STATISTICAL METHODS FOR THE DETECTION OF NON-TECHNICAL LOSSES: A CASE STUDY FOR THE NELSON MANDELA BAY MUNICIPALITY

S. Pazi

2017

STATISTICAL METHODS FOR THE DETECTION OF NON-TECHNICAL LOSSES: A CASE STUDY FOR THE NELSON MANDELA BAY MUNICIPALITY

By

Sisa Pazi

Submitted in fulfilment of the requirements for the degree of Magister Scientiae (Mathematical Statistics) to be awarded at the Nelson Mandela Metropolitan University

April 2017

Supervisor: Prof GD Sharp

Co-Supervisor: Ms CM Clohessy

DEPARTMENT OF ACADEMIC ADMINISTRATION
EXAMINATION SECTION
SUMMERSTARND NORTH CAMPUS
PO Box 77000
Nelson Mandela Metropolitan University
Port Elizabeth
6031

Nelson Mandela Metropolitan University
for tomorrow

Enquiries: Postgraduate Examination Officer

DECLARATION BY CANDIDATE

NAME: SISA PAZI

STUDENT NUMBER: 209067822

QUALIFICATION: MSC (MATHEMATICAL STATISTICS)

TITLE OF PROJECT: STATISTICAL METHODS FOR THE DETECTION OF NON-TECHNICAL LOSSES: A CASE STUDY FOR THE NELSON MANDELA BAY MUNICIPALITY

DECLARATION:

In accordance with Rule G5.6.3, I hereby declare that the above-mentioned dissertation is my own work and that it has not previously been submitted for assessment to another University or for another qualification.

SIGNATURE: _____

DATE: _____

ACKNOWLEDGEMENTS

First and foremost, I would like to thank God for His love, faithfulness, grace and guidance in my life. A special thank you to my supervisors, Prof Gary Sharp and Ms Chantelle Clohessy for sharing their knowledge and time with me. Their encouragement and support played an important role in my study. Many thanks to my sisters, Unathi Faku and Noluthando Pazi for their love, support and patience. I would like to express my gratitude to the colleagues from the Department of Statistics, thank you for all your support.

Thank you to the following institutions and organisations:

- Amat Security and Cleaning Services (Pty) Ltd
- Nelson Mandela Bay Municipality
- South African Statistical Association
- National Research Foundation
- Nelson Mandela Metropolitan University RCD Department.

Lastly, I would like to thank my grandmother for supporting my studies, even though she never had a formal education.

ABSTRACT

Electricity is one of the most stolen commodities in the world. Electricity theft can be defined as the criminal act of stealing electrical power. Several types of electricity theft exist, including illegal connections and bypassing and tampering with energy meters. The negative financial impacts, due to lost revenue, of electricity theft are far reaching and affect both developing and developed countries. . Here in South Africa, Eskom loses over R2 Billion annually due to electricity theft. Data mining and nonparametric statistical methods have been used to detect fraudulent usage of electricity by assessing abnormalities and abrupt changes in kilowatt hour (kWh) consumption patterns. Identifying effective measures to detect fraudulent electricity usage is an active area of research in the electrical domain. In this study, Support Vector Machines (SVM), Naïve Bayes (NB) and k-Nearest Neighbour (KNN) algorithms were used to design and propose an electricity fraud detection model. Using the Nelson Mandela Bay Municipality as a case study, three classifiers were built with SVM, NB and KNN algorithms. The performance of these classifiers were evaluated and compared.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1: INTRODUCTION

1.1: BACKGROUND

The Republic of South Africa, commonly referred to as South Africa, is the southernmost country in Africa and is bordered by the South Atlantic and Indian Oceans. South Africa is divided into nine provinces, of which there are 52 districts, comprising eight metropolitan and 44 district municipalities. Eskom, is the primary energy supplier in the country, generating almost 95% of South Africa's energy. It also transmits and distributes electricity to industrial, mining, commercial, agricultural and residential customers and redistributors. Municipalities purchase electricity from Eskom to distribute to its customers, thereby generating revenue to supplement their operational expenses. Each year, municipalities and Eskom lose revenue from electricity theft. Electricity theft is one of the major problems faced by not only South Africa but the rest of the world, in both developing and developed countries alike. The annual Emerging Markets Smart Grid: Outlook 2015 study revealed that electricity theft resulted in global losses of almost US$90 billion (T&DWorldMagazine, 2015). According to this study, India suffered annual losses amounting close to $16.2 Billion, followed by Brazil ($10.5 Billion) and Russia with $5.1 Billion (T&DWorldMagazine, 2015). PowerNews (2013) stated that the United States of America lose almost $6 billion (R61 billion) annually due to electricity theft, whereas in the United Kingdom about £299 million (R4.7 billion) is lost each year from gas and electricity theft.

The Nelson Mandela Bay Municipality (NMBM) is one of the eight metropolitans in South Africa and is located on the coast of Algoa Bay in the Eastern Cape Province. It comprises the city of Port Elizabeth, the nearby towns of Uitenhage and Despatch, as well as surrounding rural areas. The NMBM is also facing issues of electricity theft. According to the regional radio broadcaster, the NMBM loses close to R80 million annually due to electricity theft. In 2014, the metropolitan lost approximately R218 million due to both technical and non-technical losses (AlgoaFM, 2014). These two types of losses are discussed further in the following section.

1.2: NON-TECHNICAL LOSSES

When investigating electricity supply from source to end users, electric energy losses refer to the amounts of electricity injected into the transmission and distribution grids that users have not paid for. These losses are categorised into technical and non-technical (Nagi etal.,2008). Technical losses consist primarily of power dissipation in the conductors and equipment used for transmission and distribution lines. In contrast, non-technical losses are caused

mainly by electricity theft, non-payment by customers, and errors in the billing systems. The minimisation of technical losses is an engineering problem and falls outside the scope of this research. The main focus for this study is to identify a statistical model which can be used to understand better the nature of non-technical losses.

1.3: THE STATEMENT OF THE PROBLEM

In NMBM, electric customers are billed according to their monthly electricity usage in kilowatt hour (kWh). The billing systems used per customer rely on the type of meter used, i.e. credit or prepaid. In the case of credit meters, customers are charged based on consumption meter readings. In the case of pre-paid meters, customers are charged when they buy their energy meter units. These records are then stored in NMBM's billing systems and databases. However, some customers tamper with their meters or perform illegal connections and are not billed for the amount of electricity they consume. As a result, the NMBM experiences a loss in revenue. . In an attempt to reduce this non-technical loss, the municipality contracted Amat Security and Cleaning Services (Pty) Ltd (Amat), a company with specialist expertise to track and expose fraudulent clients.

Amat conducts on-site inspections of electric meters and checks and reports any irregularities. However, inspections are carried out at random, are time consuming and difficult to conduct. Although the NMBM's database comprises a large amount of stored data, the municipality currently does not have an intelligent system of extracting meaningful patterns or information from these databases in order to identify non-technical losses. An intelligent system incorporating statistical methods will help the NMBM identify suspicious consumption patterns, focus customer inspections and target most likely fraudulent customers. Such a model is proposed in this study.

1.4: OBJECTIVES OF THE STUDY

The main objective of this study was to propose a model to detect and predict fraudulent activities by studying abnormalities in customers' monthly kWh consumption patterns.

Support Vector Machines (SVM), Naïve Bayes (NB) and k-Nearest Neighbour (KNN) statistical classification algorithms were used to develop three classifiers. These classifiers were compared and the classifier with the highest fraud detection rate and lowest misclassification rate was selected as the most suitable fraud detection model.

1.5: CHAPTER SUMMARY

The remainder of this dissertation is as follows. Chapter 2 provides a comprehensive literature review of studies assessing non-technical losses and Chapter 3 presents the research methodology and data. The results of this study and corresponding discussion are provided in Chapter 4, and the conclusion in Chapter 5.

CHAPTER 2: LITERATURE REVIEW

2.1: INTRODUCTION

Analysing non-technical losses is a topical area of research in the electricity sector and involves finding patterns in electricity consumption data. Data mining is a computational process that discovers patterns in such large data sets, and incorporates artificial intelligence, machine learning, statistics and database systems. Data mining aims to find insights that are statistically reliable from the data (Elkan, 2001). As there is no known statistical distribution associated with electricity consumption patterns, data mining techniques are therefore often the preferred method of analysis, when compared to the alternative model-based statistical methods (Jaya & Tamilselvi, 2013).

Many studies investigating the detection of non-technical losses have been conducted using electrical engineering methods. These include; the use of central observer meters (Bandim, et al., 2003), vigilant energy metering (Anand, De, & Naveen, 2003), power line impedance (Pasdar & Mirzakuchaki, 2007), smart meters (Depuru, Wang, & Devabhaktuni, 2010), and many others. However, there are relatively few published research studies, particularly in South Africa, focussed on the use of data mining techniques to detect non-technical losses. Where available, studies related to the detection of non-technical losses, and potential data mining techniques, are reviewed in this chapter. The majority of this chapter is on critical evaluation of the different methodologies used in analysing non-technical losses so as to identify the appropriate approach for investigating these losses in NMBM.

2.2: SUPPORT VECTOR MACHINES

Statistical learning theory was introduced by Vapnik in the 1960's (Vapnik, 1999). This is a theoretical analysis of the problem of function estimation from a given collection of data (Vapnik, 1999). In the 1990's, new types of learning algorithms based on the statistical learning theory named Support Vector Machines (SVM), were proposed. The SVM is a classification method used in fraud identification and prediction, for classification of both linear and nonlinear data. This technique uses nonlinear mapping to transform the original training data into a higher dimension. It then searches for the linear optimal separating hyperplane on this new dimension. This hyperplane is called the decision boundary and it separates tuples of one class from another. The SVM finds the decision boundary using important training tuples called the "support vectors" and the "margin". The support vectors and the margins will be formally defined in Chapter 3. Although the foundation for the SVM method has been around since the 1960's, the first paper on this method was presented in

1992 by Boser, Guyon and Vapnik. Since 1992, the SVM classification method gained popularity in many applications such as; text classification (Joachims, 1998), gene analysis (Guyon et al, 2002), facial expression recognition (Michel & Kaliouby, 2003), and many others (Fradkin & Muchnik, 2006).

Nagi et al. (2008) conducted a study to detect non-technical losses using SVM in the power system of Tenaga Nasional Berhad in Malaysia. The proposed SVM classifier identified transactions of potential fraudulent customers in order to facilitate an onsite inspection. This study reported that out of the total number of candidates shortlisted by the SVM classifier, 53% were confirmed, upon inspection, to be fraudulent cases. That percentage is called the fraud detection hit rate. A similar study by the same authors was conducted in 2010 (Nagi et al.,2010). The main objective of this study was to improve the hit rate achieved in the previously mentioned 2008 study. In addition to monthly energy consumptions, theft of electricity information, credit worthiness rating information, high risk customer information and irregularity report information, were all included in the fraud detection model. The inclusion of the additional data in the fraud detection model improved the hit rate from 53% to 60% (Nagi et al, 2010).

Depuru, Wang and Devabhaktuni (2011) addressed some of the factors associated with the detection of electricity theft such as; geographical location, size of customer [agricultural (small, medium), commercial (small, medium, large) and residential (very small, small, medium, large)] and, season of the year. In their study, the SVM classifier was trained and tested on data sets collected by smart meters in different geographical locations in India. Smart meters are used by electric utilities to record electricity consumption during specific intervals of time. Electrical energy consumption patterns under normal operations and under electricity theft were studied and customers classified into three groups. A training set consisting of 440 customers was used to train the SVM classifier. Then, the trained SVM classifier was used on a test set consisting of 220 customers. Accuracy is defined as the proportion of customers that are correctly classified by a classifier. The learned SVM classifier achieved an accuracy of 98.4%.

Several studies on SVM have been conducted in other domains. Bhattacharyya et al (2011) used SVM to detect credit card fraud. The SVM classifier was compared with the Random Forests (RF) and Logistic Regression (LR) classifiers. The SVM classifier proved to be competitive with the RF and LR classifiers. In telecommunications, the SVM classifier was used to detect fraudulent subscriptions of customers (Hamid & Sepehri, 2011). The results demonstrated that the SVM classifier had the best performance in comparison to the

Decision Tree classifier and the Neural Network classifier. Ravisankar et al (2011) aimed to identify companies that resorted to financial statement fraud. They studied and compared the Neural Network, SVM, LR, Generic Programming and Group Method of Data Handling. A data set of companies in China was used to train and test the classifiers and the SVM classifier performed the best.

2.3: NAÏVE BAYES

In statistics, the NB classifiers are probabilistic classifiers, based on applying Bayes Theorem with strong (naïve) class independence assumptions between the features. The NB method was introduced in the 1960's, and since then it has been applied to fraud detection. The NB classifiers have been found to be competitive, more advanced data mining techniques, such as SVM. These classifiers are easy to implement and have fast processing time (Rennie et al., 2003).The following section reviews some of the published research studies on the NB classifiers.

Nizar, Zhao and Zhang (2007) used the NB in conjunction with the DT algorithm to determine what arrangement of load profile data provided the most successful fraudulent detection scenario. In their study, the data was rearranged into different time frames that were averaged over days, weeks, months and years. The DT algorithm proved to be more accurate when the data was arranged into daily and monthly averages. The limitation to the success of the DT approach was the computational time, which was much slower than the NB classifier. For both methods, the lowest accuracy was found when the data was arranged on a yearly basis. The highest accuracy for both classification systems was for daily average consumptions.

A case study on automobile insurance claims was conducted by Viaene, Derrig and Dedene (2004). The NB classifier was used to identify and detect fraudulent claims. This was a comparative study, with the NB classifier compared to the AdaBoosted Naïve Bayes (AB) and AdaBoosted weights of evidence (ABWOE). The ABWOE had the best performance, with accuracy of 84.43%, followed by the AB, with 84.41% and lastly the NB classifier, at 83.03% accuracy. It is important to note that the difference between accuracy of these classifiers was very small.

2.4: K-NEAREST NEIGHBOUR

The k-Nearest Neighbour (KNN), first proposed by Fix and Hodges (1952), is a non-parametric statistical method used to estimate probability density functions for statistical

classification. The KNN method is an intuitive, easily explained procedure (Henley & Hand, 1996). In this algorithm, unknown tuples are predicted according to the class of its k nearest neighbours. Figure 2.1 illustrates the 3-nearest neighbours and 7-nearest neighbours of the unknown tuple, which is placed at the centre of circle (the red star). For the case of 3-nearest neighbours, there are two neighbours of class B and one neighbour of class A. As the number of nearest neighbours for class B is greater than the number of nearest neighbours for class A, the unknown tuple is classified as class B. For the second case, there are four tuples belonging to class A and three tuples belonging to class B, and as the number of nearest neighbour for Class A is greater than that of Class B, the unknown tuple is given class B. To avoid a tie between classes it is advisable to use an odd number for the value of k. The illustrations depicted in Figure 2.1 show the importance of the value of k.

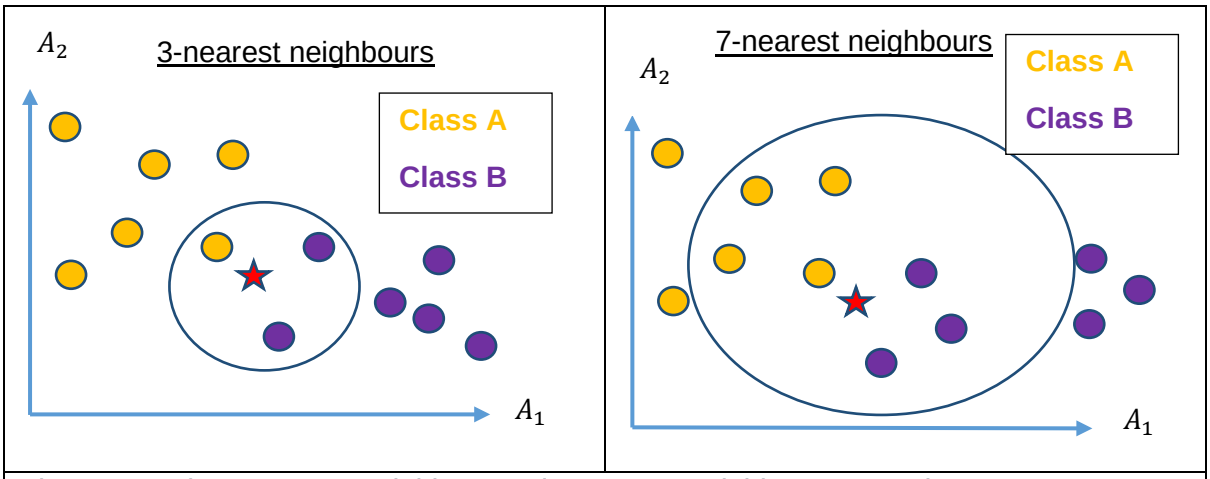


Figure 2.1: The 3-nearest neighbour and 7-nearest neighbour comparison

A distance metric used to measure similarity between the unknown tuples and the training tuples is another important parameter in the KNN method as it influences the performance of the classifier. In an attempt to find the best distance metric to use for subsequent training of the KNN classifier, Mulak and Talhar (2015) compared the Euclidean distance, Chebychev distance and Manhattan distance (also known as the City Block distance). These distance metrics were compared in terms of accuracy, sensitivity and specificity. For a binary classification problem with positive and negative classes, accuracy is defined as the proportion of all correctly classified tuples. “Sensitivity” is the proportion of all positive tuples that are correctly classified and Specificity is the proportion of all negative tuples that are correctly classified. Sensitivity and specificity are performance measures and are formally defined in Chapter 3. The KNN classifier was trained using the three mentioned distance metrics on a dataset from Knowledge Discovery in Databases (KDD). The results in terms of accuracy, sensitivity and specificity are depicted in Figure 2.2, 2.3and 2.4 respectively (Mulak & Talhar, 2015).

