformedCustomer.dummy_full = dummy.data.frame(test.transformedCustomer)
test.transformedCustomer.dummy = test.transformedCustomer.dummy_full[-3]
train.dataframe = data.frame(x=train.transformedCustomer.dummy,y=train.channel_label)
test.dataframe = data.frame(x=test.transformedCustomer.dummy,y=test.channel_label)
#########################################################################
#TWO CLASSES - SUPPORT VECTOR MACHINE
#########################################################################
```

```
tune.out <- tune(svm, y~., data = train.dataframe, kernel = "radial",
               ranges = list(cost = c(0.1,1,10,100,1000),
                            gamma = c(0.5,1,2,3,4)))
(bestmod <- tune.out$best.model)
ypred <- predict(bestmod, test.dataframe)
(misclass <- table(predict = ypred, truth = test.dataframe$y))
mean(ypred==test.dataframe$y)
svmfit <- svm(y~., data = train.dataframe, kernel = "linear", gamma = 1, cost = 1)
svmfit
ypred <- predict(svmfit, test.dataframe)
(misclass <- table(predict = ypred, truth = test.dataframe$y))
#print(xtable(misclass, type = "latex"), file = "filenameSVM1.tex")
mean(ypred==test.dataframe$y)
#########################################################################################
#MANOVA
#########################################################################################
customer.manova <- lm(as.matrix(train.dataframe[, 1:8])~y,
                   train.dataframe)
summary(customer.manova)
summary(manova(customer.manova), test = "Wilks")
library(candisc)
library(library(DescTools))
customer.can <- candisc(customer.manova)
customer.can
plot(customer.can)
DescTools::plot(customer.can)
#########################################################################################
#TWO CLASSES - DICRIMINANT ANALYSIS
########################################################################################
library(MASS)
# Fit the model
model <- lda(y~., data = train.dataframe)
model
predictions <- model %>% predict(test.dataframe)
(misclass <- table(predict = predictions$class, truth = test.dataframe$y))
print(xtable(misclass, type = "latex"), file = "filenameLda1.tex")
mean(predictions$class==test.dataframe$y)
summary(manova(y~., data = train.dataframe),test="Wilks")
library(MASS)
y<-as.factor(train.dataframe$y)
model <- hda(train.dataframe[-9],y)
predicted.classes <- model %>% predict(test.dataframe)
# Model accuracy
mean(predicted.classes == test.dataframe$y)
plot(model)
plot(model, dimen=1, type="both")
library(klaR)
all_data<-rbind(train.dataframe,test.dataframe)
partimat(y~.,data=train.dataframe,method="lda")
partimat(y~.,data=train.dataframe,method="lda",plot.matrix = TRUE, imageplot = FALSE)
```

```r
print(xtable(head(all_data), type = "latex"), file = "filenameHead2.tex")
M <-cor(all_data[-9])
library(xtable)
as.table(round(M,2))
print(xtable(as.table(round(M,2)), type = "latex"), file = "filename2.tex")
library(Hmisc)
mydata.rcorr = rcorr(as.matrix(all_data[-9]))
mydata.rcorr
predictions <- model %>% predict(test.dataframe)
names(predictions)
mean(predictions$class == test.dataframe$y)
# Predicted classes
head(predictions$class, 6)
# Predicted probabilities of class memebership.
head(predictions$posterior, 6)
# Linear discriminants
head(predictions$x, 3)
###################################################################
#THREE CLASSES - PRE-PROCESSING
###################################################################
train.region_label = as.factor(train.dataCustomer$Region)
test.region_label = as.factor(test.dataCustomer$Region)
train.data = train.dataCustomer[-2]
test.data = test.dataCustomer[-2]
train.data$Channel = train.dataCustomer$Channel
test.data$Channel = test.dataCustomer$Channel
preproc.param <- train.data %>%
  preProcess(method = c("center", "scale"))
# Transform the data using the estimated parameters
train.transformedCustomer <- preproc.param %>% predict(train.data)
test.transformedCustomer <- preproc.param %>% predict(test.data)
train.transformedCustomer.dummy.region_full = dummy.data.frame(train.transformedCustomer)
train.transformedCustomer.dummy.region = train.transformedCustomer.dummy.region_full[-1]
test.transformedCustomer.dummy.region_full = dummy.data.frame(test.transformedCustomer)
test.transformedCustomer.dummy.region = test.transformedCustomer.dummy.region_full[-1]
train.dataframe.region = data.frame(x=train.transformedCustomer,y=train.region_label)
test.dataframe.region = data.frame(x=test.transformedCustomer,y=test.region_label)
all_data_region = rbind(train.dataframe.region,test.dataframe.region)
pairs(all_data_region[c("x.Channel","x.Fresh","x.Milk","x.Grocery","x.Frozen","x.Detergents_
    ↪ Paper","x.Delicassen")], main="Pairwise␣Scatter␣Plots␣-␣Region", pch=22,
      bg=c("blue", "yellow","red")[unclass(all_data_region$y)])
train.region_dataframe = data.frame(x=train.transformedCustomer.dummy.region,y=train.region_
    ↪ label)
test.region_dataframe = data.frame(x=test.transformedCustomer.dummy.region,y=test.region_
    ↪ label)
##########################################################################
#THREE CLASSES - SUPPORT VECTOR MACHINE
##########################################################################
tune.out <- tune(svm, y~., data = train.region_dataframe, kernel = "radial",
               ranges = list(cost = c(0.1,1,10,100,1000),
                             gamma = c(0.5,1,2,3,4)))
(bestmod <- tune.out$best.model)
svmfit <- svm(y~., data = train.region_dataframe, kernel = "radial", gamma = 1, cost = 0.1)
ypred <- predict(svmfit, test.region_dataframe)
(misclass <- table(predict = ypred, truth = test.region_dataframe$y))
mean(ypred==test.region_dataframe$y)
plot(bestmod,test.region_dataframe)
##############################################################################
#THREE CLASSES - DISCRIMINANT ANALYSIS TWO CLASSES
##############################################################################
# Fit the model
model <- lda(y~., data = train.region_dataframe)
model
predictions <- model %>% predict(test.region_dataframe)
(misclass <- table(predict = predictions$class, truth = test.region_dataframe$y))
print(xtable(misclass, type = "latex"), file = "filenameLda2.tex")
mean(predictions$class==test.region_dataframe$y)
plot(model)
plot(model, dimen=1, type="both")
library(klaR)
all_data<-rbind(train.region_dataframe,test.region_dataframe)
print(xtable(head(all_data), type = "latex"), file = "filenameHead3.tex")
partimat(y~.,data=all_data,method="lda")
partimat(y~.,data=all_data,method="lda",plot.matrix = TRUE, imageplot = FALSE)
M <-cor(all_data[-9])
# Scatterplot for 3 Group Problem
pairs(all_data[c("x.Channel","x.Fresh","x.Milk","x.Grocery","x.Frozen","x.Detergents_Paper",
    ↪ "x.Delicassen")], main="Pairwise␣Scatter␣Plots", pch=22,
      bg=c("red", "yellow", "blue")[unclass(train.region_dataframe$y)])
train.region_dataframe$x.Delicassen
predictions <- model %>% predict(test.region_dataframe)
names(predictions)
predictions$class
# Predicted classes
head(predictions$class, 6)
# Predicted probabilities of class memebership.
head(predictions$posterior, 6)
# Linear discriminants
head(predictions$x, 3)
lda.data <- cbind(train.region_dataframe, predict(model$x)
ggplot(lda.data, aes(LD1, LD2)) +
  geom_point(aes(color = y))
# Fit the model
model <- qda(y~., data = train.region_dataframe)
model
# Make predictions
predictions <- model %>% predict(test.region_dataframe$y)
# Model accuracy
mean(predictions$class == test.region_dataframe$y)
library(mda)
```