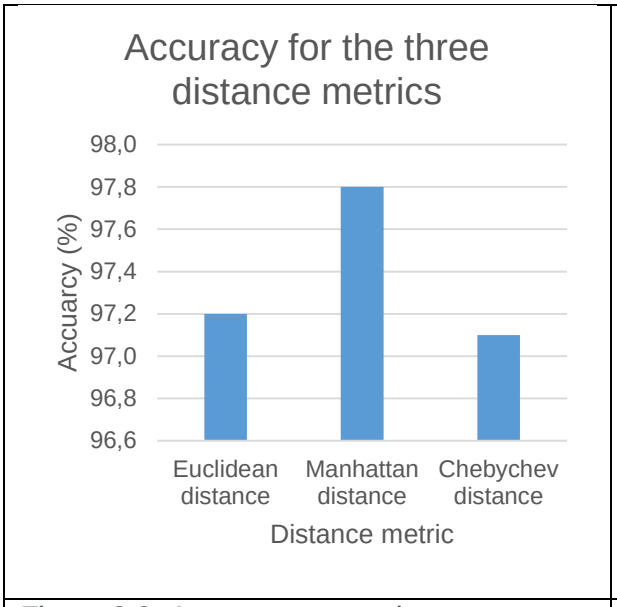


Figure 2.2: Accuracy comparison

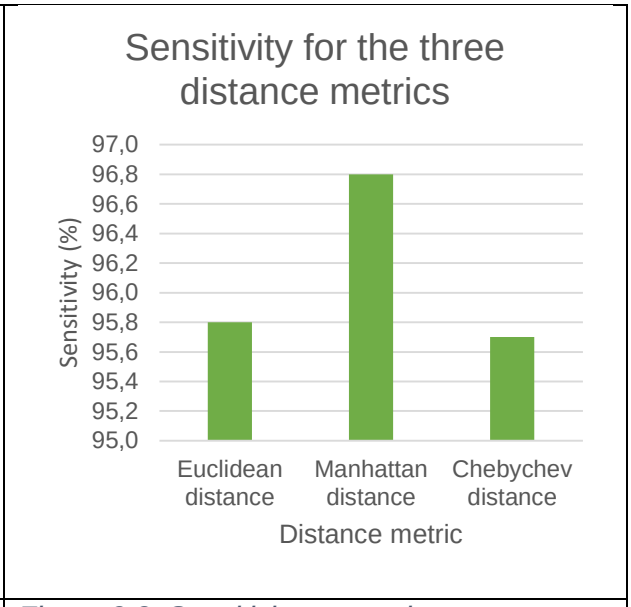


Figure 2.3: Sensitivity comparison

It is evident from Figure 2.2 that the Manhattan distance was the most accurate. The comparative results using sensitivity are shown in Figure 2.3. In terms of sensitivity, the Manhattan distance was the most successful when compared to the Euclidean and Chebychev distances. , The specificity rate is described for the three distance metrics Figure 2.4 and again, the Manhattan distance proved better than the other two distance metrics.

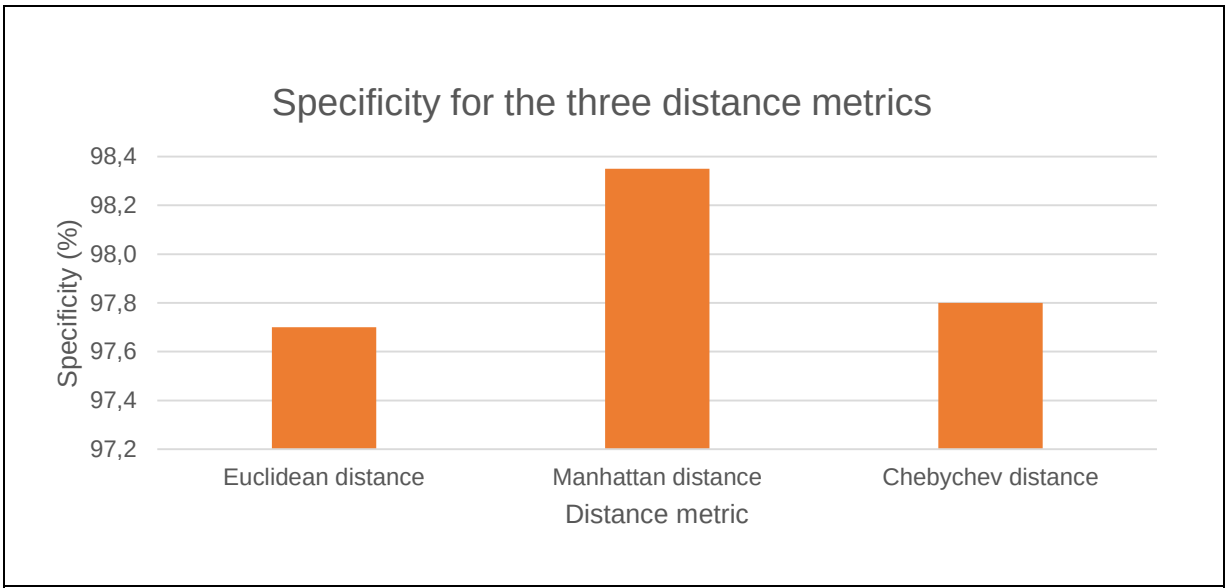


Figure 2.4: Distance metrics compared in terms of specificity (Mulak & Talhar, 2015)

Henley and Hand (1996) proposed the KNN classifier as a method to assess credit worthiness of consumers. The KNN classifier was compared with traditionally used methods such as the Linear Regression, LR and DT. A data set consisting of “bad” and “good” credit

consumers from a large mail order company was used. The KNN method's performance was better in predicting consumer credit risk when compared to the traditional methods. The KNN classifier was compared with the Fuzzy Multivariate Rule Building Expert System (FuRES), Artificial Neural Networks (ANN), Canonical Variates Analysis (CVA), Classification and Regression Trees (CART) and the Discriminate Partial Least Squares (DPLS) in classification of pyrolysis mass spectra (Alsberg et al, 1997). Two pyrolysis mass data sets were used to solve the classification problem. The DPLS and ANN methods achieved the highest accuracy in classification of both data sets, followed by FuRES and KNN methods. Although the KNN classifier did not achieve the highest accuracy, the results showed that it was competitive with the other methods.

2.5: RELATED RESEARCH STUDIES IN SOUTH AFRICA

Davidson (2002) proposed a method to estimate non-technical losses. In the study, sources of non-technical losses were identified as; non-payment of electricity bills, unauthorised line tapping and diversion, and, losses due to faulty meters and equipment etc. According to Davidson (2002), the cost associated with non-technical losses can be estimated using the following equation

$$C_{NTL} = U_{Cost} \times E_{Loss} + M_{Cost} - C_{TLoss}$$

where,

$U_{Cost} =$ Unit cost of electricity,

$E_{Loss} =$ Total system loss defined as the difference between energy generated and delivered,

$M_{Cost} =$ Maintenance and additional operation costs,

$C_{TLoss} =$ Technical loss cost component.

To estimate the cost of non-technical losses, the magnitude of technical losses, maintenance costs, operational costs, and other additional costs, needs to be evaluated.

The relationship between the distribution of data and a classifier performance was studied by van der Walt and Barnard (2006). The data was simulated using a multivariate Normal distribution. Six different classifiers, namely the NB classifier, the Gaussian classifier, the DT, the KNN classifier, the multi-layer perceptron (MLP) and SVM, were used to design three experiments with different scenarios. In experiment 1, data sets for correlated and uncorrelated variables were simulated and each data set had three classes. For the data with correlated variables, the Gaussian classifier had the lowest error rate in comparison to

the other classifiers. The error rate is defined as the number of observations that are incorrectly classified, and will be formally defined in Chapter 3. The NB classifier achieved the lowest error rate over all the uncorrelated data sets in experiment 1. Experiment 2 involved various data sets with noise added to the output data. The aim of this experiment was to study the effect of noise in choosing the optimal value of k in the KNN classifier. It was found that the value of k increases monotonically as the noise in the output data increases. In the third and final experiment, two-dimensional data with different standard deviation was used to study the effect of a constant distance metric use by the KNN classifier. It was found that the KNN classifier performs poorly with high error rate when a constant distance metric is used over different data sets (van der Walt & Barnard, 2006).

Doorduin et al.(2004) used simulations and models based on prepaid electricity consumption to discuss the feasibility of using remote check meters to measure the magnitude of electric energy losses and electricity theft in a low voltage reticulation network. The study concluded that mobile remote check meters can be used to detect illegal consumption of electricity. Potential fraudulent consumers can then be inspected to confirm the simulated results. Another study involved the use of the DT and the NB classifier to detect malware infections on a network (Stalmans & Irwin, 2011). The analysis was performed on network traffic from a large South African university as well as a local school. The results demonstrated that the NB classifier outperformed that of the DT.

There is no study known by the author that used data mining algorithms for the detection of non-technical losses in South Africa. However, some of these algorithms have been employed in other areas, such as image segmentation (van der Merwe & Engelbrecht, 2003), computer science (Stalmans & Irwin, 2011), and medical diagnostic (Johnson, 2012).

2.6: CHAPTER SUMMARY

The objective of this study was to propose an electricity fraud detection model. A review of the literature demonstrated that data mining and nonparametric statistical methods have been used extensively in fraud detection. The most popular methods include the Artificial Neural Networks, Decision Trees, Support Vector Machines, Naïve Bayes, and k-Nearest Neighbour. In electricity fraud detection, the most widely used nonparametric statistical method is Support Vector Machines. The Support Vector Machines have been found to be theoretical, easy to analyse, data driven and robust (Jaya & Tamilselvi, 2013). In fraud detection, this method outperforms other classifiers such as the Artificial Neural Networks, Logistic Regression and Discriminant Analysis (Ravisankar et al, 2011). The originality of this study is in the use of the Support Vector Machines method together with the Naïve

Bayes method, which is fast and easy to use (Rish, 2001) and the k-Nearest Neighbour method, to propose a model for the detection of fraudulent electricity consumption. There is no evidence of the use of these methods in the detection of electricity fraud in South Africa.

CHAPTER 3: RESEARCH METHODOLOGY AND DATA

3.1: INTRODUCTION

Statistical data classification is a form of data analysis that can be used to extract useful information from data. Such analysis is used to build models or classifiers to describe important data classes. For instance, a classifier can be used to classify a bank loan application as either “safe” or “risky”. Another example would be assigning an email to “spam” or “not spam”, or classifying a credit card transaction as either “fraudulent” or “not fraudulent” (Jaya & Tamilselvi, 2013). There are many applications of data classification, including fraud detection, target marketing, manufacturing and medical diagnosis etc. In this study, statistical data classification was used to analyse non-technical losses. In this chapter, techniques for statistical data classification and the background theory to analyse non-technical losses are discussed.

3.2: STATISTICAL DATA CLASSIFICATION

Data classification is a two-step process. The first step, called the “training phase” involves building a classifier from a set of data using a classification algorithm. In this step a classification algorithm builds a classifier by learning from a training set comprising database tuples and their corresponding labels (Kohavi, 1995). A tuple is characterised by an n-dimensional vector, $X = (x_1, x_2, \dots, x_n)$ showing n measurements made on the tuple from n database attributes, $A_1, A_2, \dots, A_n$, respectively. Each attribute represents a feature of X. Each tuple is assumed to belong to a predefined class which is determined by another attribute called “class label”. Each value on the “class label” attribute serves as a category or a class, and these values are discrete and unordered. The individual tuples making up the training set are called the training tuples. There are two types of training phase, namely “supervised learning” and “unsupervised learning”. In “supervised learning”, the class label of each tuple is known, whereas in “unsupervised learning”, the class labels are not known in advance (Kohavi, 1995).

In the second step, a test set is used to evaluate performance of the learned classifier, this is called the “testing phase”. This set consists of test tuples and associated class labels. These tuples are randomly selected from the general data set and were not used to build the classifier. It is conventional to designate two thirds of the data as training set and the remaining one third as a test set (Kohavi, 1995). “Accuracy of a classifier on a given test set is the proportion of test set tuples that are correctly classified by the classifier” (Han &

Kamber, 2006). When evaluating performance, the class label of each test tuple is compared with the class label predicted by a classifier for that tuple. If the accuracy of a classifier is acceptable, then that classifier can be used to predict data tuples with unknown class labels. Data with unknown labels are referred to as “previously unseen” data. In addition to accuracy, sensitivity and specificity were used to evaluate performance of classifiers. Sensitivity and specificity are formally defined in Section 3.8.

3.3: DATA ACQUISITION AND INTEGRATION

The data was acquired from the NMBM in Microsoft Office Excel format and stored in relational tables in Microsoft SQL Server Management Studio 2014. Pre-processing was performed in order to transform the raw data into appropriate format for the subsequent analysis. The data consisted of 247 552 customers for a period of 24 months from March 2013 to February 2015.

Depicted in Table 3.1 is the structure of the data for a particular month. The “ATTP Customer” column indicates whether or not a customer was eligible for free basic electricity. Those who were eligible, were appointed a “Yes” and any remaining customers, a “No”. The tariff code was assigned to a customer based on where the customer resided. Units billed indicated the customer’s electricity consumption for the whole month in kWh. The customer’s electricity consumption for all the months on the database were merged into one table using SQL Server 2014 Management Studio queries. Each customer was given an identity called customers’ id, which were integers ranging from 1 to 247 552. The merged data is summarised in Table 3.2.

Table 3.1: Electricity customers’ information for a particular month

Meter number	**ATTP customer**	**Tariff code**	**Units billed (kWh)**	**Address**
Meter Number 1	Yes	T01	36.00	Address 1
⋮	⋮	⋮	⋮	⋮
Meter Number 247 552	No	A32	256.71	Address 247 552

Table 3.2: Electricity customers’ consumptions in kWh

Customer’s Id	**Mar 2013**	**Apr 2013**	**May 2013**	**Jun 2013**	...	**Feb 2015**
1	467.7	470.5	432.9	418.0	...	512.7
⋮	⋮	⋮	⋮	⋮	...	⋮
247 552	595.9	676.6	679.9	710.5	...	788.8

The following line graph (Figure 3.1) illustrates the total consumption for each of the months recorded in Table 3.2. The integers on the horizontal axis correspond to the 24 months of the period starting from March 2013 and ending in February 2015, where, month one corresponds to March 2013.There is is no apparent seasonality within each month, as the data show some cyclic behaviour with a period of 12 months. There was no apparent trend in the data during this period.

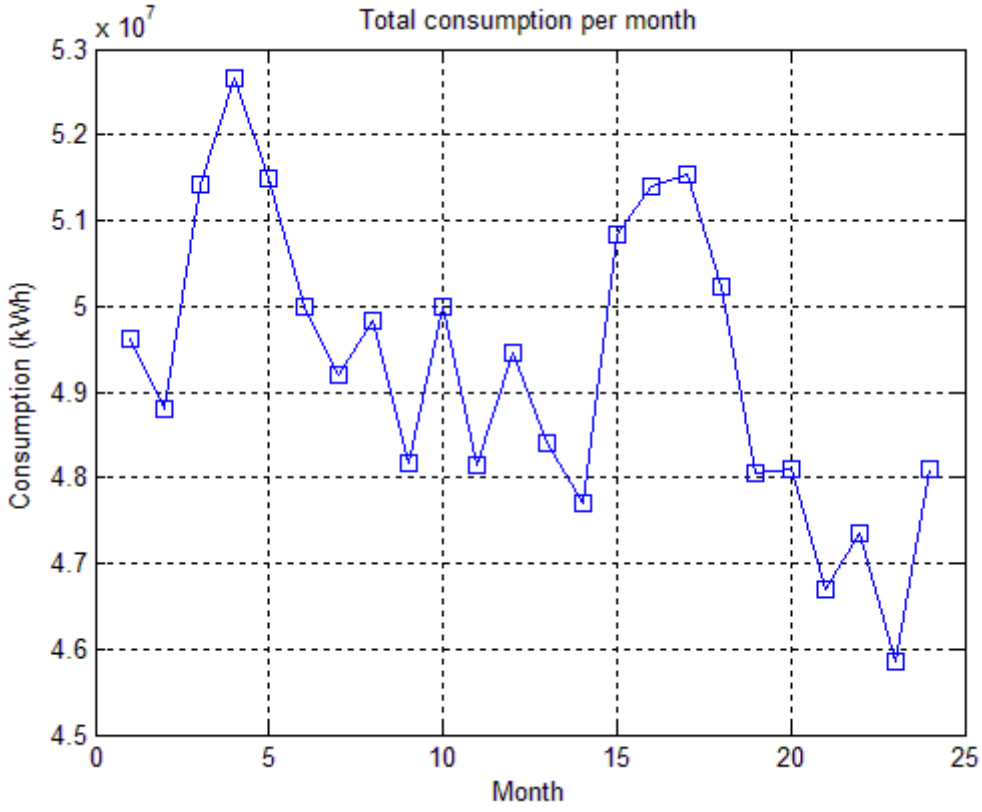


Figure 3.1: Line graph for total consumption in kWh per month

The data was pre-processed prior to statistical assessment in order to correct inconsistencies and to remove noisy data (i.e. data with incorrect attribute values). Noisy data arise from a number of sources, for example; the wrong address or multiple addresses assigned to a meter number, a faulty data collection instrument, human or computer errors at data entry, or during data transmission, inconsistencies in naming convention or data code used, and, inconsistent formats for input fields such as date.

The data in Table 3.2 was filtered for extraction of customer load profiles and features. SQL Server 2014 Management Studio queries were used to remove all the duplicates in the monthly kWh consumption data, all the customers that had zero consumption (0 kWh) throughout the whole 24 month period, and all the customers who registered after the first month. Duplicates were removed to avoid redundancy in the data. All customers with zero

consumption throughout the period, and the customers who registered after the first month, were manually inspected and did not form part of the subsequent analysis. After customer filtering, only 110 740 customer records were found usable for statistical assessment and these formed the population of this study.

Inspections were performed on randomly selected customers in the Nelson Mandela Bay region in order to identify fraudulent and honest customers. A total of 3 156 customers were inspected during the period dated March 2015 to December 2015. Of these, 2420 were classified as clean cases and 736 as fraudulent. The status of the electricity meters for these customers was tested using electrical engineering devices. All customers found to have committed fraud were tagged in the database as “fraudulent” and honest customers as “clean”. The 3 156 inspected customers were extracted from the population of 110 740 customers Table 3.4. Each inspected customer was given either a “-1”, if the customer was clean, or a “1”, if the customer was fraudulent. This coding was used to transform categorical values into numerical values. The structure of this data is shown in Table 3.3.