```
model <- mda(y~., data = train.region_dataframe)
model
# Make predictions
predicted.classes <- model %>% predict(test.region_dataframe)
# Model accuracy
mean(predicted.classes == test.region_dataframe$y)
library(hda)
model <- hda(train.region_dataframe[-8],as.factor(train.region_dataframe$y),crule = TRUE)
prediction = predict(model,test.region_dataframe[-8],task="c")
mean(prediction$prediction == test.region_dataframe$y)
```

B.3.2 Balanced Data

```
#BalancedDataChannel.R
library(dummies)
library(scutr)
data(bullseye)
library(tidyverse) # data manipulation and visualization
library(kernlab) # SVM methodology
library(e1071) # SVM methodology
library(ISLR) # contains example data set "Khan"
library(RColorBrewer) # customized coloring of plots
library(caret)
library(corrplot)
library(MASS)
library(MLmetrics)
library(mltest)
ldaAccuracy = 0
svmConMatrix<-NULL
svmConMatrix<-NULL
ypred_class<-NULL
predictions_class<-NULL
test_balanced_data_region_class<-NULL
mltest_results<-NULL
mltest_results_svm <-NULL
while(ldaAccuracy<0.3){
rm(list=ls())
data <-read.csv(file="Wholesale_customers_data_Dummy.csv", header=TRUE, sep=",")
Region = as.factor(data$Region)
data_1 <- data[-2]
preproc.param <- data_1 %>%
  preProcess(method = c("center", "scale"))
data_1.transformedCustomer <- preproc.param %>% predict(data_1)
dummied_data <- dummy.data.frame(data_1.transformedCustomer,drop = FALSE)
data_final =cbind(Region,dummied_data)
balanced_customer_data_region_partioning <- data_final$Region %>%
  createDataPartition(p = 0.8, list = FALSE)
train_balanced_data_region <- data_final[balanced_customer_data_region_partioning,]
test_balanced_data_region <- data_final[-balanced_customer_data_region_partioning,]
test_balanced_data_region[, 1:1] <- sapply(test_balanced_data_region[, 1:1], as.factor)
test_balanced_data_region[, 1:1] <- sapply(test_balanced_data_region[, 1:1], as.factor)
balanced_customer_data_region <- SCUT(train_balanced_data_region, "Region", undersample =
    ↪ undersample_hclust,
                                          usamp_opts = list(dist_calc="manhattan"))
tune.out <- tune(svm, Region~., data = balanced_customer_data_region, kernel = "radial",
               ranges = list(cost = c(0.1,1,10,100,1000),
                          gamma = c(0.5,1,2,3,4)))
svmfit <- svm(Region~., data = balanced_customer_data_region, kernel = "radial", gamma = 1,
    ↪ cost = 0.1)
ypred <- predict(svmfit, test_balanced_data_region)
ypred_class <- factor(ypred,levels = c("Oporto","Lisbon","Other␣Region"))
test_balanced_data_region_class <- factor(test_balanced_data_region$Region,levels=c("Oporto"
    ↪ ,"Lisbon","Other␣Region"))
svmConMatrix <- confusionMatrix(data=ypred_class,reference=test_balanced_data_region_class,
    ↪ positive = "Oporto")
mltest_results_svm <-ml_test(ypred_class, test_balanced_data_region_class, output.as.table =
    ↪  FALSE)
model <- lda(Region~., data = balanced_customer_data_region)
model
predictions <- model %>% predict(test_balanced_data_region)
predictions_class <- factor(predictions$class,levels = c("Oporto","Lisbon","Other␣Region"))
test_balanced_data_region_class <- factor(test_balanced_data_region$Region,levels=c("Oporto"
    ↪ ,"Lisbon","Other␣Region"))
ldaConMatrix <- confusionMatrix(data=predictions_class,reference=test_balanced_data_region_
    ↪ class,positive = "Oporto")
mltest_results <-ml_test(predictions_class, test_balanced_data_region_class, output.as.table
    ↪  = FALSE)
ldaAccuracy <-as.numeric(mltest_results$accuracy)
out <- paste0("ldaAccuracy", ldaAccuracy, ".") # Some output
print(out)
}
svmConMatrix
svmConMatrix
#SVM
precision(data=ypred_class,reference = test_balanced_data_region_class,relevant = "Oporto")
recall(data=ypred_class,reference = test_balanced_data_region_class,relevant = "Oporto")
F1_Score(test_balanced_data_region_class,ypred_class,positive = "Oporto")
#LDA
precision(data=predictions_class,reference = test_balanced_data_region_class,relevant = "
    ↪ Oporto")
recall(data=predictions_class,reference = test_balanced_data_region_class,relevant = "Oporto
    ↪ ")
F1_Score(test_balanced_data_region_class,predictions_class,positive = "Oporto")
```

B.3.3 Balanced Data

```
library(dummies)
library(scutr)
data(bullseye)
library(tidyverse) # data manipulation and visualization
library(kernlab) # SVM methodology
library(e1071) # SVM methodology
library(ISLR) # contains example data set "Khan"
library(RColorBrewer) # customized coloring of plots
library(caret)
library(corrplot)
library(MASS)
rm(list=ls())
data <-read.csv(file="Wholesale_customers_data_Dummy.csv", header=TRUE, sep=",")
Channel = as.factor(data$Channel)
data_2 <- data[-1]