Table 3.3: The sample data used for statistical analysis

Customer id*	Mar-13 (kWh)	Apr-13 (kWh)	...	Feb-15 (kWh)	Class label
1	240.5	192.5	···	334.4	-1
2	787.2	862.1	···	292.9	1
3	459.3	419.1	···	711.0	-1
4	128.5	128.5	···	121.5	1
⋮	⋮	⋮	...	⋮	⋮
3 155	153.9	96.2	···	83.6	-1
3 156	115.1	141.8	···	158.8	1

Daily average consumptions were computed for each customer on the 24 month database. This was achieved by taking the monthly consumption and dividing it by the number of days in each corresponding month. These daily average consumptions were selected as features related to customer’s load profiles (Nagi et al, 2008). A customer’s load profile was characterised by the vector $X_i = [x_{i1}, x_{i2}, \dots, x_{in}]$, $i = 1,2, \dots, N$, where N =3 156 was the total number of customers in the sample and $n = 24$, the total number of months. Each customer had a load profile characterised by 24 features. Table 3.4 documents typical daily load profiles for different customers.

Table 3.4: Daily load profiles of 3 156 customers with 24 features

Customer id*	Mar-13 (kWh)	Apr-13 (kWh)	...	Feb-15 (kWh)	Class label
1	7.7581	6.4167	···	11.9429	-1
2	25.3935	28.7367	···	10.4607	1
3	14.8161	13.9700	···	25.3929	-1
⋮	⋮	⋮	...	⋮	⋮
3 156	3.7129	4.7267	···	5.6714	1

3.4: DATA NORMALISATION

In statistical classification, data is normalised into a form appropriate for analysis. Each load profile is normalised by scaling its values so that they fall within a small specified interval. Normalisation speeds up the training phase and prevents attributes with initially large values from outweighing those with initial small ones. Various techniques are used for data normalisation, such as min-max normalisation, z-score normalisation and normalising by decimal scaling (Han & Kamber, 2006).

If a load profile, X_i has a minimum value of min_{x_i} and a maximum value of max_{x_i}, then, in min-max normalisation, the original data is linearly transformed by mapping a value, x_{ij}, of X_i to x_{ij}^* in the range $[min_{x_i}^*, max_{x_i}^*]$ by calculating

$$x_{ij}^* = \frac{x_{ij} - min_{x_i}}{max_{x_i} - min_{x_i}} \left(max_{x_i}^* - min_{x_i}^*\right) + min_{x_i}^*.$$

In z-score normalisation, the values of an attribute are normalised using the mean and the standard deviation of the attribute. A value, a, of an attribute, A , is normalised to a^* by computing

$$a^* = \frac{a - \mu_A}{\sigma_A},$$

where μ_A and σ_A are the mean, and standard deviation, of the attribute A respectively.

Normalising, by decimal scaling, moves the decimal point of the original value. The original value, x_{ij} ,of a load profile, X_i , is normalised to x_{ij}^* by calculating

$$x_{ij}^* = \frac{x_{ij}}{10^t},$$

where t is the smallest integer, such that the $\max(|x_{ij}^*|) < 1$.

The z-score normalisation is useful when the minimum and maximum values of the attributes are not known. Normalisation by decimal scaling and z-score normalisation tend to change the values of the original data by a large margin. In contrast, the min-max normalisation preserves the relationships between the attributes. Another advantage of the min-max normalisation is that the new interval can be determined in advance, which is not possible in the other two methods.

After considering the limitations and advantages of the three methods, it was decided to use the min-max normalisation to normalise the daily average consumptions shown in Table 3.4. These daily average consumptions were normalised such that all the values fell within the interval $[0,1]$ (Nagi et al, 2008). The daily average consumptions were normalised as follows

$$NL = \frac{L - \min(L)}{\max(L) - \min(L)},$$

where L represents the daily average consumption of a customer, $\min(L)$ and $\max(L)$ are minimum and maximum values in the feature set respectively. The normalised data was then used for statistical analysis, the methods of which are discussed in the following sections. The theory of SVM is presented in the following Section, 3.5, and the NB and the KNN methods are discussed in Sections 3.6 and 3.7 respectively. These three methods were implemented in the subsequent building and testing of the SVM, NB and KNN classifiers.

3.5: SUPPORT VECTOR MACHINES

The SVM method is used to classify linear and nonlinear data. For linear data, two cases exist - the case when the data is linearly separable, and when it's linearly inseparable (Boser et al.,1992). In this section, the methodology followed to classify linear and nonlinear data using linear and nonlinear SVM classifiers respectively, is discussed. The linear SVM classifier for separable cases is presented in Section 3.5.1 and for inseparable cases, Section 3.5.2. The nonlinear SVM classifier is introduced in Section 3.5.3.

3.5.1: LINEARLY SEPARABLE DATA

In this section, the SVM method is discussed using a binary problem for the case when data is linearly separable. Let D be a given training set with associated training tuples X_i and class labels, y_i, $i = 1,2,3,\dots,d$, where D is defined as follows

$$D = \begin{bmatrix} X_1 & y_1 \\ X_2 & y_2 \\ \vdots & \vdots \\ X_d & y_d \end{bmatrix}.$$

Each y_i can take one of two values, either -1 or 1, that is, $y_i \epsilon \{-1,1\}$, corresponding to "clean" and "fraudulent" cases respectively. To help with visualisation, let's consider a case where there are only two input attributes A_1 and A_2, as depicted in Figure 3.2. The data shown in Figure 3.2 is linearly separable as there exists a straight line separating all the tuples of class 1 from those of class 2. It is evident that there are infinitely many straight lines that can be drawn to separate the tuples of these two classes. When the SVM classifier s trained, it searches for the best straight line to separate the data with minimum classification error on previously unseen tuples. This best straight line is called the decision boundary. If there were only three input attributes, the decision boundary would be the separating plane. In general, in an n-dimensional space, where there are n input attributes, the decision boundary is a hyperplane.

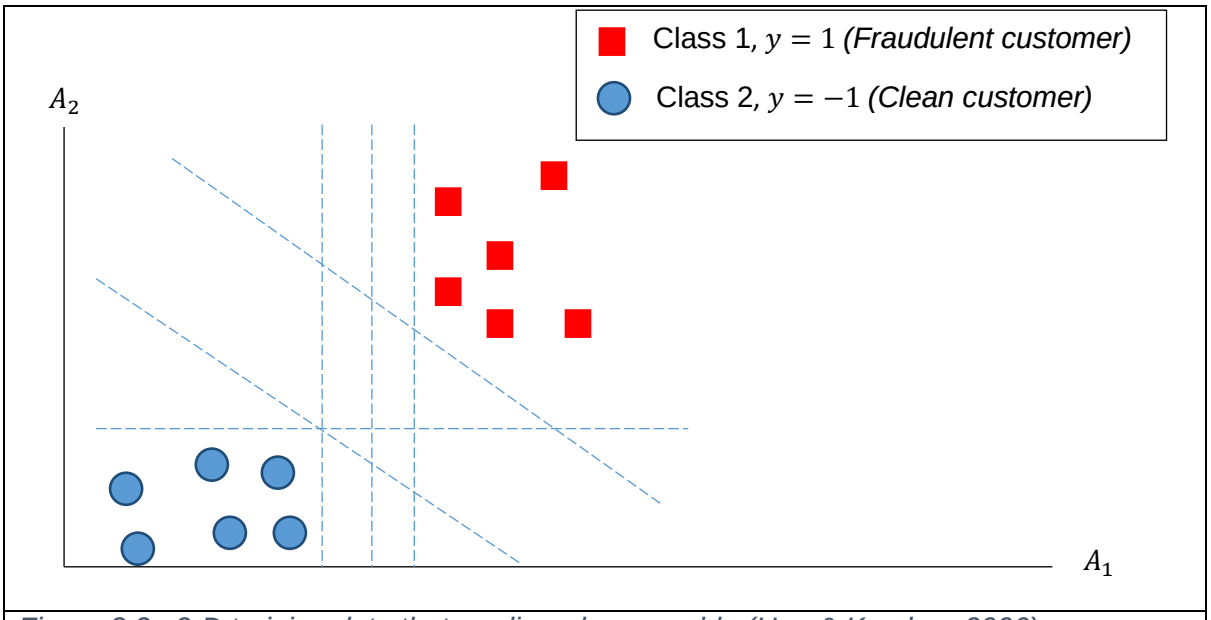


Figure 3.2: 2-D training data that are linearly separable (Han & Kamber, 2006)

In the SVM method, the decision boundary is the hyperplane with maximum margin. To understand the concept of a margin consider Figure 3.3 and 3.4.These illustrate two possible hyperplanes and their associated margins. Both hyperplanes correctly classified all the data tuples, with no classification error. However, the hyperplane in Figure 3.3 has a smaller margin when compared to the hyperplane in Figure 3.4.

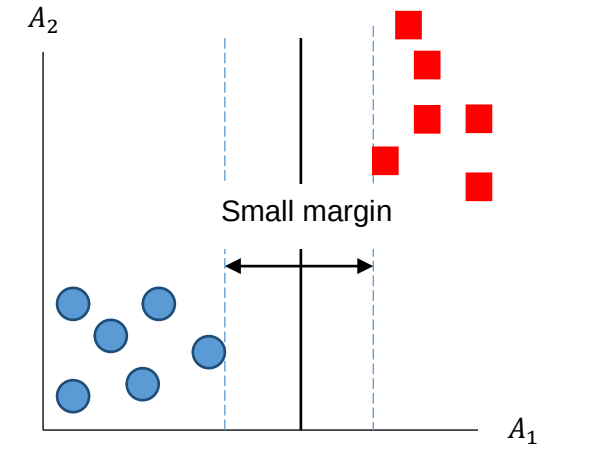


Figure 3.3: Small margin hyperplane

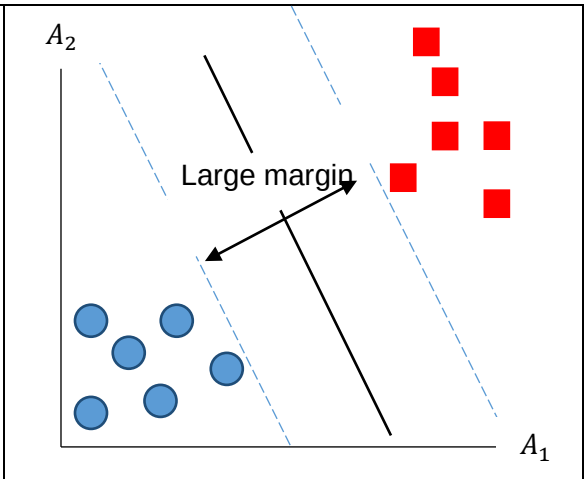


Figure 3.4: Large margin hyperplane

The hyperplane with the larger margin is more accurate at classifying future data tuples than that with the smaller margin. This is because hyperplanes with large margins are likely to have a better generalisation error than those with small margins. Generalisation error is defined as a measure of how accurately an algorithm is able to predict output values for previously unseen data (Steinbach, Vipin, & Tan, 2006). Classifiers that produce hyperplanes with small error margin are likely to be affected by model over-fitting and tend to generalise poorly on unseen data (Steinbach et al, 2006)

Structural risk minimisation (SRM) is a statistical learning principle that relates the margin of a linear classifier to its generalisation error (Steinbach et al, 2006). The SRM principle provides an upper bound of a generalisation error R, with probability $1-\eta$ as: $R \leq R_e + \varphi\left(\frac{h}{d}, \frac{\log(\eta)}{d}\right)$, where φ is a monotone increasing function of model capacity h, R_e is the training error and d is the number of training tuples. The capacity of a linear classifier is inversely proportional to its margin. Classifiers with small margins have higher capacity and are more capable of fitting many training sets. However, according to the SRM principle, as the classifier's capacity increases, the generalisation error bound will also increase. Therefore is it desirable to have linear classifiers whose decision boundaries have maximum margins and respectively smaller generalisation error bounds. The SVM classifier is one of the classifiers for which the SRM principle holds (Steinbach et al, 2006).

3.5.2: THE LINEAR SVM CLASSIFIER: SEPARABLE CASE

A linear SVM is a classifier that searches for a linear hyperplane with the largest margin. For this reason, a linear SVM classifier is known as a maximum margin classifier. To understand how a SVM classifier searches for such boundary, the definitions of a decision boundary and the margin of a linear classifier are required.

DECISION BOUNDARY OF A LINEAR SVM CLASSIFIER

Consider a binary classification problem consisting of d training tuples. Each tuple is denoted by,$(X_i, y_i), i = 1,2, \dots, d$, where $X_i = [x_{i1}, x_{i2}, \dots, x_{in}]$ corresponds to an attribute set for the i^{th} tuple. Let $y_i \, \epsilon \, \{-1,1\}$ be the class label of a training tuple. For any linear SVM classifier, the boundary decision can be expressed as

$$W \cdot X + b = 0,$$

where W is a weight vector , and b is a scalar, called the bias, and $W \cdot X$ is a dot product of W and X.

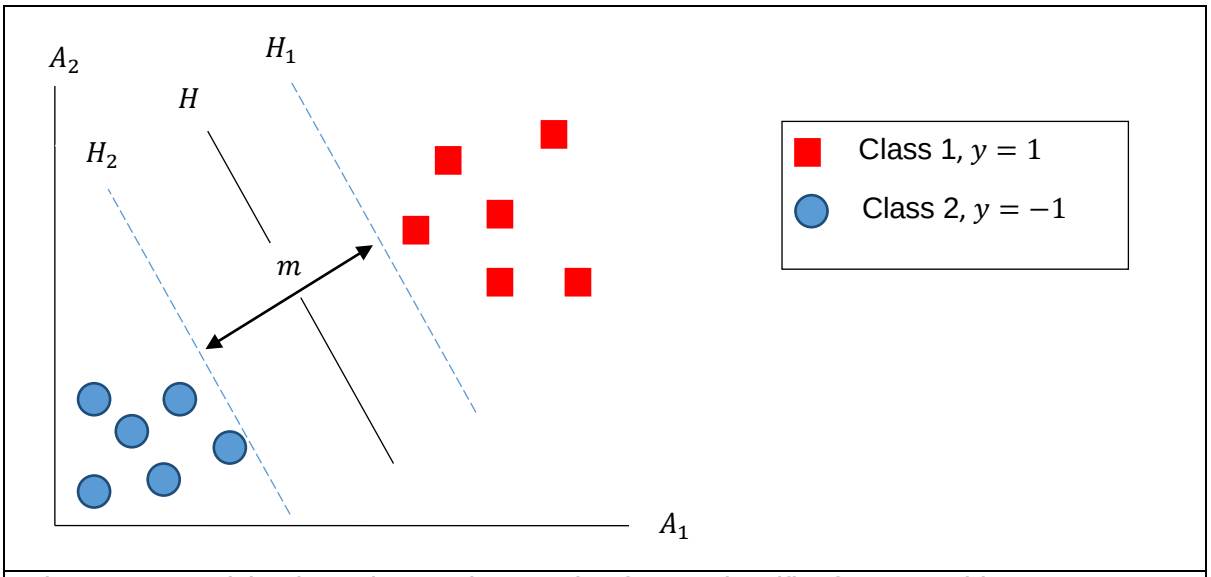


Figure 3.5: Decision boundary and a margin of SVM classifier for separable case

Depicted in Figure 3.5 is a two-dimensional training set consisting of two attributes A_1 and A_2. A decision boundary that separates the training tuples into their respective classes is shown with the solid black line, namely, H. All the tuples located on this line must satisfy the equation, $W \cdot X + b = 0$. For example, if X_a and X_b are two tuples located on the decision boundary then

$$W \cdot X_a + b = 0$$

and

$$W \cdot X_b + b = 0 \, .$$

where $X_b - X_a$ is a vector parallel to the decision boundary and is directed from X_a to X_b (Steinbach et al, 2006). As the dot product $W \cdot (X_b - X_a)$ is equal to zero, the direction of

W is perpendicular to the decision boundary. Any training tuples that belong to class 1 satisfy the following equation

$$W \cdot X_s + b = k,$$

where $k > 0$, and for any training tuple belonging to class 2, the following must hold

$$W \cdot X_c + b = k',$$

where $k' < 0$. Therefore, the class label of any test tuple, Z, can be predicted as

$$y = \begin{cases} 1 \; if \; W \cdot Z + b > 0 \\ -1 \; if \; W \cdot Z + b < 0. \end{cases}$$

THE MARGIN OF A LINEAR SVM CLASSIFIER

The two hyperplanes, H_1 and H_2, which are parallel to the decision boundary, H, on Figure 3.5 can be expressed algebraically as

$$H_1: W \cdot X_1 + b = 1,$$

$$H_2: W \cdot X_2 + b = -1.$$

The margin of the decision boundary,m, is given by the distance between these two hyperplanes. Subtracting the two equations, the margin of the decision boundary, m can be calculated as follows

$$W \cdot (X_1 - X_2) = 2$$

$$\|W\| \times m = 2$$

$$\therefore m = \frac{2}{\|W\|}.$$

LEARNING A LINEAR SVM CLASSIFIER

The training phase of the SVM method involves the estimation of the weight vector, W and the bias, b, from a given training dataset. W and b are called the parameters of the SVM model, and are chosen in such a way that for any training tuple, X_i

$$W \cdot X_i + b \geq 1 \text{ , if } y_i = 1 \text{ and } W \cdot X_i + b \leq -1 \text{ if } y_i = -1.$$

These inequalities can be combined into one inequality, as follows

$$y_i(W \cdot X_i + b) \geq 1, i = 1,2, \dots, d\,.$$

For the SVM classifier, the margin of the decision boundary must be the maximum margin. Finding this maximum margin is equivalent to minimising the following objective function

$$f(W) = \frac{\|W\|^2}{2}.$$

The inequality, $y_i(W \cdot X_i + b) \geq 1$, and the function, $f(W) = \frac{\|W\|^2}{2}$, are used to formally define the learning task involved in the linear SVM method, for the case when data is linearly separable.