preproc.param <- data_2%>%
  preProcess(method = c("center", "scale"))
data_2.transformedCustomer <- preproc.param %>% predict(data_2)
dummied_data <- dummy.data.frame(data_2.transformedCustomer,drop = FALSE)
data_final_channel =cbind(Channel,dummied_data)
balanced_customer_data_channel_partioning <- data_final_channel$Channel %>%
  createDataPartition(p = 0.8, list = FALSE)
train_balanced_data_channel <- data_final_channel[balanced_customer_data_channel_partioning
    ↪ ,]
test_balanced_data_channel <- data_final_channel[-balanced_customer_data_channel_partioning
    ↪ ,]
train_balanced_data_channel[, 1:1] <- sapply(train_balanced_data_channel[, 1:1], as.factor)
test_balanced_data_channel[, 1:1] <- sapply(test_balanced_data_channel[, 1:1], as.factor)
training_balanced_customer_data_channel <- SCUT(train_balanced_data_channel, "Channel",
    ↪ undersample = undersample_hclust,
                                   usamp_opts = list(dist_calc="manhattan"))
training_balanced_customer_data_channel[,1:1] <- sapply(training_balanced_customer_data_
    ↪ channel[, 1:1], as.factor)
tune.out <- tune(svm, Channel~., data = training_balanced_customer_data_channel, kernel = "
    ↪ radial",
               ranges = list(cost = c(0.1,1,10,100,1000),
                          gamma = c(0.5,1,2,3,4)))
svmfit <- svm(Channel~., data = training_balanced_customer_data_channel, kernel = "radial",
    ↪ gamma = 1, cost = 0.1)
ypred <- predict(svmfit, test_balanced_data_channel)
ypred_class <- factor(ypred,levels = c("Retail","Horeca"))
test_balanced_data_channel_class <- factor(test_balanced_data_channel$Channel,levels=c("
    ↪ Retail","Horeca"))
confusionMatrix(data=ypred_class,reference=test_balanced_data_channel_class,positive = "
    ↪ Retail")
model <- lda(Channel~., data = training_balanced_customer_data_channel)
model
predictions <- model %>% predict(test_balanced_data_channel)
predictions_class <- factor(predictions$class,levels = c("Retail","Horeca"))
test_balanced_data_channel_class <- factor(test_balanced_data_channel$Channel,levels = c("
    ↪ Retail","Horeca"))
confusionMatrix(data=predictions_class,reference = test_balanced_data_channel_class,positive
    ↪  = "Retail")
```