Definition 1 (Linear SVM: Separable Case) (Steinbach et al, 2006)

Formally, the learning task in separable linear SVM is the optimisation problem

$$\min_W \frac{\|W\|^2}{2}$$

$$\text{subject to } y_i(W \cdot X_i + b) \geq 1, \qquad i = 1,2, \dots, d.$$

The above problem is a convex optimisation one, as the objective function is quadratic and the constraints are linear. This convex optimisation problem can be solved by the standard Lagrange multiplier method.

Assume that $\lambda_i \geq 0$, for $i = 1,2, \dots, d$. The Lagrangian for the optimisation problem is then given by

$$L_p = \frac{1}{2}\|W\|^2 - \sum_{i=1}^{d} \lambda_i (y_i(W \cdot X_i + b) - 1),$$

where the parameters λ_i are Lagrange multipliers for $i = 1,2, \dots, d$. The Lagrange multiplier method involves writing a constrained problem as an unconstrained one. The problem in Definition 1 is equivalent to the following unconstrained problem

$$\min L_p = \frac{1}{2}\|W\|^2 - \sum_{i=1}^{d} \lambda_i (y_i(W \cdot X_i + b) - 1)\,.$$

To minimise L_p , derivatives with respect to W and b must be taken, and set to zero yielding

$$\frac{\partial L_p}{\partial W} = 0 \Rightarrow W = \sum_{i=1}^{d} \lambda_i y_i X_i,$$

$$\frac{\partial L_p}{\partial b} = 0 \Rightarrow \sum_{i=1}^{d} \lambda_i y_i = 0 .$$

To find feasible solutions for W, b and λ_i, the inequality constraints in Definition 1 re transformed to equality constraints. This transformation lead to the following Karush-Kuhn-Tucker (KKT) conditions, $\lambda_i \geq 0$ and $\lambda_i[y_i(W \cdot X_i + b) - 1] = 0$.

These KKT conditions suggest that the Lagrange multipliers λ_i, are zero, except when $y_i(W \cdot X_i + b) = 1$. Training tuples, where $\lambda_i > 0$ lie on the hyperplanes H_1 and H_2 (refer to Figure 3.5) and are known as support vectors. All the remaining training tuples, that lie neither on H_1 norH_2, have $\lambda_i = 0$. From the equations, $W = \sum_{i=1}^{d} \lambda_i y_i X_i$ and $\lambda_i[y_i(W \cdot X_i + b) - 1] = 0$, it is evident that the parameters W and b depend only on the support vectors. The problem of solving the quadratic optimisation can be simplified by writing the Lagrangian as the function of Lagrange multipliers only (this is known as dual problem). Substituting $W = \sum_{i=1}^{d} \lambda_i y_i X_i$ and $\sum_{i=1}^{d} \lambda_i y_i = 0$ into $\min L_p = \frac{1}{2}\|W\|^2 - \sum_{i=1}^{d} \lambda_i (y_i(W \cdot X_i + b) - 1)$,yields the dual Lagrangian

$$L_d = \sum_{i=1}^{d} \lambda_i - \frac{1}{2} \sum_{i,j} \lambda_i \lambda_j y_i y_j X_i \cdot X_j .$$

The main differences between primal and dual Lagrangians are summarised in Table 3.5.

Table 3.5: The main differences between primal and dual Lagrangians

Primal Lagrangian	**Dual Lagrangian**
Involves only Lagrange multipliers and training data.	Involves Lagrange multipliers, training data and the parameters of decision boundary.
Minimisation problem.	Maximisation problem.

To obtain the Lagrange multipliers λ_i, the dual optimisation problem is solved using quadratic programming numerical methods. Once λ_i are known, the support vectors are used to obtain the parameters W and b of the decision boundary using the equations, $W =$

$\sum_{i=1}^{d} \lambda_i y_i X_i$ and $\lambda_i[y_i(W \cdot X_i + b) - 1] = 0$. Therefore, the decision boundary can now be written as, $\sum_{i=1}^{d} \lambda_i y_i X_i \cdot X + b = 0$, and any test tuple, Z, classified as

$$f(Z) = sign(W \cdot Z + b) = sign\left(\sum_{i=1}^{d} \lambda_i y_i x_i \cdot z + b\right).$$

If $f(Z) = 1$, then the test tuple is classified as a positive class, that is class 1, and if $f(Z) = -1$, the test tuple classified as a negative class, namely, class 2.

3.5.3: THE LINEAR SVM CLASSIFIER: INSEPARABLE CASE

The data shown in Figure 3.6 is similar to that in Figure 3.5. However, two new tuples, P and Q, are illustrated in Figure 3.6. This is an example of a data set that is linearly inseparable, as the linear hyperplane with the maximum margin cannot correctly classify all the tuples. This hyperplane, H, misclassify both P and Q with some training error. Note that even though the linear hyperplane, I, with a smaller margin, classify all the tuples with zero training error, H is still the preferred hyperplane as a decision boundary due to its larger margin and corresponding smaller generalisation error. The formulation of the SVM classifier in the previous Section constructs decision boundaries that did not accommodate training errors. To incorporate small training errors in the model, a soft margin approach is used, allowing the SVM method to construct a linear decision boundary even when the data is not linearly separable. Therefore, the problem in Definition 1 is re-formulated to incorporate the case when the data is linearly inseparable.

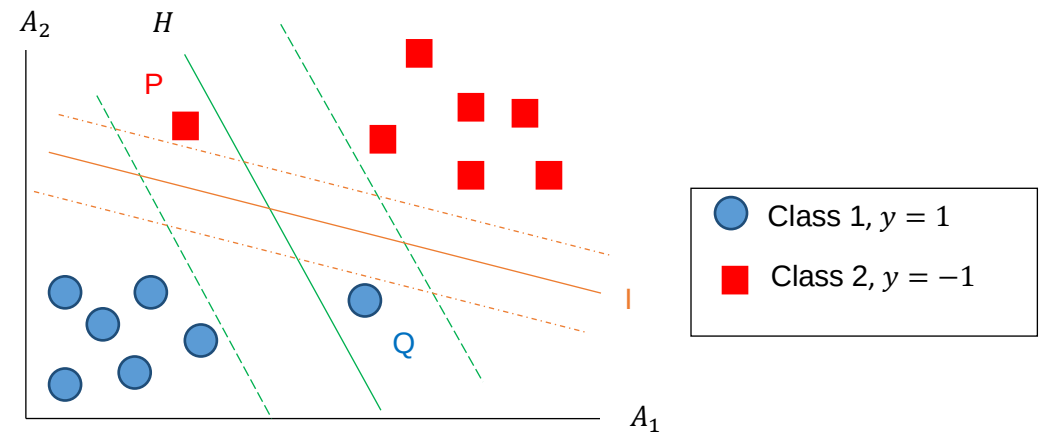


Figure 3.6: Decision boundary and a margin of SVM classifier for inseparable case

In order to do this, a positive-valued slack variable, (ξ_i), is introduced, resulting in the constraints of the optimisation problem in Definition 1 being

$$W \cdot X_i + b \geq 1 - \xi_i \quad \text{if } y_i = 1,$$

$$W \cdot X_i + b \leq -1 + \xi_i \text{ if } y_i = -1,$$

$$\forall i : \xi_i > 0.$$

The objective function must also be modified to penalise a decision boundary with large values of slack variables. The modified objective function is given by

$$f(W) = \frac{\|W\|^2}{2} + C \sum_{i=1}^{d} \xi_i,$$

where C is the parameter representing the penalty for misclassifying training tuples. The choice of the parameter C depends on the model's performance on the validation set. The parameter C is termed the capacity. The primal Lagrangian of the new optimisation problem can be expressed as follows

$$L_p = \frac{\|W\|^2}{2} + C \sum_{i=1}^{d} \xi_i - \sum_{i=1}^{d} \lambda_i [y_i (W \cdot X_i + b) - 1 + \xi_i] - \sum_{i=1}^{d} \mu_i \xi_i \,.$$

The inequality constraints are then transformed to equality constraints using KKT conditions, as follows

$$\xi_i \geq 0, \lambda_i \geq 0, \mu_i \geq 0 \,,$$
$$\lambda_i [y_i (W \cdot X_i + b) - 1 + \xi_i] = 0,$$
$$\mu_i \xi_i = 0.$$

Note that $\lambda_i \neq 0$ if the training tuples lie on the lines $W \cdot X_i + b = \pm 1$ or $\xi_i > 0$. Lagrange multipliers, μ_i are equal to zero for any misclassified training tuples. When the data is linearly separable the slack variables, $\lambda_{i,}$ are equal to zero, however for linearly inseparable data, the slack variables are positive and less than the capacity, that is, $0 \leq \lambda_i \leq C$. The Lagrange multipliers can be obtained by solving the dual problem numerically, using quadratic programming techniques. Consequently, these multipliers can be used to find the parameters W and b of the decision boundary. The SVM formulations explained in this Section and in Section 3.5.2 provide a methodology for constructing linear decision boundaries that classify training tuples into their respective classes. The following Section describes an approach to be adopted when using the SVM method to data sets that have non-linear decision boundary.

3.5.4: THE NON-LINEAR SVM CLASSIFIER

In the case of a non-linear SVM classifier, the decision boundary is non-linear. For this reason, the data is transformed from the original coordinate space X into a transformed space, $\Phi(X)$, such that a linear decision boundary can be used to separate training tuples in the transformed space. The drawback of this approach is the potential to suffer from the curse of dimensionality for problems with high-dimensional data. However, this can be avoided by using a method known as the kernel trick. The linear decision boundary in the transformed space is expressed as

$$W \cdot \Phi(X) + b = 0.$$

Definition 2 (Non-Linear SVM)

The learning task for a non-linear SVM can be formalised as the following optimisation problem: $\min_{W,b} \frac{1}{2}\|W\|^2 + C\sum_{i=1}^{d} \xi_i$

$$\text{subject to } y_i(W \cdot \Phi(X) + b) \geq 1 - \xi_i\,, i = 1,2,\dots,d.$$

The dual Lagrangian of this optimisation problem is given by

$$L_d = \sum_{i=1}^{d} \lambda_i - \frac{1}{2}\sum_{i,j} \lambda_i \lambda_j y_i y_j\, \Phi(X_i) \cdot \Phi\big(X_j\big).$$

The Lagrange multipliers, λ_i are obtained using numerical methods for quadratic programming. The parameters, W and b, of the linear decision boundary in the transformed space are then determined using the following equations

$$W = \sum_{i=1}^{d} \lambda_i y_i \Phi(X_i),$$

$$\lambda_i \left[y_i \left(\sum_{i=1}^{d} \lambda_j y_j \Phi(X_i) \cdot \Phi\big(X_j\big) + b \right) - 1 \right] = 0,$$

where $0 \leq \lambda_i \leq C, \forall i$. Finally, any training, Z, can be classified by using the following equation: $f(Z) = sign(W \cdot \Phi(Z) + b) = sign\big(\sum_{i=1}^{d} \lambda_i y_i \Phi(X_i) \cdot \Phi(Z) + b\big)$.

Learning a non-linear SVM classifier involves calculating the dot product between the pairs of vectors in the transformed space, that is, $\Phi(X_i) \cdot \Phi(X_j)$. This can be computationally costly and the problem can suffer from the curse of dimensionality. To avoid these issues, the kernel trick method is used.

Definition 3 (The Kernel trick) (Steinbach et al, 2006)

The dot product $\Phi(X_i) \cdot \Phi(X_j)$ can be regarded as a measure of similarity between any two tuples, X_i and X_j, in the transformed space. The Kernel trick is the method used to compute similarities in that space using the original attribute set. The similarity function, K is defined as

$$K(X_i, X_j) = \Phi(X_i) \cdot \Phi(X_j).$$

The function, K, is known as the kernel function. A few issues needs to be addressed when using the non-linear SVM method. Firstly, the form of the mapping function, Φ, is not known. Secondly, working in high-dimensional feature space could be very costly. The kernel trick addresses both problems. Firstly, the kernel function is the function of the original attribute set, thus the form of the mapping function, Φ, need not be known. Secondly, since the computations re performed in the original space, issues related to the curse of dimensionality can be avoided. Mercer's Theorem provide conditions under which the kernel function can be expressed as the dot product of any two input vectors. The kernel function used in non-linear SVM must satisfy the Mercer's Theorem.

Mercer's Theorem (Steinbach et al, 2006)

A kernel function K can be expressed as

$$K(u, v) = \Phi(u) \cdot \Phi(v),$$

If, and only if, for any function $g(x)$ such that $\int g(x)^2 dx$ is finite, then

$$\int K(x, y) g(x) g(y) dx dy \geq 0.$$

The radial basis function (rbf) kernel, also known as Gaussian kernel, is a kernel function based on Euclidean distance (Nagi et al, 2008). In this study, Gaussian kernel was used to write the dot product $\Phi(X_i) \cdot \Phi(X_j)$ as function of original data as follows

$$K(X_i, X_j) = \Phi(X_i) \cdot \Phi(X_j) = e^{-\gamma \left\| X_i - X_j \right\|^2}.$$

Note that the Gaussian kernel has only one parameter, namely, γ. This parameter is called the rbf kernel parameter. Another important parameter in the SVM method is the capacity C, which was introduced in Section 3.5.3. The rbf kernel parameter and the capacity are obtained experimentally. Performance measures are used to find the optimal values of, γ, and C, several of which are presented in Section 3.8. The training framework proposed for SVM parameter optimisation is shown in Figure 3.7.

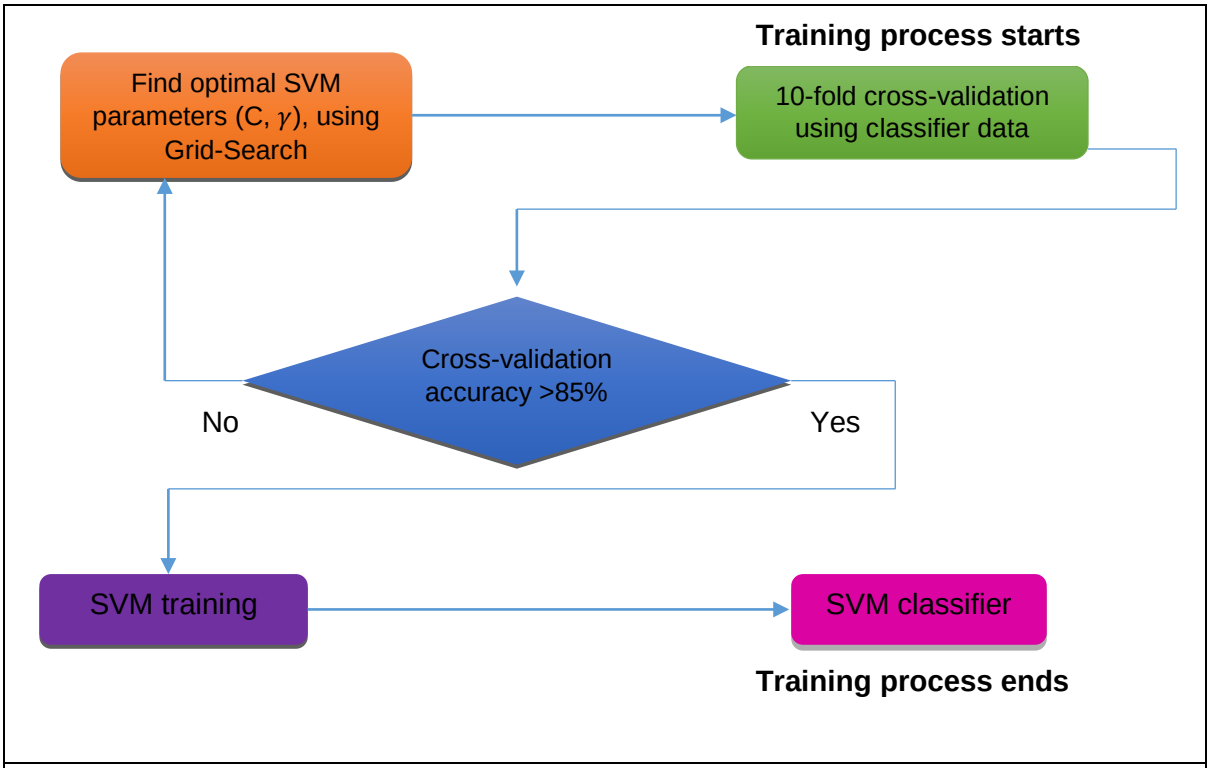


Figure 3.7: Flowchart of training engine (Nagi et al. 2008)

In summary, the SVM is a supervised learning method used for statistical classification of linear and non-linear data. Given a set of training tuples, each belonging to one of two classes, the SVM training algorithm builds a classifier that assign new tuples to one of the two classes. Training is performed by finding the parameters of the decision boundary, which is a hyperplane with largest margin. The decision boundary is then used to classify tuples with unknown class labels.

3.6: NAÏVE BAYES

The NB classifiers are statistical classifiers that predict class membership with associated probabilities. That is, in addition to classifying a given tuple, a Bayes classifier also assigns a probability that a given tuple belongs to a particular class. Bayes classification is based on Bayes Theorem. Bayes classifiers exhibit high accuracy and speed when applied to high dimensional data (Han & Kamber, 2006).

Naïve Bayes classifiers assume that the effect of an attribute value on a given class is independent of the values of the other attributes. This assumption is called class conditional independence (Han & Kamber, 2006). The class conditional independence is made to simplify computations and for this reason is said to be naïve. Bayesian belief networks are graphical models that account for class conditional dependence. For the purpose of this study, only NB classifiers will be discussed and subsequently used for classification.

3.6.1: BAYES THEOREM

Let X be a given data tuple, characterised by measurements made on a set of n attributes, $A_1, A_2, \dots, A_n$, such that, $X = (x_1, x_2, \dots, x_n)$. In Bayesian Theory, X is considered the evidence. Define, H as the hypothesis that the data tuple, X, belongs to a particular class, C. In the NB classification, the objective is to compute the probability that the hypothesis, H, is true, given observed tuple, X, that is $P(H|X)$. This probability, $P(H|X)$, is called the posterior probability of H, conditioned on X. The probability that the hypothesis is true, $P(H)$, is called the prior probability of H. $P(X)$ is the prior probability of X. The posterior probability of H conditioned on X, can be computed using the Bayes Theorem.