```
table(train_balanced_data_channel$Channel)
table(training_balanced_customer_data_channel$Channel)
table(test_balanced_data_channel$Channel)
#SVM
precision(data=ypred_class,reference = test_balanced_data_channel_class,relevant = "Retail")
recall(data=ypred_class,reference = test_balanced_data_channel_class,relevant = "Retail")
F1_Score(test_balanced_data_channel_class,ypred_class,positive = "Retail")
#LDA
precision(data=predictions_class,reference = test_balanced_data_channel_class,relevant = "
    ↪ Retail")
recall(data=predictions_class,reference = test_balanced_data_channel_class,relevant = "
    ↪ Retail")
F1_Score(test_balanced_data_channel_class,predictions_class,positive = "Retail")
```

Appendix C

Clustering Case Study Appendices

C.1 Gaussian Mixture Clustering

C.1.1 Maximum Likelihood Estimation for a Gaussian Distribution

If $X_1, \ldots, X_n$ are independent observations from a Gaussian distribution with an unknown μ and known σ^2, the maximum likelihood estimate of μ is found by setting the derivative of the log-likelihood with respect to μ, $l(\mu)$, equal to zero and solving for μ.

$$\begin{aligned}
L(\mu) &= \prod_{j=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} exp - \frac{(x_j-\mu)^2}{2\sigma^2} \\
l(\mu) &= \sum_{j-1}^{n} [log(\frac{1}{\sqrt{2\pi\sigma^2}}) - \frac{(x_j-\mu)^2}{2\sigma^2}] \\
\frac{d}{d\mu} l(\mu) &= \sum_{j-1}^{n} \frac{x_j-\mu}{\sigma^2} \\
\sum_{j-1}^{n} \frac{x_j-\mu}{\sigma^2} &= 0 \\
\Rightarrow \mu_{MLE} &= \frac{1}{n} \sum_{j=1}^{n} x_j
\end{aligned}$$

C.1.2 Maximum Likelihood Estimation for a Gaussian Mixture

The following likelihood estimation is an extract from the notes of Bonakdarpour (2016).

If $X_1, \ldots, X_n$ comes from one of the K mixture components, there is a label $Z_j \in \{1, \ldots, K\}$ which shows which component X_j comes from. In most cases Z_j is not observable, in such cases it is called a latent variable. From the law of total probability we have,

$$P(X_j = x) = \sum_{k=1}^{K} P(X_j = x | Z_j = k) P(Z_j = k) = \sum_{k=1}^{K} P(X_j = x | Z_j = k) \pi_k, \tag{C.1}$$

where π_k are mixture proportions (weights) representing the probability that X_j belongs to the k^{th}

mixture component. The probability density function of the mixture model then becomes,

$$f_x(x) = \sum_{k=1}^{K} \pi_k f_{x|Z_k}(x|Z_k). \tag{C.2}$$

In the GMM the k^{th} component is $N(\mu_k, \sigma_k)$ with the proportion π_k. Assuming independent samples $X_1, \ldots, X_n$ from this mixture and weights $\pi = (\pi_1, \ldots, \pi_K)$, the likelihood function becomes, with $\theta = \{\mu_1, \ldots, \mu_K, \sigma_1, \ldots, \sigma_K, \pi_1, \ldots, \pi_K\}$, the likelihood function is given as,

$$L(\theta|X_i, \ldots, X_n) = \prod_{j=1}^{n} \sum_{k=1}^{K} \pi_k N(x_j; \mu_k, \sigma_k^2). \tag{C.3}$$

And log-likelihood,

$$l(\theta) = \sum_{i=1}^{n} log(\sum_{k=1}^{K} \pi_k N(x_j; \mu_k, \sigma_k^2)) \tag{C.4}$$

The summation inside the log-likelihood renders the likelihood estimate to have no closed form solution, unlike in appendix C.1.1 above.

C.1.3 The mechanics of Expectation Maximization Algorithm

The summation problem in equation (C.4) lands us to the use of the EM algorithm. Bonakdarpour (2016) simplifies equation (C.4) further by differentiating with respect to μ_k and setting the derivative equals to zero,

$$\sum_{j=1}^{n} \frac{1}{\sum_{k=1}^{K} \pi_k N(x_j; \mu_k, \pi_k)} \pi_k N(x_j; \mu_k, \pi_k) \frac{x_j - \mu_k}{\sigma_k^2} = 0 \tag{C.5}$$

As can be seen in equation (C.5), there is not direct solution to μ_k unlike in appendix C.1.1. If the latent variable Z_j was known, that is, if the prevailing mixture component was known, estimating MLEs would have been similar to the solution in appendix C.1.1.

Next, the posterior distribution of Z_j given the observations is examined,

$$P(Z_j = k|X_j) = \frac{P(X_j|Z_j = k)P(Z_j = k)}{P(X_j)} = \frac{\pi_k N(\mu_k, \sigma_k^2)}{\sum_{k=1}^{K} N(\mu_k, \sigma_k^2)} = \gamma_{Z_j}(k) \tag{C.6}$$

.

Substituting back in equation (C.4) simplifies to,

$$\gamma_{Z_j}(k) \frac{x_j - \mu_k}{\sigma_k^2} = 0 \tag{C.7}$$

The log-likelihood in equation (C.7) but $\gamma_{Z_j}(k)$ still depends on μ_k but in order to make the solution more tractable we imagine we have all the information required about $\gamma_{Z_j}(k)$. That enables the derivation of the value of μ_k,

$$\hat{\mu_k} = \frac{\sum_{j=1}^{n} \gamma_{Z_j}(k) x_j}{\sum_{j=1}^{n} \gamma_{Z_j}(k)} = \frac{1}{N_k} \sum_{j=1}^{n} \gamma_{Z_j}(k) x_j \tag{C.8}$$

$N_k = \sum_{j=1}^{n} \gamma_{Z_j}(k)$ and N_k is seen as the number points allocated to the k^{th} mixture component, and $\hat{\mu_k}$ an average of the data with weights $\gamma_{Z_j}(k)$. The estimates of $\hat{\sigma^2}$ and $\hat{\pi_k}$ are derived in a similar fashion,

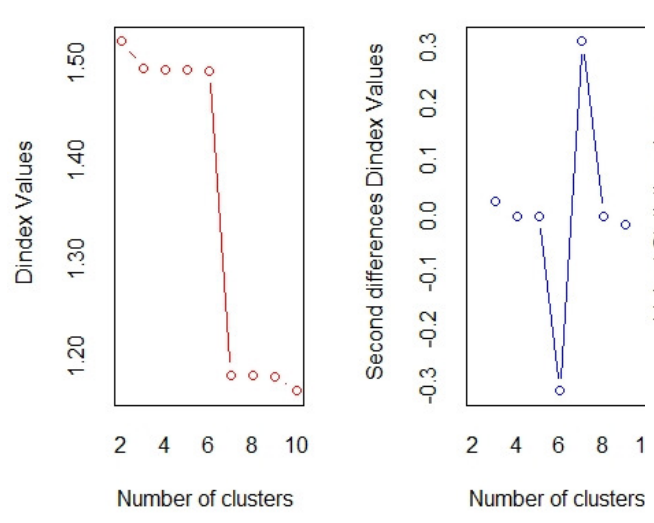
$$\hat{\sigma^2} = \frac{1}{N_k}\sum_{j=1}^{n} \gamma_{Z_j}(k)(x_j - \mu_k)^2 \tag{C.9}$$

$$\hat{\pi_k} = \frac{N_k}{n} \tag{C.10}$$

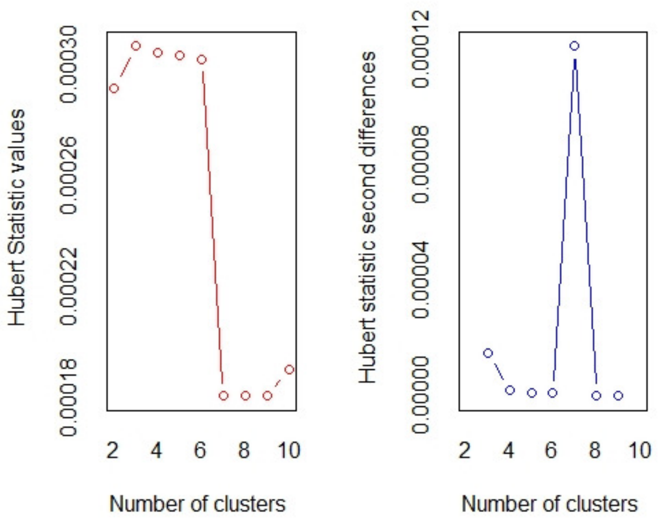
because the estimate equations about still depend on the unknown $\gamma_{Z_j}(k)$, they still not closed-form.

C.1.4 Number of Cluster using NBClust R package

"The Hubert index is a graphical method of determining the number of clusters. In the plot of Hubert index, we seek a significant knee that corresponds to a significant increase of the value of the measure i.e the significant peak in Hubert index second differences plot.
The D index is a graphical method of determining the number of clusters. In the plot of D index, we seek a significant knee (the significant peak in Dindex second differences plot) that corresponds to a significant increase of the value of the measure" (Charrad et al., 2014).



(a) NbClust Dindex Plot



(b) NbClust Hubert Plot

Figure C.1: NbClust Plots for Determining No. of Clusters

Table C.1: NbClust Package Output for optimal No. of Clusters

Number of Cluster using NBClust R package based on majority of indices								
No. of indices	0	1	2	3	4	**7**	9	10
Clusters	2	1	8	2	1	**10**	1	1
According to the majority rule, the best number of clusters is 7								

C.1.5 Summary of Results GMM-EM

Table C.2: Fitted Mixutre Model

Mclust VEV *ellipsoidal, equalshape* **model with 7 components**							
log-likelihood	n	df	BIC	ICL			
-9412.116	3863	57	-19295.01	-20000.83			
Clustering table							
Cluster	1	2	3	4	5	6	7
Cluster size	777	864	193	331	693	750	255
Mixing probabilities							
Cluster	1	2	3	4	5	6	7
Probability	0.20005368	0.22249670	0.06302376	0.08588798	0.17129908	0.19352086	0.06371793
Means							
Cluster	1	2	3	4	5	6	7
recency.z	-0.4992681	-0.5710891	-0.1197149	1.227124	0.1736734	0.07985965	1.3165984
frequency.z	0.7021322	1.0849905	0.3350786	-1.041919	-0.2654913	-1.04191935	-1.0419194
monetary.z	0.7782564	0.5637699	-0.1480648	-1.177456	-0.1648601	-0.58499772	-0.4585748

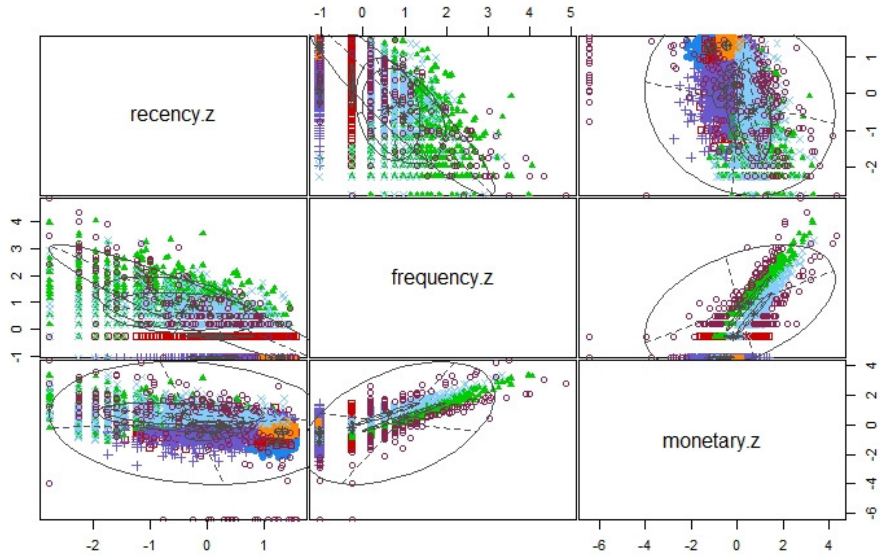


Figure C.2: GMM Cluster Classification Plot Using Mclust

C.1.6 Model Validation - Nonparametric Bootstrap

Table C.3: Bootstrap LRT

Bootstrap sequential LRT for the number of mixture components		
Model	VEV	
Replications	999	
	LRTS	bootstrap p-value
1 vs 2	2154.2930	0.001
2 vs 3	2201.3051	0.001
3 vs 4	917.0034	0.001
4 vs 5	1372.7310	0.001
5 vs 6	874.7586	0.001
6 vs 7	811.1973	0.001
7 vs 8	-644.1151	0.865

Table C.4: Bootstrap Confidence Intervals

Resampling confidence intervals							
Model	VEV						
Num. of mixture components	7						
Replications	999						
Type	nonparametric bootstrap						
Confidence level	0.95						
Mixing probabilities							
	{1}	{2}	{3}	{4}	{5}	{6}	{7}
2.5%	0.1754440	0.1612893	0.04856662	0.07257213	0.1592925	0.1747912	0.05151983
97.5%	0.2330226	0.2490901	0.12586741	0.10308833	0.1844758	0.2059157	0.08184972
Means							
{1}							
	recency.z	frequency.z	monetary.z				
2.5%	-0.6436942	0.6026737	0.6607849				
97.5%	-0.3700357	0.8796493	0.9047186				
{2}							
	recency.z	frequency.z	monetary.z				
2.5%	-0.7278719	0.8676812	0.3638801				
97.5%	-0.4568941	1.3679799	0.7382950				
{3}							
	recency.z	frequency.z	monetary.z				
2.5%	-0.299747390	0.2196332	-0.4391577				
97.5%	-0.004762327	0.8826979	0.2497110				
{4}							
	recency.z	frequency.z	monetary.z				
2.5%	1.150471	-1.041919	-1.248109				
97.5%	1.276839	-1.041919	-1.095493				
{5}							
	recency.z	frequency.z	monetary.z				
2.5%	0.08049883	-0.2654913	-0.2137088				
97.5%	0.26541084	-0.2654913	-0.1208137				
{6}							
	recency.z	frequency.z	monetary.z				
2.5%	0.01998159	-1.041919	-0.6331720				
97.5%	0.14188578	-1.041919	-0.5367345				
{7}							
	recency.z	frequency.z	monetary.z				
2.5%	1.245015	-1.041919	-0.5531835				
97.5%	1.358370	-1.041919	-0.3924755				

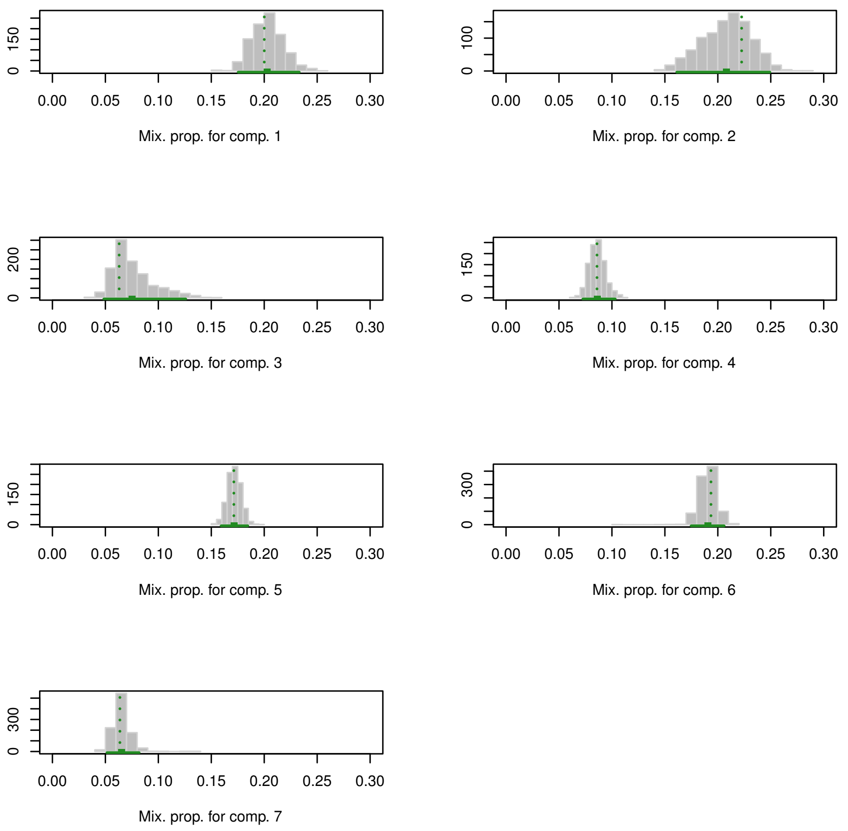


Figure C.3: Mixing Proportions Bootstrap CI's

C.2 K-means Clustering

Table C.5: Fitted K-means Model

K-means Clustering							
clusters	recency.z	requency.z	monetary.z				
1	-0.1512098	-0.86970244	-0.6993589				
2	0.5514715	-0.92545515	-6.1748337				
3	-0.2364563	1.00832024	0.8993242				
4	0.5203006	0.01961307	0.1540284				
5	-1.1861888	0.24841920	0.1665349				
6	1.1633664	-0.93706785	-0.8054396				
7	-1.5761249	1.95376531	1.5830640				
Clustering table							
Cluster	1	2	3	4	5	6	7
Cluster size	700	20	605	816	538	846	338
Within cluster sum of squares by cluster							
clusters	1	2	3	4	5	6	7
withinss	386.7962	33.2139	332.5557	426.6187	394.8918	334.3922	406.3537
between_SS/total_SS			80.0%				

Table C.6: K-means Jaccard Bootstrap Results

Cluster stability assessment							
Number of resampling runs		100					
Number of clusters found in data			7				
Cluster	1	2	3	4	5	6	7
Clusterwise Jaccard bootstrap mean	0.9865171	0.9692296	0.9308999	0.8658098	0.9164822	0.8750385	0.9305640
dissolved clusters	0	0	0	2	0	0	0
recovered clusters	100	100	100	84	98	88	98

C.3 Fuzzy C-means Clustering

C.3.1 Fuzzy C-means Model

Table C.7: Fuzzy C-mean Model

Fuzzy c-means clustering with 7 cluster centers							
Cluster centers							
cluster	1	2	3	4	5	6	7
recency.z	-0.02727329	-1.47890719	-0.51277986	1.19804973	0.04048628	-1.19009914	0.68580826
frequency.z	0.77262795	2.00253122	-0.08052568	-0.99523917	-0.96238606	0.86428551	-0.12651525
monetary.z	0.72917134	1.58973735	-0.10540659	-0.85675029	-0.73020454	0.69296295	-0.01245123
Clustering table							
Cluster	1	2	3	4	5	6	7
Cluster size	565	300	550	756	613	434	645
Membership probabilities (top 6)							
	1	2	3	4	5	6	7
1	0.11522487	0.079530040	0.14582433	0.21012379	0.18936857	0.10647193	0.15345646
2	0.06753919	0.600421024	0.04669823	0.01484405	0.02171453	0.22156852	0.02721445
3	0.14009794	0.354408794	0.10524371	0.06290867	0.07351817	0.17457166	0.08925105
4	0.10727686	0.208188889	0.08315885	0.02138948	0.03406345	0.50424448	0.04167799
5	0.09990678	0.079766381	0.16003650	0.02629630	0.05042431	0.53330621	0.05026352
6	0.02435295	0.008986579	0.04145553	0.70515465	0.12898202	0.01665349	0.07441478
Av silhouette index		0.2980478					
Dunn's Index		1.147474					
Within SS		2666.221					

C.3.2 Fuzzy C-means Cluster Structure

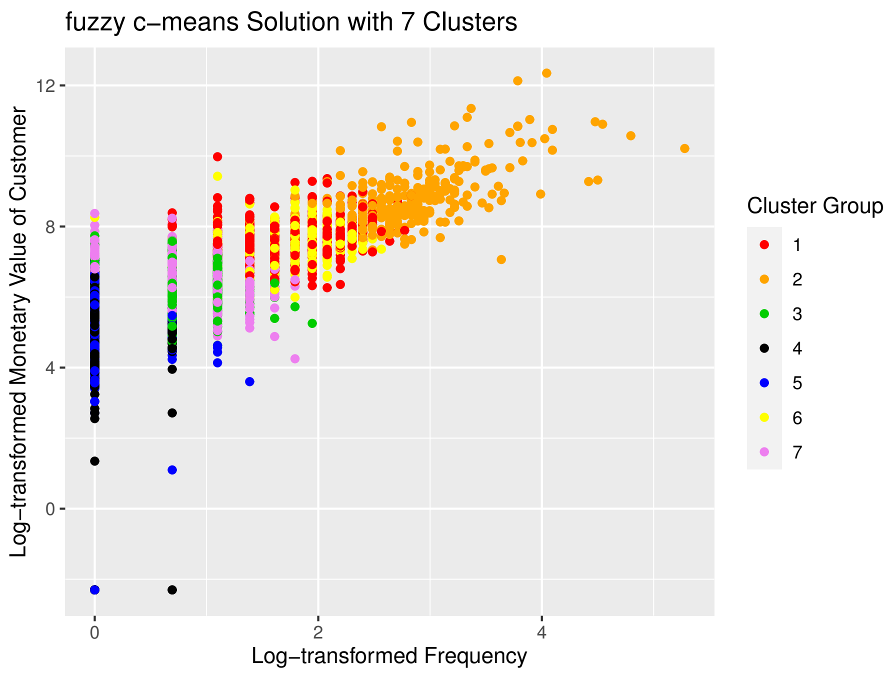


Figure C.4: Fuzzy C-means Cluster Structure

C.4 GMM and K-means Code Listing