Bayes Theorem (Han & Kamber, 2006)

$$P(H|X) = \frac{P(X|H)P(H)}{P(X)}.$$

where the probabilities, $P(H), P(X|H), P(X)$ are estimated from the given data. The following Section presents a methodology for using Bayes Theorem to build a NB classifier.

3.6.2: NAÏVE BAYES CLASSIFICATION

The NB classification works as follows (Han & Kamber, 2006):

1. Let D be a given training data set with associated class labels, where each training tuple is described by a vector $X = (x_1, x_2, \dots, x_n)$.
2. Suppose that there are t classes, $C_1, C_2, \dots, C_t$. Given a tuple, X, the NB classifier will predict that X belongs to the class having the highest posterior probability, conditioned on X. That is, the NB classifier predicts that any tuple X belongs to the class C_i if and only if

   $$P(C_i|X) > P(C_j|X) \text{ for } 1 \leq j \leq t, j \neq i.$$

   Thus $P(C_i|X)$ is maximized, the class C_i for which $P(C_i|X)$ is maximised is called maximum posteriori hypothesis. The posterior probability, $P(C_i|X)$ is computed using Bayes theorem as

   $$P(C_i|X) = \frac{P(X|C_i)P(C_i)}{P(X)}.$$

3. Since $P(X)$ is the same for all classes, only $P(X|C_i)P(C_i)$ need to be maximised. The class prior probabilities can be estimated by

   $$P(C_i) = \frac{|C_{i,D}|}{d},$$

   where $|C_{i,D}|$ is the number of training tuples of a class i in D, and d is the total number of training tuples in D.
4. The values of the attributes are assumed to be conditionally independent of one another given the class label of the tuple, therefore for $i = 1,2, \dots, t$,

   $$P(X|C_i) = \prod_{k=1}^{n} P(x_k|C_i)$$

   $$= P(x_1|C_i) \times P(x_2|C_i) \times \dots \times P(x_n|C_i).$$

   The probabilities $P(x_1|C_i)$, $P(x_2|C_i), \dots, P(x_n|C_i)$ are estimated from the training tuples. The continuous valued attributes are assumed to follow a Normal distribution with mean, μ and standard deviation, σ defined by the density function

   $$g(x, \mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{(x-\mu)^2}{2\sigma^2}},$$

   such that $P(x_k|C_i) = g(x_k, \mu_{C_i}, \sigma_{C_i})$, where μ_{C_i} and σ_{C_i} are mean and standard deviation of the values of A_k for training tuples of class C_i.
5. To predict the class label of X, $P(X|C_i)P(C_i)$ is computed for each class C_i. The NB classifier then predicts that the class label of tuple X is C_i if and only if

   $$P(X|C_i)P(C_i) > P(X|C_j)P(C_j) \text{ for } 1 \leq j \leq t, j \neq i.$$

Despite the fact that the independence assumptions are often inaccurate, the NB classifier has properties that make it remarkably useful in practice. In particular, the separation of the class conditional feature distributions means that each distribution can be independently estimated as one-dimensional distribution. This avoids problems stemming from the curse of dimensionality. Another advantage of the NB classifier is its ability to explicitly associate probabilities to each predicted tuple.

3.7: K-NEAREST NEIGHBOUR METHOD

The k-nearest neighbour is a non-parametric method that is used for the purpose of statistical classification. This method, first introduced in the early 1950's was computationally expensive when given high dimensional data. As a result, KNN only gained popularity in the 1960's when improved computational power was available (Han & Kamber, 2006).

3.7.1: K-NEAREST NEIGHBOUR CLASSIFICATION

The KNN classifiers are said to learn by analogy as they compare a given test tuple with k training tuples that are similar to it. The training and test tuples are described by n −dimensional vectors in a n −dimensional pattern space. Given a test tuple with unknown class label, the KNN classifier will search for k training tuples in the pattern space that are closest to the given test tuple. The "closeness" is defined in terms of distance metrics, such as Euclidean distance and City Block distance. The k training tuples are called the k nearest neighbours, hence the name k-Nearest Neighbour. The test tuple is then given the label that is most common among its k nearest neighbours. The Euclidean and the City Block distance metrics are defined as follows

The **Euclidean distance** between any two tuples, $X_1 = (x_{11}, x_{12}, \dots, x_{1n})$ and $X_2 = (x_{21}, x_{22}, \dots, x_{2n})$ is defined as

$$d(X_1, X_2) = \sqrt{\sum_{i=1}^{n} (x_{1i} - x_{2i})^2},$$

and the **City Block distance** (also known as the **Manhattan distance**) between two tuples, $X_1 = (x_{11}, x_{12}, \dots, x_{1n})$ and $X_2 = (x_{21}, x_{22}, \dots, x_{2n})$ is defined as

$$d_1(X_1, X_2) = \sum_{i=1}^{n} |x_{1i} - x_{2i}|.$$

The determination of the value of k and choosing a distance metric are done experimentally. The value of k heavily depends on the empirical data. Performance measures such as sensitivity, error rate or accuracy are used to determine the best value of k (Han & Kamber, 2006). The sensitivity and other performance measures are formally defined in Section 3.8. In this study, sensitivity was chosen as a primary measure of accuracy and was used to find the optimal value of k.

In this study, the procedure followed in finding the value of k is described as follows

- Sensitivity was computed for each value of k from 1 to 100.
- For each value of k, 500 different training and test sets combinations were used to compute sensitivity estimates for that particular value of k.
- The average sensitivity was then computed for each value of k.
- The averages of sensitivity estimates were then plotted against the values of k.
- The value of k that maximises the sensitivity was chosen to train the final KNN classifier.

The procedure described above was repeated twice, first, using the Euclidean distance and secondly, the City Block distance. The distance metric that produced highest sensitivity estimates was then used to train the final KNN classifier.

3.8: CLASSIFIER PERFORMANCE MEASURES

After a classifier has been trained, the following questions arise:

1. How accurately can the classifier predict class labels of previous unseen data?
2. How can performance of different trained classifiers be compared?
3. What is performance?
4. How can performance be estimated?

The aim of this section is to address the above questions. Section 3.8.1 presents measures to determine performance of a classifier. These measures can be implemented using the holdout method, which is discussed in Section 3.9.

3.8.1: PERFORMANCE MEASURES OF A CLASSIFIER

Using a training data set to build a classifier, as well as using the same training set to estimate performance, can pose the danger of model over-fitting. In fact, if a training set is used to estimate the performance, then the estimates will be optimistic of the true

performance. For this reason, a test set, with class labelled tuples that were not used to train the classifier, s used to estimate the performance of a classifier.

The following are the definitions of various performance measures for a binary classification problem

Table 3.6: Definitions of several performance measures (Han & Kamber, 2006)

Performance measure	**Definition**
Accuracy	The proportion of test tuples that are correctly classified by the classifier.
Misclassification rate /Error rate	The proportion of test tuples that are incorrectly classified by the classifier.
Positive tuples	The tuple of main class of interest (e.g. Fraudulent customers).
Negative tuples	The tuple of the other class (e.g. Clean customers).
True positives	Positive tuples that are correctly classified by the classifier.
True negatives	Negative tuples that are correctly classified by the classifier.
False positive	Negative tuple that are incorrectly classified as positive tuples.
False negative	Positive tuples that are incorrectly classified as negative tuples.
Sensitivity	Proportion of positive tuples that are correctly classified.
Specificity	Proportion of negative tuples that are correctly classified.

The confusion matrix is an important tool used to analyse how well a classifier classifies tuples of different classes. A confusion matrix for two classes, C_1 and C_2 is shown in Table 3.7.

Table 3.7: A confusion matrix for positive and negative tuples

	$\boldsymbol{C_1}$	$\boldsymbol{C_2}$
$\boldsymbol{C_1}$	true positives	false negatives
$\boldsymbol{C_2}$	false positives	true negatives

Sensitivity, specificity and accuracy are defined algebraically as follows (Han & Kamber, 2006)

$$sensitivity = \frac{t\_pos}{pos},$$

$$specificity = \frac{t\_neg}{neg},$$

$$accuracy = sensitivity\left(\frac{pos}{pos + neg}\right) + specificity\left(\frac{neg}{pos + neg}\right).$$

Where; t_pos is the number of true positive tuples correctly classified, pos is the total number of positive tuples, t_neg the number of true negative tuples correctly classified, and, neg the total number of negative tuples.

3.9: THE HOLDOUT METHOD

The Holdout method was used to calculate performance measures from a given empirical data. This method was also useful to compare different trained classifiers. In this method, a given data set was randomly partitioned into a training set and a test set. It is a convention to allocate two-thirds of the data to the training set and the remaining one-third to the test set (Kohavi, 1995).The training set was used to build a classifier and the test set to estimate the performance of the learned classifier. This method is illustrated graphically in Figure 3.8. The random subsampling method is a variation of the holdout method, in which the holdout method is repeated p times (Han & Kamber, 2006). The overall accuracy was then defined as the average of accuracy estimates obtained from each iteration. Sensitivity, specificity and error rate estimates were defined in a similar manner under the random subsampling method. The advantage of using the random subsampling method is that several estimates can be calculated for each performance measure. This enabled clarity on how the estimates varied from sample to sample.

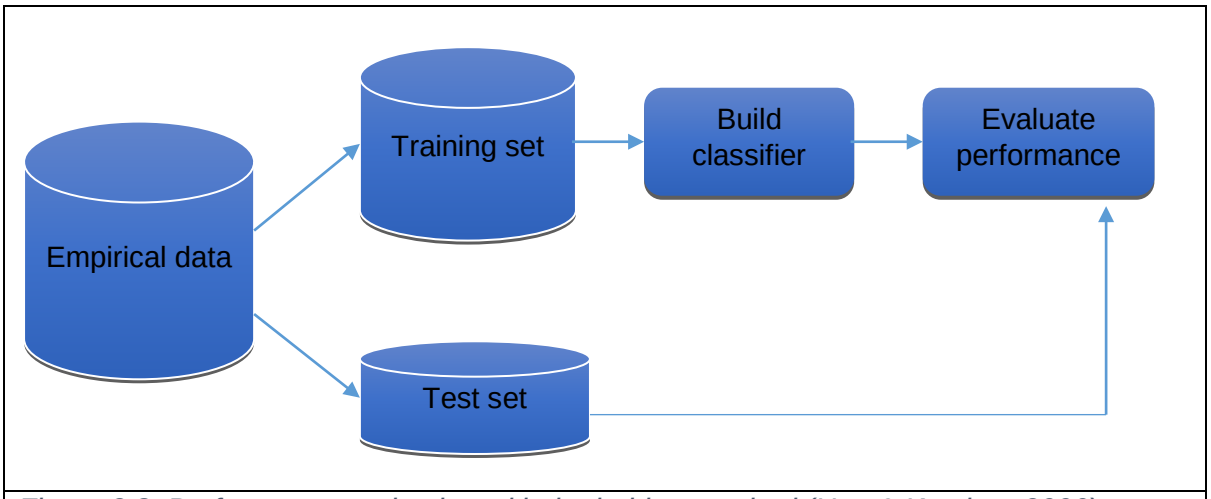


Figure 3.8: Performance evaluation with the holdout method (Han & Kamber, 2006)

3.10: FRAMEWORK FOR CUSTOMER CLASSIFICATION

In this Section, the framework of the approach followed to design and to develop the classification model is presented. Depicted in Figure 3.9 is the general framework for the approach adopted to propose the fraud detection model. The Holdout method was repeated 500 times and for each case, sensitivity, specificity and error rates determined for the SVM, NB and KNN classifiers. Sensitivity, specificity and error rate were then used to compare the performance of these classifiers.

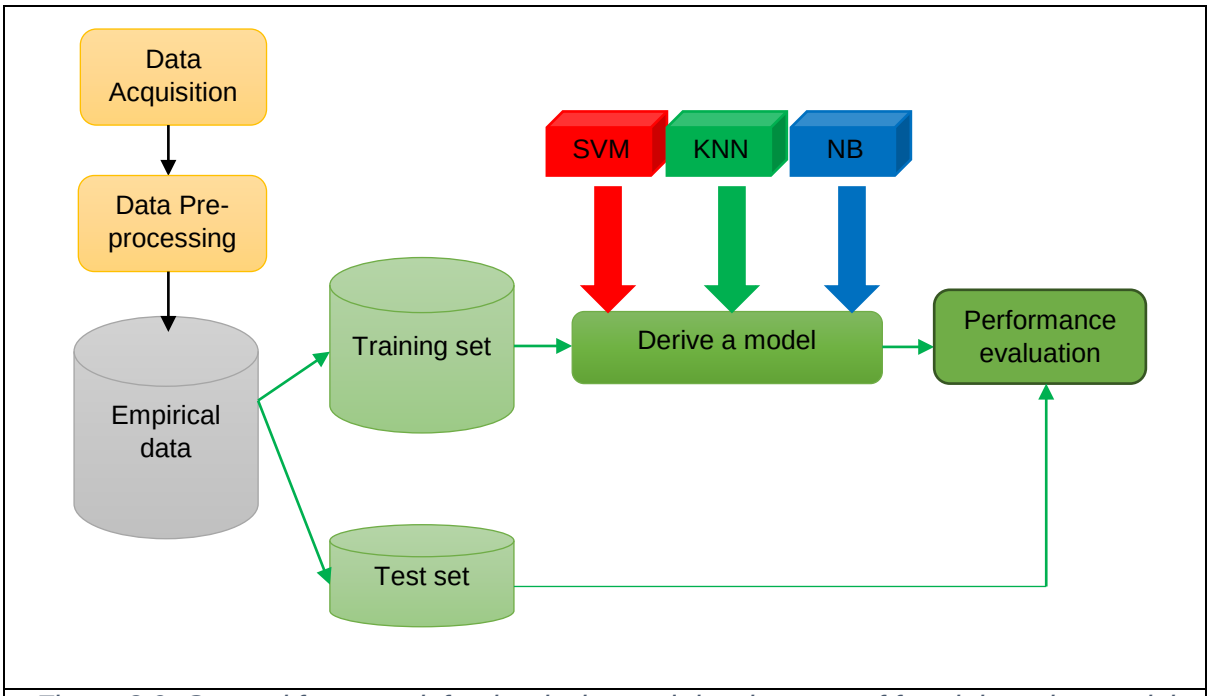


Figure 3.9: General framework for the design and development of fraud detection model

3.11: CHAPTER SUMMARY

In this chapter, the methodology followed to derive the fraud detection model was presented. The method involved using three classification algorithms to derive three classifiers, namely, the SVM, NB and KNN classifiers. These were compared and the classifier with highest fraud detection ability chosen. The ability to detect electricity fraud was determined using the sensitivity, specificity and error rate estimates. The random subsampling method was used to build and test the said classifiers. The results are presented in the following Chapter.

CHAPTER 4: RESULTS AND DISCUSSION

4.1: INTRODUCTION

The aim of this study was to design a classifier that could be used as a model to identify and detect electricity fraud in NMBM. Three classification algorithms, Support Vector Machines (SVM), Naïve Bayes (NB) and k-Nearest Neighbour (KNN) were used to train three classifiers. Matlab R2012a (7.14.0.739) 32 bit (win32), was used for the purpose of training and testing the classifiers. The performance of these classifiers was then evaluated and compared, the results of which are presented this chapter. Firstly, the results with respect to the SVM classifier are discussed, followed by those of the NB and KNN classifiers. Finally, the results of all three classifiers are compared in Section 4.5.

4.2: SUPPORT VECTOR MACHINES RESULTS

The holdout method was used to train and test the SVM classifier using the empirical data in Table 3.3. The dataset consisted of 3 156 customers, 2420 of which were classified as fraudulent, and 736 as clean, cases. This data set was randomly partitioned into a training and a test set. Roughly two thirds of the data tuples formed a training set and the remaining one third a test set. The training set consisted of 2 102 customers, of which approximately 78% were clean, and 22% fraudulent, cases. By comparison, the test set comprised of 1 052 customers, of which 75% were clean and 25% were fraudulent. The number of customers allocated to training and test sets is shown in Table 4.1.

Table 4.1: The summary of training and test sets for the SVM classifier

	Clean	**Fraudulent**	**Total**
Training set	1 632	470	2 102
Test set	788	264	1 052
Total	2 420	734	3 154

The SVM classifier was trained on the training set and the performance thereof estimated using the test set. The test set was then used to calculate the values in the confusion matrix. The corresponding confusion matrix, illustrating the number of correct and incorrect classifications, is shown in Table 4.2. The columns correspond to the true labels, and the rows to the class labels predicted by the SVM classifier. The numbers of correct classifications are the values in the main diagonal of the confusion matrix, and misclassifications, the off diagonal values. Ideally, all the values should lie in the main

diagonal and all the off diagonal values should be zero. This would mean that the classifier correctly classifies all the test tuples. However in practice, the off diagonal values are hardly zeros. Thus, a good classifier is characterised by having off diagonal values that are very small in comparison to the values in the main diagonal.

Table 4.2: Confusion matrix for the SVM classifier

Predicted labels		**True labels**		
		Fraudulent	Clean	Total
	Fraudulent	209	104	313
	Clean	55	684	739
	Total	264	788	1052

From the above confusion matrix, sensitivity, specificity, error rate and accuracy estimates were calculated as follows

$$sensitivity = \frac{209}{264} \times 100\% = 79.17\%,$$

$$specificity = \frac{684}{788} \times 100\% = 86.80\%,$$

$$error\ rate = \frac{55 + 104}{1052} \times 100\% = 15.11\%,$$

The sensitivity represented the percentage of fraudulent cases correctly identified by the classifier. This was an important measure, given the aim of this study, and a desirable estimate of this measure would be at least 80%. The number of clean cases was always very large when compared to the number of fraudulent ones, and as such, the specificity estimate was always be greater than the sensitivity. The high specificity rate implied that there would be very small number of clean cases identified as fraudulent. The error rate was another imperative measure to consider, as it measured the percentage of misclassified customers.

According to the confusion matrix in Table 4.2, the SVM classifier produced a sensitivity which was just above to 79%. The specificity estimate was well above 86% and the error rate estimated to be close to 15%. Although all three measures produced satisfactory results, the sensitivity estimate was still preferred to be at least 80%.