```
library(XLConnect)
library(mclust)
library(factoextra)
library(kmeansstep)
library(fpc)
data <-read.csv(file="Online␣Retail.csv", header=TRUE, sep=",")
length(unique(data$CustomerID))
sum(is.na(data$CustomerID))
data <- subset(data, !is.na(data$CustomerID))
range(as.Date(data$InvoiceDate,format = "%m/%d/%Y"))
data <- subset(data, as.Date(data$InvoiceDate,format = "%m/%d/%Y") >= as.Date("12/09/2010",
    ↪ format = "%m/%d/%Y"))
range(as.Date(data$InvoiceDate,format = "%m/%d/%Y"))
table(data$Country)
data <- subset(data, Country == "United␣Kingdom")
length(unique(data$InvoiceNo))
length(unique(data$CustomerID))
# Identify returns
data$item.return <- grepl("C", data$InvoiceNo, fixed=TRUE)
data$purchase.invoice <- ifelse(data$item.return=="TRUE", 0, 1)
##################################
# Create customer-level dataset #
##################################
customers <- as.data.frame(unique(data$CustomerID))
names(customers) <- "CustomerID"
###########
# Recency #
###########
data$recency <- as.Date("12/10/2011",format="%m/%d/%Y") - as.Date(data$InvoiceDate,format="%
    ↪ m/%d/%Y")
# remove returns so only consider the data of most recent *purchase*
temp <- subset(data, purchase.invoice == 1)
# Obtain # of days since most recent purchase
recency <- aggregate(recency ~ CustomerID, data=temp, FUN=min, na.rm=TRUE)
remove(temp)
# Add recency to customer data
customers <- merge(customers, recency, by="CustomerID", all=TRUE, sort=TRUE)
remove(recency)
customers$recency <- as.numeric(customers$recency)
#############
# Frequency #
#############
customer.invoices <- subset(data, select = c("CustomerID","InvoiceNo", "purchase.invoice"))
customer.invoices <- customer.invoices[!duplicated(customer.invoices), ]
customer.invoices <- customer.invoices[order(customer.invoices$CustomerID),]
row.names(customer.invoices) <- NULL
# Number of invoices/year (purchases only)
```

```
annual.invoices <- aggregate(purchase.invoice ~ CustomerID, data=customer.invoices, FUN=sum,
    ↪ na.rm=TRUE)
names(annual.invoices)[names(annual.invoices)=="purchase.invoice"] <- "frequency"
# Add # of invoices to customers data
customers <- merge(customers, annual.invoices, by="CustomerID", all=TRUE, sort=TRUE)
remove(customer.invoices, annual.invoices)
range(customers$frequency)
table(customers$frequency)
# Remove customers who have not made any purchases in the past year
customers <- subset(customers, frequency > 0)
################################
# Monetary Value of Customers #
################################
# Total spent on each item on an invoice
data$Amount <- data$Quantity * data$UnitPrice
# Aggregated total sales to customer
annual.sales <- aggregate(Amount ~ CustomerID, data=data, FUN=sum, na.rm=TRUE)
names(annual.sales)[names(annual.sales)=="Amount"] <- "monetary"
# Add monetary value to customers dataset
customers <- merge(customers, annual.sales, by="CustomerID", all.x=TRUE, sort=TRUE)
remove(annual.sales)
# Identify customers with negative monetary value numbers, as they were presumably returning
    ↪ purchases from the preceding year
hist(customers$monetary)
customers$monetary <- ifelse(customers$monetary < 0, 0, customers$monetary) # reset negative
    ↪ numbers to zero
hist(customers$monetary)
customers <- customers[order(-customers$monetary),]
# Apply Pareto Principle (80/20 Rule)
pareto.cutoff <- 0.8 * sum(customers$monetary)
customers$pareto <- ifelse(cumsum(customers$monetary) <= pareto.cutoff, "Top␣20%", "Bottom␣8
    ↪ 0%")
customers$pareto <- factor(customers$pareto, levels=c("Top␣20%", "Bottom␣80%"), ordered=TRUE
    ↪ )
levels(customers$pareto)
round(prop.table(table(customers$pareto)), 2)
remove(pareto.cutoff)
customers <- customers[order(customers$CustomerID),]
# Log-transform positively-skewed variables
customers$recency.log <- log(customers$recency)
customers$frequency.log <- log(customers$frequency)
customers$monetary.log <- customers$monetary + 0.1 # can't take log(0), so add a small value
    ↪ to remove zeros
customers$monetary.log <- log(customers$monetary.log)
# Z-scores
customers$recency.z <- scale(customers$recency.log, center=TRUE, scale=TRUE)
customers$frequency.z <- scale(customers$frequency.log, center=TRUE, scale=TRUE)
customers$monetary.z <- scale(customers$monetary.log, center=TRUE, scale=TRUE)
library(ggplot2)
library(scales)
```