The measures calculated from Table 4.2 were attained using one training set and one test set. Would the results be the same if different training and test sets were used? To address this question, the random subsampling method was used. In this method, the holdout method was repeated 500 times. That is, 500 combinations of training and test sets were used to train and test the SVM classifier respectively. Sensitivity, specificity and error rate estimates were calculated for each iteration. The results for the 500 iterations, with associated averages and standard deviations for each measure, are depicted in Table 4.3. These measures were then plotted on one set of axes against the number of the iteration. The resulting line graphs are shown in Figure 4.1. All of the estimates were relatively constant over different test sets. The sensitivity estimate was around 80% for all the test sets, while the specificity estimate was constant, at approximately 88%, and the error rate close to 14% for all the sets. The standard deviation, for all the estimates, was less than 3%.

Table 4.3: Performance measure estimates for SVM classifier before parameter optimisation

Number of the iteration	Sensitivity	Specificity	Error rate
1	0.7556	0.9021	0.1293
2	0.7826	0.8936	0.1331
⋮	⋮	⋮	⋮
500	0.7917	0.8680	0.1511
Average:	0.7954	0.8812	0.1389
Standard deviation:	0.0254	0.0114	0.0093

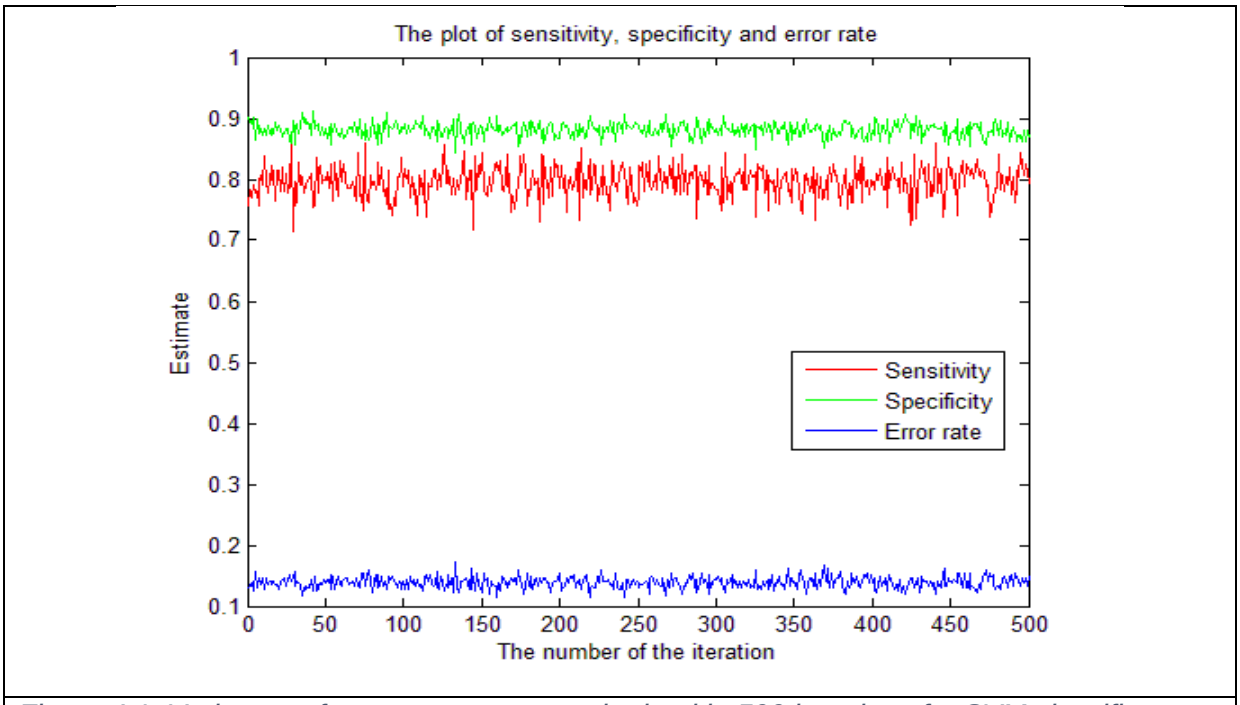


Figure 4.1: Various performance measures obtained in 500 iterations for SVM classifier

4.2.1: PARAMETER OPTIMISATION

Experimentally it was found that the optimal values of the capacity and the radial basis function (rbf) parameter were, $C = 2.1639$ and $\gamma = 1.3542$ respectively. The SVM classifier was then trained and tested using the optimal parameters. The confusion matrix calculated using the optimal parameters is shown in Table 4.4. This confusion matrix was then used to calculate the sensitivity, specificity and error rate estimates.

Table 4.4: SVM confusion matrix with optimum parameters

		True labels		
		Fraudulent	*Clean*	*Total*
Predicted labels	*Fraudulent*	199	102	301
	Clean	42	709	751
	Total	241	811	1052

$$Sensitivity = \frac{199}{241} \times 100\% = 82.57\%,$$

$$Specificity = \frac{709}{811} \times 100\% = 87.42\%,$$

$$Error\ rate = \frac{42 + 102}{1052} \times 100\% = 13.69\%.$$

Even though the specificity and the error rate estimates did not change significantly after parameter optimisation, the sensitivity estimate improved from 80% to 82.5%. Again, to test whether this change was significant or due to chance, the random subsampling method was used with the optimal parameters to obtain 500 different estimates for each performance measure. These estimates are shown in Table 4.5 and corresponding line graphs depicted in Figure 4.2.

Table 4.5: Performance measure estimates for SVM classifier after parameter optimisation

Number of the iteration	**Sensitivity**	**Specificity**	**Error rate**
1	0.8434	0.8618	0.1426
⋮	⋮	⋮	⋮
500	0.8257	0.8742	0.1369
Average:	0.8249	0.8663	0.1434
Standard deviation:	0.0223	0.0113	0.0087

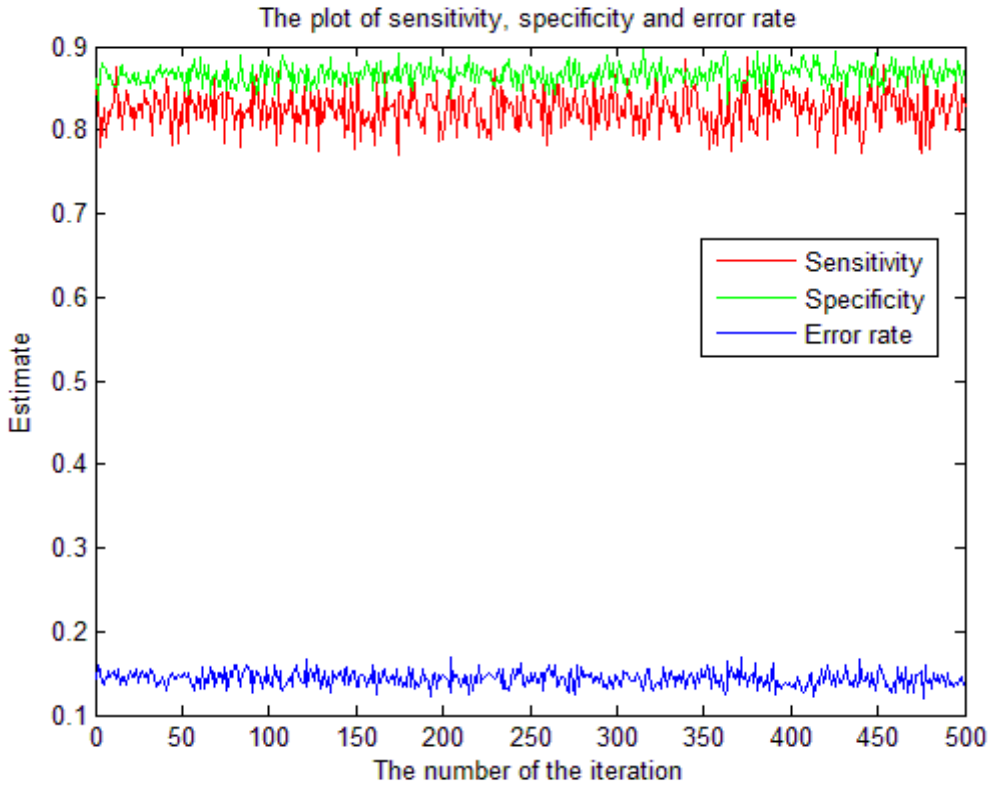


Figure 4.2: SVM performance measures with optimised parameters

The random subsampling method was useful to determine how each performance measure changed over different test sets. In this Section, this method was used extensively to evaluate the performance of the SVM classifier. The SVM classifier produced satisfactory results to predict fraudulent usage of electricity using kWh consumption, with average sensitivity of approximately 83%, specificity of roughly 87% and error rate close to 14%. In the following Sections, the random subsampling was also used to evaluate the performance of the NB and the KNN classifiers. This method was also used to compare the performance of the three classifiers of interest in Section 4.5.

4.3: NAÏVE BAYES CLASSIFICATION RESULTS

The NB classifier was built using the holdout method and the given empirical data. The training and test sets are summarised in Table 4.6. Depicted in Table 4.7 is a confusion matrix corresponding to the test set.

Table 4.6: The summary of training and test set for NB classifier

	Clean	**Fraudulent**	**Total**
Training set	1622	480	2102
Test set	798	254	1052
Total	2420	734	3154

Table 4.7: NB confusion matrix on a particular test set

Predicted labels		**True labels**		
		Fraudulent	*Clean*	*Total*
	Fraudulent	179	120	299
	Clean	75	678	753
	Total	254	798	1052

The following are estimates calculated using the confusion matrix in Table 4.7.

$$sensitivity = \frac{179}{254} \times 100\% = 70.47\%,$$

$$specificity = \frac{678}{798} \times 100\% = 84.96\%,$$

$$error\ rate = \frac{75 + 120}{1052} \times 100\% = 18.54\%,$$

The specificity estimate was approximately 85%, and the sensitivity close to 70%, with an error rate of less than 20%. Therefore, the NB classifier predicted clean cases with a high probability, almost 85%, whereas the probability of detecting fraudulent cases was lower, approximately 70%.

To effectively evaluate the performance of the NB classifier, different samples of training and test sets were used to calculate the sensitivity, specificity and error rate estimates. The random subsampling method was used to provide 500 partitions of training and test sets. For each partition of training and test set, the sensitivity, specificity and error rates estimates were calculated, the results of which are shown in Table 4.8.

Table 4.8: The NB accuracy estimates for 500 combinations of training and test sets

Number of the iteration	Sensitivity	Specificity	Error rate
1	0.7200	0.8354	0.1920
2	0.7229	0.8431	0.1854
3	0.6920	0.8466	0.1901
⋮	⋮	⋮	⋮
500	0.7047	0.8496	0.1854
Average:	0.7331	0.8472	0.1795
Standard deviation:	0.0254	0.0120	0.0102

The results achieved from the 500 samples were consistent with those from the NB confusion matrix in Table 4.7. That is, the average sensitivity estimate was just above73%, with average specificity of around 85% with an average error rate of less than 20%. All the estimates had small sample variation, with standard deviations less than 3%. These estimates are graphically plotted in Figure 4.3.

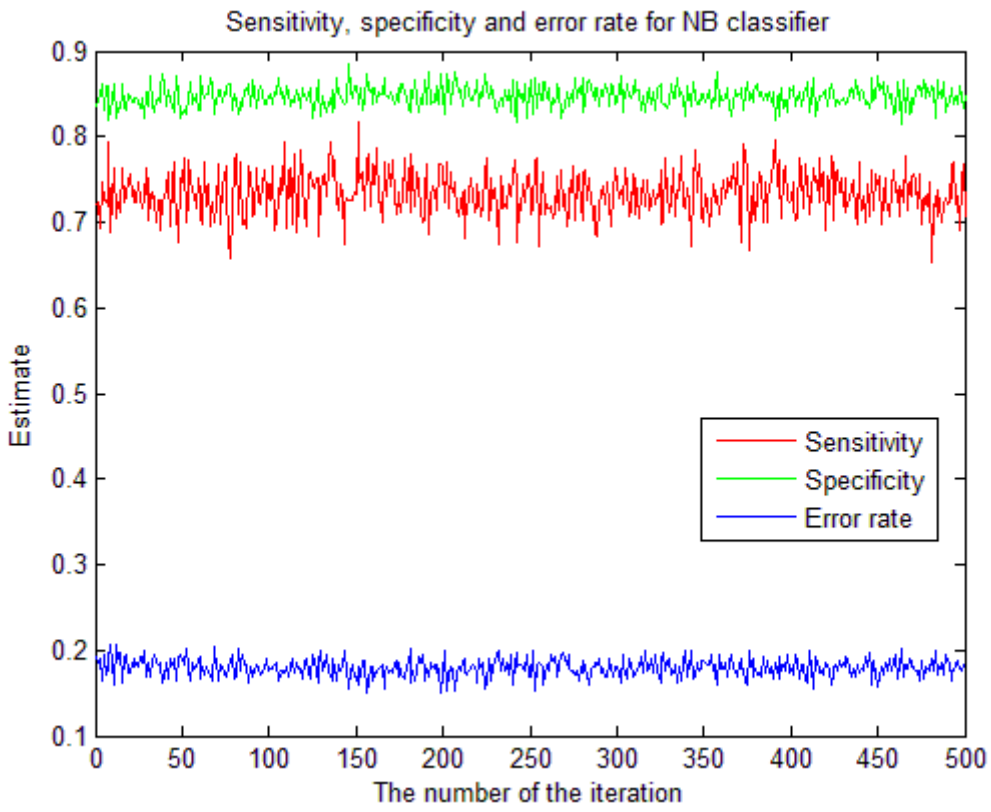


Figure 4.3: Performance measure for NB classifier

4.4: K-NEAREST NEIGHBOUR

In the KNN method, a test tuple was given the class label most common amongst k training tuples closest to it. The similarity was measured in terms of distance metrics. Before the KNN classifier was trained, two important questions needed consideration. Firstly, what value of k should be used? Secondly, which distance metric should be applied? In this Section, the methodology followed in finding the optimum value of k and choosing the best distance metric is presented and the subsequent results of KNN classifier discussed.

4.4.1: FINDING THE OPTIMUM VALUE OF K

The optimum value of k was chosen from the sequence of values that grow linearly from 1 to 100, that is, $(1,2,3,\ldots.,100)$. For each value of k, the random subsampling method was used to calculate 500 estimates for sensitivity. This analytical routine was repeated twice, firstly using the Euclidean distance and then using the City Block distance. To minimise computational time, only sensitivity, which was equivalent to fraud detecting rate, was used as the performance measure. The results are presented in Table 4.9 and Table 4.10 respectively for the Euclidean and the City Block distance metrics.

Table 4.9: KNN sensitivity estimates for different values of k using Euclidean distance.

Iteration number	**k=1**	**k=2**	**k=3**	**...**	**k=10**	**…**	**k=99**	**k=100**
1	0.5691	0.5397	0.6595	...	0.6420	...	0.5138	0.5451
2	0.5858	0.5950	0.6781	...	0.6582	...	0.5323	0.5310
3	0.5709	0.6409	0.6563	...	0.6423	...	0.5496	0.5403
4	0.5991	0.6375	0.6429	...	0.7021	...	0.5792	0.5214
5	0.5809	0.6275	0.6280	...	0.6574	...	0.5000	0.5583
6	0.5837	0.5737	0.6111	...	0.6513	...	0.5519	0.5267
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
500	0.6513	0.5830	0.6154	...	0.7083	...	0.5394	0.5385
Average:	0.6039	0.6056	0.6267	...	0.6460	...	0.5337	0.5373
Standard deviation:	0.0318	0.0326	0.0305	...	0.0289	...	0.0267	0.0279

Table 4.10: KNN sensitivity estimates for different values of k using City Block distance

Iteration number	k=1	k=2	k=3	...	k=10	...	k=99	k=100
1	0.7386	0.7000	0.7195	...	0.7778	...	0.6278	0.6667
2	0.7149	0.6695	0.7352	...	0.7336	...	0.6545	0.7126
3	0.6504	0.6496	0.7561	...	0.8049	...	0.6967	0.6923
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
500	0.6778	0.6417	0.7239	...	0.7804	...	0.5394	0.5385
Average:	0.6820	0.6791	0.7167	...	0.7507	...	0.6852	0.6887
Standard deviation:	0.0306	0.0309	0.0279	...	0.0263	...	0.0266	0.0256

The average sensitivity estimates were calculated for each value of k and for each distance metric. The line graphs of average sensitivity versus the value of k is shown in Figure 4.4. It was evident that the City Block distance produced higher average sensitivity estimates when compared to the Euclidean distance. Thus, the City Block distance was chosen for the subsequent training and testing of the KNN classifier. Although the City Block distance yielded higher average sensitivity estimates, the values of k which yielded maximum average sensitivity estimates, ranging from 10 to 30, were not unique. The average of these values was 20, but as that is an even number, the optimum value of k was chosen to be 21. The odd number was chosen to avoid ties between the classes of k nearest neighbours.

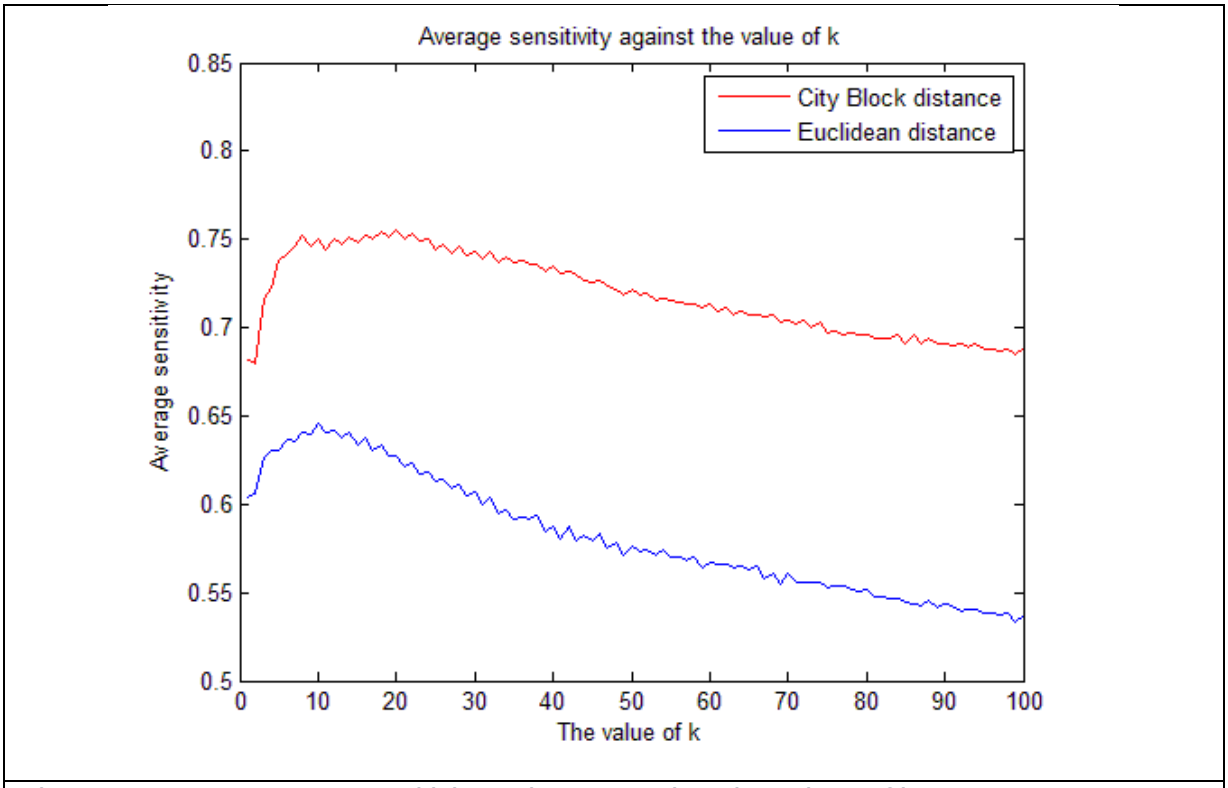


Figure 4.4: KNN average sensitivity estimates against the values of k

Using the City Block distance and k equal to 21, the KNN classifier was then trained and tested using the holdout method on the given empirical data. A confusion matrix for the KNN classifier is depicted in Table 4.11.

Table 4.11: KNN confusion matrix for a particular test set

Predicted labels		**True labels**		
		Fraudulent	*Clean*	*Total*
	Fraudulent	185	88	273
	Clean	61	718	779
	Total	246	806	1052

The estimates for performance measures, obtained using the KNN confusion matrix are as follows

$$sensitivity = \frac{185}{246} \times 100\% = 75.20\%,$$

$$specificity = \frac{718}{806} \times 100\% = 89.08\%,$$

$$error\ rate = \frac{61 + 88}{1052} \times 100\% = 14.16\%,$$

The accuracy estimates varied from sample to sample. To investigate this variation, the random subsampling method was used to calculate sensitivity, specificity and error rate estimates from 500 different samples of training and test set. For each iteration, a training set was used to train the KNN classifier, then a test set applied to calculate sensitivity, specificity and error rate estimates. The evaluation results are depicted in Table 4.12 and in Figure 4.5.

Table 4.12: KNN accuracy estimates for 500 iterations

Number of the iteration	**Sensitivity**	**Specificity**	**Error rate**
1	0.7610	0.8964	0.1359
2	0.7259	0.9066	0.1397
3	0.7609	0.8905	0.1378
⋮	⋮	⋮	⋮
500	0.7520	0.8908	0.1416
Average:	0.7503	0.8980	0.1365
Standard deviation:	0.0246	0.0105	0.0092

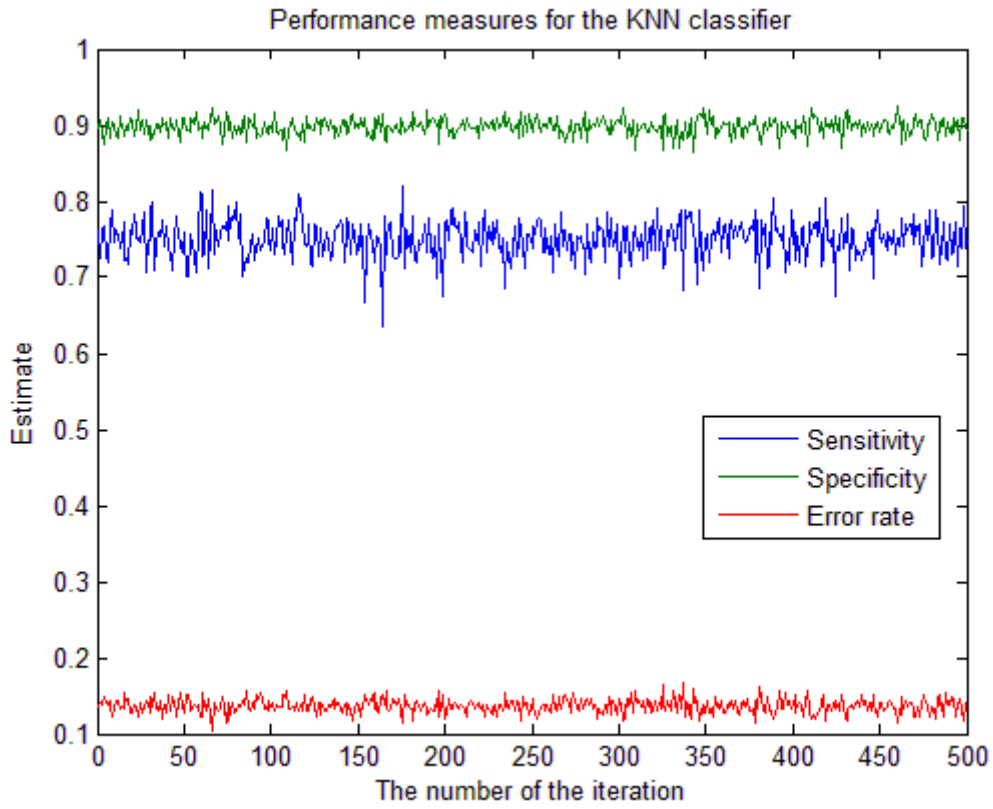


Figure 4.5: Performance measures for the KNN classifier

The results from the confusion matrix were consistent with the averages calculated in Table 4.12. The KNN classifier achieved average sensitivity of approximately 75%. The probability of correctly predicting clean cases was very high, almost 90%. The error rate was less than 15%.

4.5: COMPARISON OF THE THREE CLASSIFIERS

In the previous Sections, individual performances of the SVM, NB and KNN classifiers were studied independently. In this Section, the performance of these classifiers were investigated collectively. That is, using the same training set, all three classifiers were trained and then tested on one test set. The random subsampling method was then used to compute 500 estimates for each performance measure, namely, sensitivity, specificity and error rate. Finally, the performance of these classifiers was compared.

Sensitivity estimates for the three classifiers of interest are shown in Table 4.13 and the corresponding line graphs plotted in Figure 4.6. The classifier with the highest sensitivity was the SVM, with an average rate of 83%. The KNN and the NB classifiers had approximately the same ability to detect fraudulent customers, with sensitivity rates of roughly 75% and 73%, respectively.

Table 4.13: Sensitivity estimates comparison between the three classifiers

Number of the iteration	SVM	KNN	NB
1	0.8536	0.7657	0.7113
2	0.7759	0.6940	0.6897
3	0.8268	0.7532	0.7532
4	0.8093	0.7821	0.7743
5	0.8511	0.7824	0.7443
6	0.8452	0.7381	0.7421
7	0.8392	0.7490	0.7647
8	0.8306	0.7379	0.7379
⋮	⋮	⋮	⋮
500	0.8145	0.7783	0.7421
Average:	0.8251	0.7500	0.7340
Standard deviation:	0.024	0.028	0.027

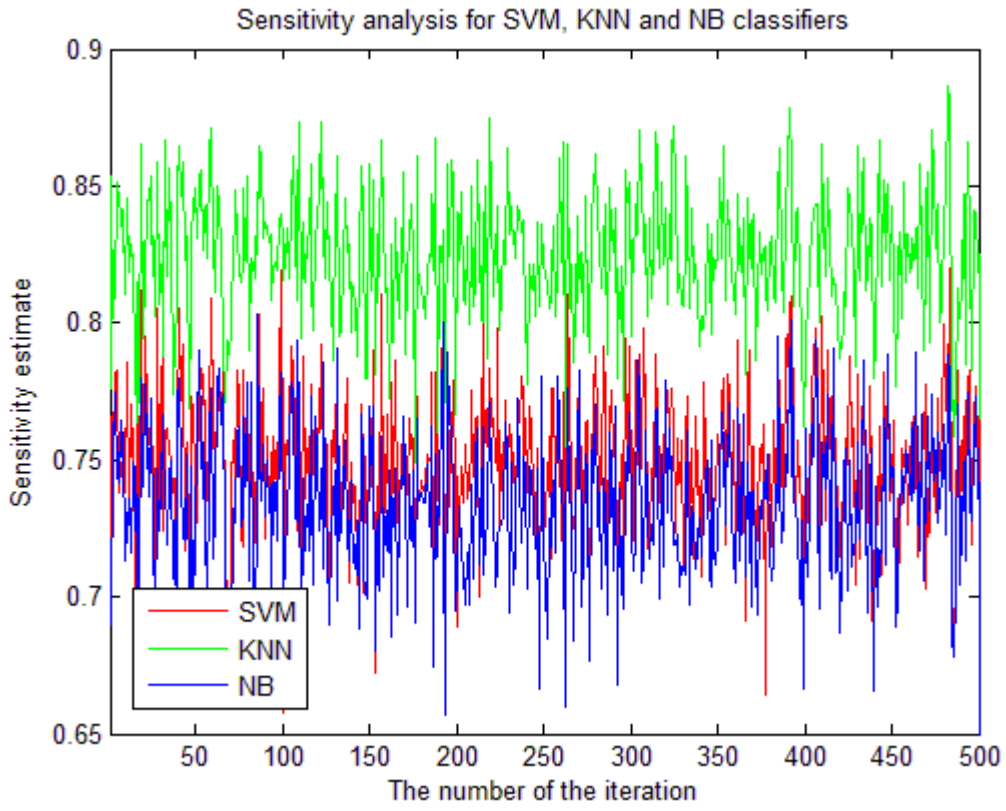


Figure 4.6: Graphical representation of sensitivity estimate for the three classifiers

Depicted in Table 4.14 are various specificity rate estimates and their corresponding line graphs are illustrated in Figure 4.7. The KNN classifier proved to be more accurate than its competitors in predicting clean cases, with an average specificity close to 90%. The NB classifier performed the worst in terms of specificity, at roughly 85%, compared to the SVM classifier, which fared slightly better, with a probability of 87%.

Table 4.14: Specificity estimates comparison between the three classifiers

Number of the iteration	SVM	KNN	NB
1	0.8733	0.9016	0.8352
2	0.8817	0.9085	0.8744
3	0.8490	0.8794	0.8173
⋮	⋮	⋮	⋮
500	0.8700	0.8929	0.8315
Average:	0.8661	0.8976	0.8461
Standard deviation:	0.0121	0.0108	0.0131

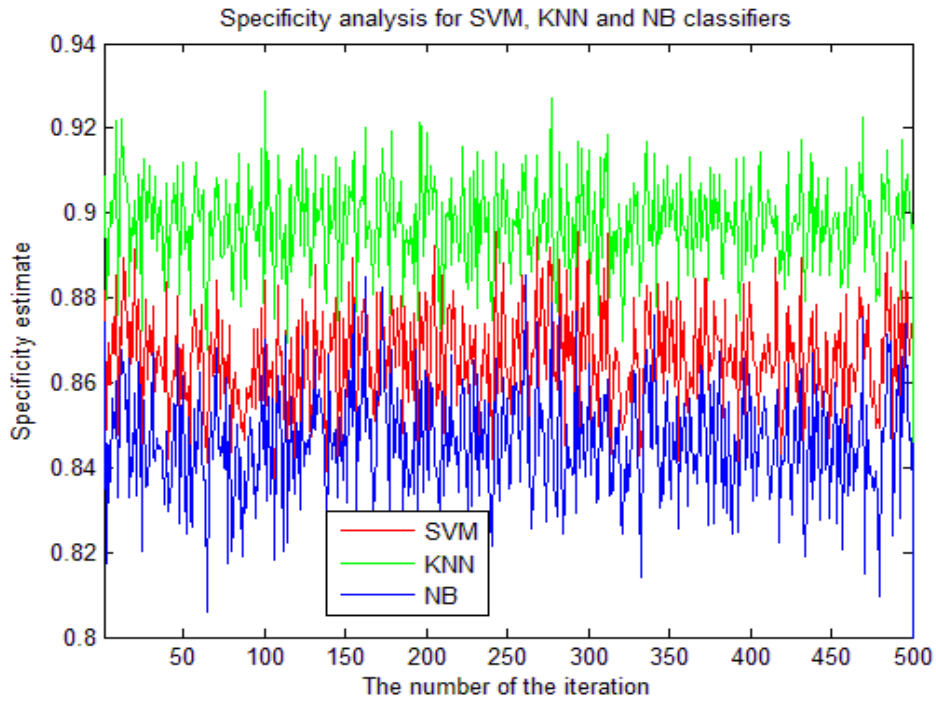


Figure 4.7: Graphical representation of specificity estimates for the three classifiers

All learned classifiers need to have small error rates in order to guarantee that the lowest proportion of fraudulent cases are classified as clean, and vice versa. A good classifier is characterised by a small error rate. The SVM and KNN classifiers had jointly the smallest error rates, of roughly 14%, in comparison to the NB classifier, at 18%. Various error rate estimates are depicted in Table 4.15 and line graphs thereof shown in Figure 4.8 for the three classifiers.

Table 4.15: Error rate estimates comparison between the three classifiers

Number of the iteration	SVM	KNN	NB
1	0.1312	0.1293	0.1930
2	0.1416	0.1388	0.1663
3	0.1559	0.1483	0.1968
⋮	⋮	⋮	⋮
500	0.1416	0.1312	0.1873
Average:	0.1435	0.1369	0.1801
Standard deviation:	0.0096	0.0092	0.0105

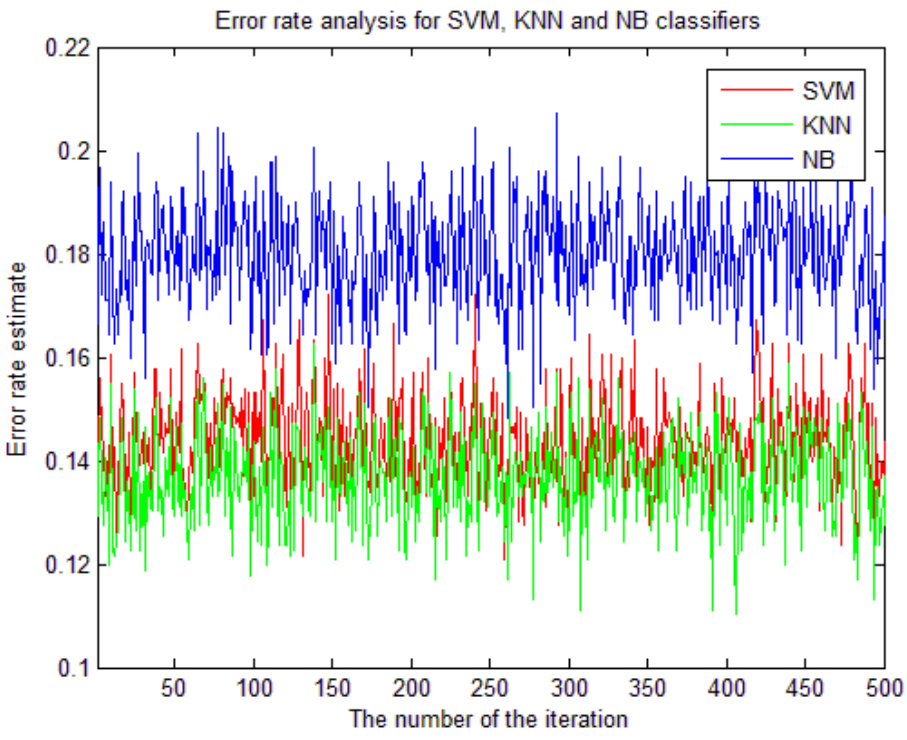
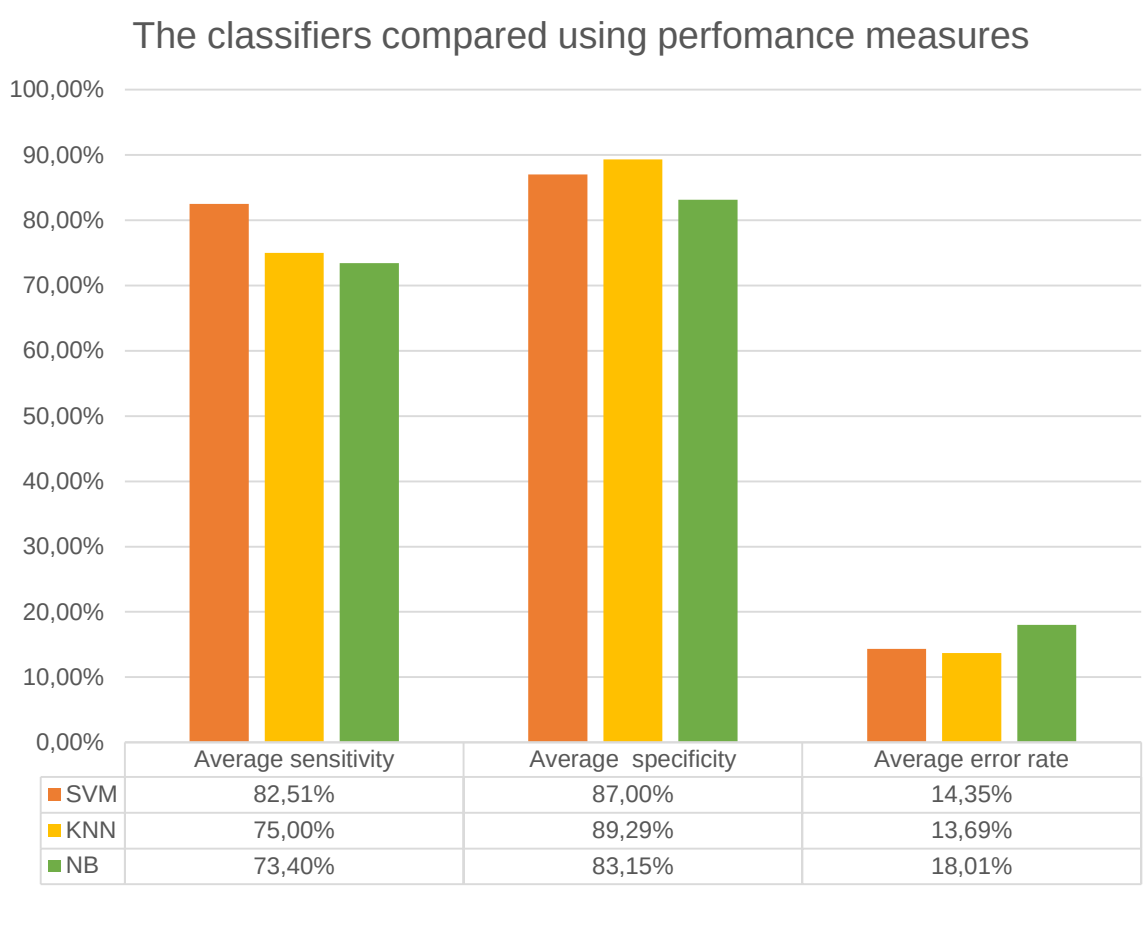


Figure 4.8: Graphical representation of specificity estimates for the three classifiers

4.6: CHAPTER SUMMARY

In this chapter, the SVM, NB and KNN classification algorithms were used to build three classifiers. Statistical assessment of these classifiers was done and the performance thereof evaluated and compared using the sensitivity, specificity and error rate. The results are summarised in Figure 4.9. The SVM classifier demonstrated a high probability of correctly classifying the fraudulent cases with a rate of approximately 83%. In comparison, the KNN and the NB classifiers achieved a fraud detection rate of 75% and 73.4% respectively. The KNN classifier had the highest probability of correctly classifying the clean cases, followed by the SVM classifier and the NB classifier, which had the lowest classification rate in terms of the clean cases. The classifiers with the lowest misclassification rates were the KNN and the SVM ones, with error rates of approximately 14%. The NB classifier had an error rate of roughly 18%.