```
# Original scale
scatter.1 <- ggplot(customers, aes(x = frequency, y = monetary))
scatter.1 <- scatter.1 + geom_point(aes(colour = recency, shape = pareto))
scatter.1 <- scatter.1 + scale_shape_manual(name = "80/20␣Designation", values=c(17, 16))
scatter.1 <- scatter.1 + scale_colour_gradient(name="Recency\n(Days␣since␣Last␣Purchase))")
scatter.1 <- scatter.1 + scale_y_continuous(label=dollar)
scatter.1 <- scatter.1 + xlab("Frequency␣(Number␣of␣Purchases)")
scatter.1 <- scatter.1 + ylab("Monetary␣Value␣of␣Customer␣(Annual␣Sales)")
scatter.1
# Log-transformed
scatter.2 <- ggplot(customers, aes(x = frequency.log, y = monetary.log))
scatter.2 <- scatter.2 + geom_point(aes(colour = recency.log, shape = pareto))
scatter.2 <- scatter.2 + scale_shape_manual(name = "80/20␣Designation", values=c(17, 16))
scatter.2 <- scatter.2 + scale_colour_gradient(name="Log-transformed␣Recency")
scatter.2 <- scatter.2 + xlab("Log-transformed␣Frequency")
scatter.2 <- scatter.2 + ylab("Log-transformed␣Monetary␣Value␣of␣Customer")
scatter.2
# How many customers are represented by the two data points in the lower left-hand corner of
    ↪ the plot? 18
delete <- subset(customers, monetary.log < 0)
no.value.custs <- unique(delete$CustomerID)
delete2 <- subset(data, CustomerID %in% no.value.custs)
delete2 <- delete2[order(delete2$CustomerID, delete2$InvoiceDate),]
remove(delete, delete2, no.value.custs)
# Scaled variables
scatter.3 <- ggplot(customers, aes(x = frequency.z, y = monetary.z))
scatter.3 <- scatter.3 + geom_point(aes(colour = recency.z, shape = pareto))
scatter.3 <- scatter.3 + scale_shape_manual(name = "80/20␣Designation", values=c(17, 16))
scatter.3 <- scatter.3 + scale_colour_gradient(name="Z-scored␣Recency")
scatter.3 <- scatter.3 + xlab("Z-scored␣Frequency")
scatter.3 <- scatter.3 + ylab("Z-scored␣Monetary␣Value␣of␣Customer")
scatter.3
remove(scatter.1, scatter.2, scatter.3)
preprocessed <- customers[,9:11]
j <- 10 # specify the maximum number of clusters you want to try out
models <- data.frame(k=integer(),
                   tot.withinss=numeric(),
                   betweenss=numeric(),
                   totss=numeric(),
                   rsquared=numeric())
k <-7
  print(k)
    # Run kmeans
  # nstart = number of initial configurations; the best one is used
  # iter will return the iteration used for the final model
  output <- kmeans(preprocessed, centers = k, nstart = 20)
    # Add cluster membership to customers dataset
  var.name <- paste("cluster", k, sep="_")
  customers[,(var.name)] <- output$cluster
  customers[,(var.name)] <- factor(customers[,(var.name)], levels = c(1:k))
```

```
    # Graph clusters
  cluster_graph <- ggplot(customers, aes(x = frequency.log, y = monetary.log))
  cluster_graph <- cluster_graph + geom_point(aes(colour = customers[,(var.name)]))
  colors <- c('red','orange','green3','black','blue','yellow','violet')
  cluster_graph <- cluster_graph + scale_colour_manual(name = "Cluster␣Group", values=colors
      ↪ )
  cluster_graph <- cluster_graph + xlab("Log-transformed␣Frequency")
  cluster_graph <- cluster_graph + ylab("Log-transformed␣Monetary␣Value␣of␣Customer")
  title <- paste("k-means␣Solution␣with", k, sep="␣")
  title <- paste(title, "Clusters", sep="␣")
  cluster_graph <- cluster_graph + ggtitle(title)
  print(cluster_graph)
    # Cluster centers in original metrics
  library(plyr)
  print(title)
  cluster_centers <- ddply(customers, .(customers[,(var.name)]), summarize,
                       monetary=round(median(monetary),2),# use median b/c this is the raw
                           ↪ , heavily-skewed data
                       frequency=round(median(frequency),1),
                       recency=round(median(recency), 0))
  names(cluster_centers)[names(cluster_centers)=="customers[,␣(var.name)]"] <- "Cluster"
  print(cluster_centers)
  cat("\n")
  cat("\n")
    # Collect model information
  models[k,("k")] <- k
  models[k,("tot.withinss")] <- output$tot.withinss # the sum of all within sum of squares
  models[k,("betweenss")] <- output$betweenss
  models[k,("totss")] <- output$totss # betweenss + tot.withinss
  models[k,("rsquared")] <- round(output$betweenss/output$totss, 3) # percentage of variance
      ↪  explained by cluster membership
  assign("models", models, envir = .GlobalEnv)
    remove(cluster_graph, cluster_centers, title, colors)

remove(k)
library(ggplot2)
library(scales)
# Graph variance explained by number of clusters
r2_graph <- ggplot(models, aes(x = k, y = rsquared))
r2_graph <- r2_graph + geom_point() + geom_line()
r2_graph <- r2_graph + scale_y_continuous(labels = scales::percent)
r2_graph <- r2_graph + scale_x_continuous(breaks = 1:j)
r2_graph <- r2_graph + xlab("k␣(Number␣of␣Clusters)")
r2_graph <- r2_graph + ylab("Variance␣Explained")
r2_graph
# Graph within sums of squares by number of clusters
# Look for a "bend" in the graph, as with a scree plot
ss_graph <- ggplot(models, aes(x = k, y = tot.withinss))
ss_graph <- ss_graph + geom_point() + geom_line()
ss_graph <- ss_graph + scale_x_continuous(breaks = 1:j)
ss_graph <- ss_graph + scale_y_continuous(labels = scales::comma)
ss_graph <- ss_graph + xlab("k␣(Number␣of␣Clusters)")
ss_graph <- ss_graph + ylab("Total␣Within␣SS")
ss_graph
remove(j, r2_graph, ss_graph)
##########################################################
# Using NbClust metrics to determine number of clusters #
##########################################################
library(NbClust)
set.seed(1)
nc <- NbClust(preprocessed, min.nc=2, max.nc=10, method="centroid")
table(nc$Best.n[1,])
nc$All.index # estimates for each number of clusters on 26 different metrics of model fit
barplot(table(nc$Best.n[1,]),
       xlab="Number␣of␣Clusters", ylab="Number␣of␣Criteria",
       main="Number␣of␣Clusters␣Chosen␣by␣Criteria")
head(preprocessed)
mclustBIC2 = mclustBIC(preprocessed)
fit <- Mclust(preprocessed,G=7)
#GMM Bootstrapping
boot1 <- MclustBootstrap(fit, nboot = 999, type = "bs")
summary(boot1, what = "ci")
 LRT = mclustBootstrapLRT(preprocessed, modelName = "VEV")
  par(mfrow = c(2, 4))
 plot(LRT , G = 1)
 plot(LRT , G = 2)
 plot(LRT , G = 3)
 plot(LRT , G = 4)
 plot(LRT , G = 5)
 plot(LRT , G = 6)
 plot(LRT , G = 7)
 plot(LRT , G = 8)
 plot(LRT , G = 9)
 #Kmeans bootstrap
 kmeansBoot <- clusterboot(preprocessed,clustermethod = kmeansCBI,runs = 100,iter.max=100,
     ↪ krange = 7,seed=9898)
 k = fit$G
#plot(fit) # plot resul
summary(fit) # display the best model
var.name <- paste("cluster", k, sep="_")
customers[,(var.name)] <- fit$classification
customers[,(var.name)] <- factor(customers[,(var.name)], levels = c(1:k))
# Graph clusters
cluster_graph <- ggplot(customers, aes(x = frequency.log, y = monetary.log))
cluster_graph <- cluster_graph + geom_point(aes(colour = customers[,(var.name)]))
colors <- c('red','orange','green3','black','blue','yellow','violet')
cluster_graph <- cluster_graph + scale_colour_manual(name = "Cluster␣Group", values=colors)
cluster_graph <- cluster_graph + xlab("Log-transformed␣Frequency")
cluster_graph <- cluster_graph + ylab("Log-transformed␣Monetary␣Value␣of␣Customer")
title <- paste("GMM-EM␣Solution␣with", k, sep="␣")
```

```r
title <- paste(title, "Clusters", sep="␣")
cluster_graph <- cluster_graph + ggtitle(title)
print(cluster_graph)
fit$uncertainty
# Cluster centers in original metrics
library(plyr)
print(title)
cluster_centers <- ddply(customers, .(customers[,(var.name)]), summarize,
                       monetary=round(median(monetary),2),# use median b/c this is the raw,
                           ↪ heavily-skewed data
                       frequency=round(median(frequency),1),
                       recency=round(median(recency), 0))
names(cluster_centers)[names(cluster_centers)=="customers[,␣(var.name)]"] <- "Cluster"
#print(cluster_centers)
cat("\n")
cat("\n")
# BIC values used for choosing the number of clusters
fviz_mclust(fit, "BIC", palette = "jco")
# Classification: plot showing the clustering
fviz_mclust(fit, "classification", geom = "point",
           pointsize = 1.5, palette = "jco")
# Classification uncertainty
fviz_mclust(fit, "uncertainty", palette = "jco")
library(fpc)
km_stats <- cluster.stats(dist(preprocessed), output$cluster)
md_stats <- cluster.stats(dist(preprocessed), fit$classification)
c(km_stats,md_stats)
remove(preprocessed)
res.nbclust <- NbClust(preprocessed,
                     min.nc = 2, max.nc = 7,
                     method = "kmeans")
factoextra::fviz_nbclust(res.nbclust) + theme_minimal() + ggtitle("NbClust's␣optimal␣number␣
    ↪ of␣clusters")
library(ClusterR)
opt_gmm = Optimal_Clusters_GMM(preprocessed, 10, criterion = "BIC", plot_data = FALSE)

opt_gmm = Optimal_Clusters_GMM(preprocessed, max_clusters = 10, criterion = "BIC",

                            dist_mode = "maha_dist", seed_mode = "random_subset",

                            km_iter = 10, em_iter = 10, var_floor = 1e-10,

                            plot_data = T)

gmm = GMM(preprocessed, 2, dist_mode = "maha_dist", seed_mode = "random_subset", km_iter = 1
    ↪ 0,

         em_iter = 10, verbose = T)

opt = Optimal_Clusters_KMeans(preprocessed, max_clusters = 10, plot_clusters = T,
                          verbose = F, criterion = 'silhouette', fK_threshold = 0.85)
library(e1071)
cm <- cmeans(df, 7)

output <- cmeans(preprocessed, centers = k)
# Add cluster membership to customers dataset
var.name <- paste("cluster", k, sep="_")
customers[,(var.name)] <- output$cluster
customers[,(var.name)] <- factor(customers[,(var.name)], levels = c(1:k))
# Graph clusters
cluster_graph <- ggplot(customers, aes(x = frequency.log, y = monetary.log))
cluster_graph <- cluster_graph + geom_point(aes(colour = customers[,(var.name)]))
colors <- c('red','orange','green3','black','blue','yellow','violet')
cluster_graph <- cluster_graph + scale_colour_manual(name = "Cluster␣Group", values=colors)
cluster_graph <- cluster_graph + xlab("Log-transformed␣Frequency")
cluster_graph <- cluster_graph + ylab("Log-transformed␣Monetary␣Value␣of␣Customer")
title <- paste("fuzzy␣c-means␣Solution␣with", k, sep="␣")
title <- paste(title, "Clusters", sep="␣")
cluster_graph <- cluster_graph + ggtitle(title)
print(cluster_graph)
# Cluster centers in original metrics
library(plyr)
print(title)
cluster_centers <- ddply(customers, .(customers[,(var.name)]), summarize,
                      monetary=round(median(monetary),2),# use median b/c this is the raw,
                          ↪ heavily-skewed data
                      frequency=round(median(frequency),1),
                      recency=round(median(recency), 0))
names(cluster_centers)[names(cluster_centers)=="customers[,␣(var.name)]"] <- "Cluster"
print(cluster_centers)
cat("\n")
cat("\n")
library(fpc)
cm_stats <- cluster.stats(dist(preprocessed), output$cluster)
```