	Average sensitivity	Average specificity	Average error rate
SVM	82,51%	87,00%	14,35%
KNN	75,00%	89,29%	13,69%
NB	73,40%	83,15%	18,01%

Figure 4.9: The three classifiers compared in terms of sensitivity, specificity and error rate

CHAPTER 5: CONCLUSION

5.1: INTRODUCTION

The purpose of this research study was to identify a statistical model to be applied in the detection of fraudulent electricity usage in NMBM. Such an intelligent model would predict fraudulent activities in electricity consumptions by detecting any abnormal usage patterns. Additional inspections were performed on potential fraudulent cases predicted by the model. Statistical classification algorithms were used to build three classifiers, of which the best classifier was used as the intelligent electricity fraud detection model. Generic algorithms, Support Vector Machines (SVM), Naïve Bayes (NB) and k-Nearest Neighbour (KNN) were utilised. The classifiers were built and tested on an empirical data set using the holdout and random subsampling methods.

In this chapter, the findings presented in Chapter 4 are discussed and conclusions suggested. In Section 5.2, the performance of the SVM classifier is discussed, followed by those of the NB and KNN classifiers in Sections 5.3 and 5.4 respectively. A comparison of all three classifiers is discussed in the final Section, 5.5.

5.2: THE SVM CLASSIFIER

Classification results for the SVM classifier were affected by the choice of the rbf kernel parameter, γ, and the capacity, C. Matlab R2012a, used to design and test the SVM classifier, had default parameters of, $\gamma = 1$ and $C = 1$. Firstly, the SVM classifier was trained and tested using the default parameters. Then the parameters were optimised and found to be, $\gamma = 1.3542$ and $C = 2.1639$. The random subsampling method was then used to train and test the SVM classifier 500 times. Each time, sensitivity, specificity and error rate estimates were calculated. Subsequently, averages for each of these measures were calculated. This process was carried out both before, and after, parameter optimisation.

The purpose of the proposed model was to detect fraudulent usage of electricity, and as such, sensitivity was nominated as the most significant measure of accuracy. The rationale for sensitivity, also referred to as the fraud detection rate, was that it measured the model's ability to detect fraudulent cases. Although the emphasis focussed on sensitivity, the specificity and error rates were also important, hence all three measures were studied.

On average, the error rate and the specificity were not affected by the SVM parameter optimisation, see Table 4.3 and Table 4.5. In contrast, the sensitivity rate increased from

approximately 80% to 83%. Thus, the SVM classifier, with optimised parameters, was used in the subsequent comparison of the three classifiers.

5.3: THE NB CLASSIFIER

The NB classifier assigned a posterior probability to each predicted tuple. In this study, a binary classification problem with classes, “fraudulent” and “clean” was solved. The historical electricity consumption data consisted of both fraudulent and clean customers. Training in Naïve Bayes was performed in such a way that, given the historical electricity consumption data, the probability that a customer was fraudulent, was computed and compared with the probability that the customer was clean. If the former probability was greater than the latter, that particular customer was classified as fraudulent, otherwise the customer was classified as clean. One of the test sets used to compute the performance measures for the NB classifier is shown in Table 5.1. This set consisted of 1 052 customers, of which 221 were fraudulent cases and 831 were clean. Under the columns, Mar-13, Apr-13,…,Feb-15, are normalised values of monthly kWh electricity consumptions, where, P(C|X) is the probability that the customer was clean given X, where X is 24-dimensional vector consisting of monthly kWh electricity consumptions for the 24 months period, and P(F|X) was defined similarly for the second class.

Table 5.1: A test set for testing the NB classifier

ID**	Mar-13	Apr-13	⋯	Feb-15	True labels	Predicted labels	P(C|X)	P(F|X)
1	0.4812	1.0000	⋯	0.4974	*Clean*	*Clean*	0.9980	0.0020
2	1.0000	1.0000	⋯	0.9118	*Clean*	*Clean*	1.0000	0.0000
3	1.0000	0.5043	⋯	0.6615	*Clean*	*Clean*	0.7075	0.2925
4	0.6734	0.6354	⋯	0.5463	*Fraudulent*	*Fraudulent*	0.3106	0.6894
5	0.8874	0.8874	⋯	0.8391	*Fraudulent*	*Clean*	1.0000	0.0000
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
1052	1.0000	0.9593		0.6095	*Fraudulent*	*Fraudulent*	0.0063	0.9937

When the NB classifier was trained, the posterior probabilities, P(C|X) and P(F|X) were computed and compared. Whenever P(F|X) was greater than P(C|X), that particular customer was classified as fraudulent, otherwise the customer was classified as clean. These comparisons can be seen in Table 5.1. Note that these probabilities were not always 1 and 0. For instance, there was a case where P(C|X) = 0.7075 and P(F|X) = 0.2925 for a particular customer and that customer was classified as clean as P(C|X) was greater than P(F|X). There was also one case when P(C|X)=0.3106 and P(F|X)=0.6894, and since P(F|X) was greater than P(C|X), that particular customer was classified as fraudulent.

The NB classifier was trained and tested using 500 different partitions of training and test sets. Each time 67% of the tuples were designated to the training set and 33% to the test set. Every time the NB classifier was tested, the sensitivity, specificity and error rate were computed. Finally, 500 estimates for each of these performance measures were obtained, see Table 4.8.

As the number of clean cases was very large when compared to fraudulent ones, the average specificity was always higher than the average sensitivity. The NB classifier had a fraud detection rate (sensitivity) of roughly 73%, that is, out of every 100 fraudulent customers, the NB classifier correctly detected approximately 73. On average, the NB classifier detected clean cases, with probability close to 85%, together with an error rate of lower than 20%.

5.4: THE KNN CLASSIFIER

Before training the KNN classifier, the value of k was determined experimentally to be 21. The distance metric used to measure similarity between the unknown test tuple to its k-nearest neighbours was the City Block distance. The rationale for using the City Block distance was that it performed better than the Euclidean distance, as seen in Figure 4.4. These results were consistent with findings of the study by Mulak and Tallar (2015).

The value of k equals to 21 and the City Block distance were then used to train and test the KNN classifier. To account for variation in the training and test sets, the KNN classifier was trained and tested with 500 different training and test set partitions. Each time, the sensitivity, specificity and error rate estimates were calculated, the averages of which are summarised in Table 4.12. The KNN classifier produced a fraud detection rate close to 75%, with a very high specificity of roughly 90%, and small error rate of less than 14%.

5.5: COMPARISON OF THE CLASSIFERS

A flowchart illustrating the methodology followed in comparing the three classifiers is described in Figure 5.1. The loop started with p equals to one, the empirical data was then randomly partitioned into training set and test set. The training set was used to build the SVM, NB and KNN classifiers. These learned classifiers were used in conjunction with the test set in order to estimate accuracy, which was estimated by using the sensitivity, specificity and error rates. The value of p was then incremented in steps of one. When the value of p was less than or equal to 500, the above process was repeated. At the end of this procedure, 500 different estimates for the sensitivity, specificity and error rates were

attained. The averages of these estimates were calculated and the results thereof summarised in Table 5.2.

Table 5.2: Average accuracy estimates

	Estimates		
Performance measure	**SVM**	**KNN**	**NB**
Average sensitivity	82.51%	75.00%	73.40%
Average specificity	87.00%	89.29%	83.15%
Average error rate	14.35%	13.69%	18.01%

The error rate estimate for the NB classifier was significantly higher than that of the SVM and KNN classifiers, which had similar estimates by solely investigating the error rate, the SVM and the KNN classifiers received equal votes and the NB classifier minimum votes. However, the error rates were not sufficient to determine the classifiers' ability to correctly classify previously unseen data. As a result, in addition to the error rate, the specificity and sensitivity were also compared. The ability to correctly classify clean customers was similar for both the KNN and SVM classifiers, as evident by their almost equal estimates of specificity. The NB classifier performed poorly in correctly classifying the clean cases when compared to the other two classifiers. The specificity estimate for the NB classifier was close to 83%, and although this was an acceptable estimate, the SVM and KNN classifiers achieved higher estimates, of about 87% and 89% respectively. Prior to the consideration of the sensitivity, the SVM and the KNN classifiers were tied, and they both overshadowed the NB classifier. To break this tie, sensitivity estimates were also used in addition to the error rate and specificity estimates. The NB classifier had the smallest sensitivity estimate, almost 73%, followed by that of the KNN classifier, with roughly 75%. The SVM classifier had the highest sensitivity estimate, at approximately 83%, and as such, was deemed the best classifier.

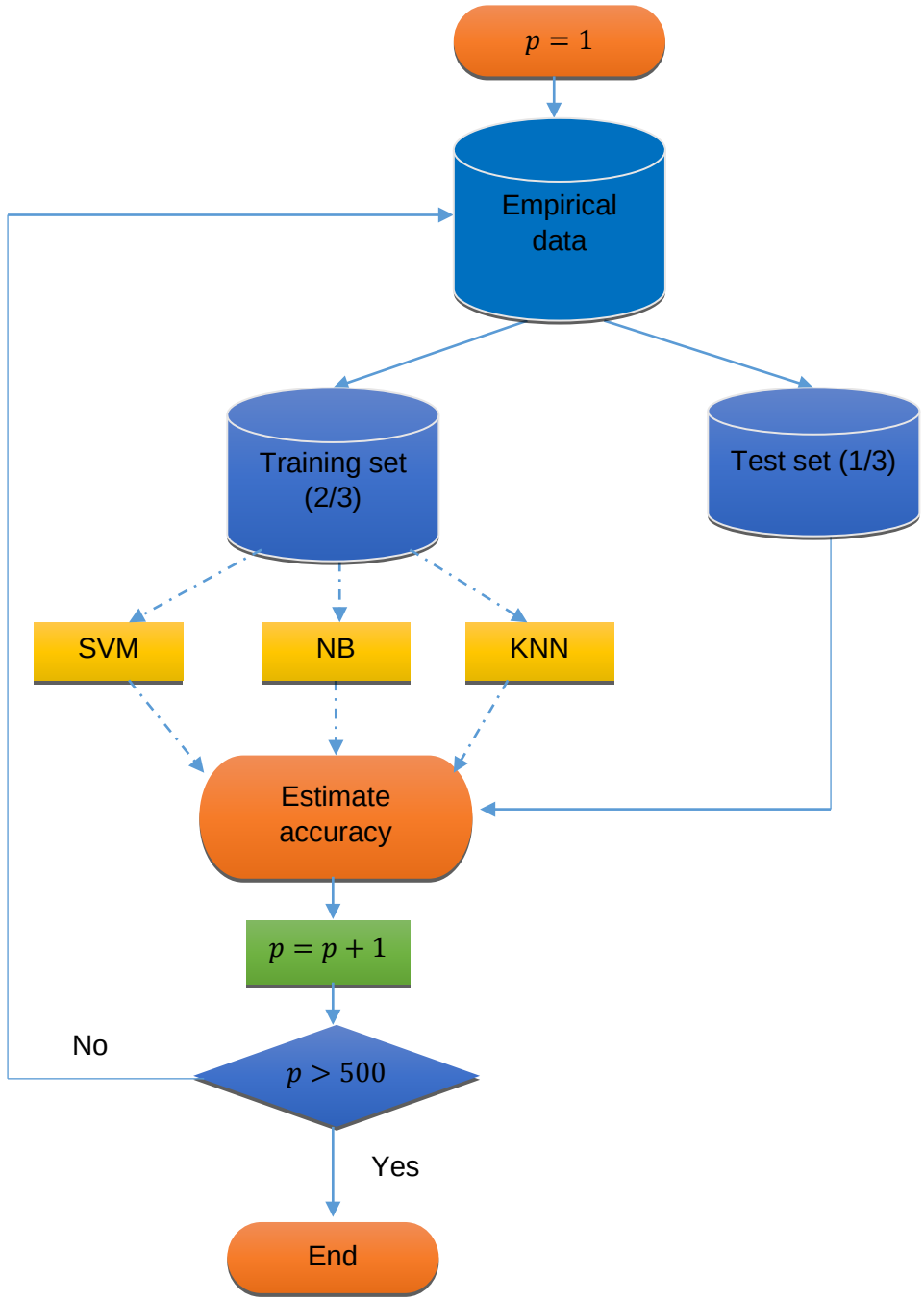


Figure 5.1: Flowchart of the methodology followed in comparing the three classifiers

5.6: CONCLUSION

The ability of any classifier to correctly classify previously unseen data can be affected by the choice of the training and test sets. Hence, the classifiers developed in this study were trained and tested on several different training and test sets. The calculated estimates had small variations, as was evident from the standard deviations, which were small for all the accuracy estimates. The performance of a classifier was determined by how well it could

classify previously unseen tuples. Hence the test data set was used to evaluate the performance of the classifiers that were built. In this study, three classifiers were developed and used to classify test data sets. The SVM, the NB and the KNN classification algorithms were used to train the developed classifiers. The sensitivity, specificity and error rates were performance measures used in order to evaluate the classifiers' performance.

On average, the NB classifier erroneously classified customers with a rate of roughly 18%, which is high when compared to the 14% of the SVM and KNN classifiers. Fraudulent customers were correctly classified by the NB and the KNN classifiers, with probabilities close to 73% and 75% respectively. The sensitivity estimate for the NB classifier was the lowest. A possible reason for this was that the NB classifier assumed the values of the attributes were independent. The attributes used in this study may have been conditionally dependent. For future research, the Bayesian Belief network, which assumes conditional dependency, could be used instead of the Naïve Bayes. In contrast, the SVM classifier had a fraud detection rate of approximately 83%, better than the NB and KNN rates. The ability to correctly classify both clean and fraudulent cases, with probabilities of over 80%, gave the SVM classifier an edge over its competitors.

This study proposed a classification model for the detection of non-technical losses in the NMBM. The proposed framework used the SVM method to build a fraud detection model for the classification of customers' electricity consumption patterns. Results showed that this model was reliable in the identification of fraudulent electricity consumers in the metro. These consumers were inspected to confirm their status. Consumers that were found guilty were fined by the municipality. The implementation of the SVM method benefited the municipality in its handling of non-technical losses and resulted in tremendous savings.

REFERENCES

AlgoaFM. (2014, July 11). *AlgoaFMNews*. Retrieved Jan 7, 2015, from Former mayor's company to help curb electricty theft in Nelson Mandela Bay: http://www.algoafm.co.za/article.aspx?id=100018

Alsberg, B. K., Goodacre, R., Rowland, J. J., & Kell, D. B. (1997). Classification of Pyrolysis Mass Spectra by Fuzzy Multivariate Rule Induction-Comparison with Regression, K-Nearest Neighbour, Neural and Decision-Tree Methods. *Analytica Chimica ACTA*, 389-407.

Anand, R. D., De, S., & Naveen, S. A. (2003). Design and Development of vigilant energy metering system. *In the Proceeding of Student Conference on Research and Development* (pp. 15-18). New Dehli, India: IEEE.

Bandim, C. J., Alves, J. E., Pinto, A. V., Souza, F. C., Loureiro, C. A., & Galvez, D. F. (2003). Identification of Energy Theft and Tampered Meters using Central Observer Meter: A Mathematical Approach. *In the proceedings of IEEE PES Transmission and Distribution Conference and Exposotion* (pp. 163-168). Rio de Janeiro, Brazil: IEEE.

Bhattacharyya, S., Jha, S., Tharakunnel, K., & Westland, J. C. (2011). Data Mining for Credit Card Fraud: A Comparative Study. *Decision Support Systems, 50*(3), 602-613.

Boser, B. E., Guyon, I. M., & Vapnik, N. V. (1992). A Training Algorithm for Optimal Margin Classifiers. *COLT'92 Proceedings of the fifth annual workshop on computational learning theory* (pp. 144-152). New York: ACM New York, NY, USA (C) 1992.

Brun, A. D., Pinto, J. O., Pinto, A. M., Sauer, L., & Colman, E. (2009). Fraud Detection in Electric Energy Using Differential Evolution. *International Conference on Intelligent Systems Applications to Power Systems*, 1-5.

Cabral, J. E., Pinto, J., & Pinto, A. (2009). *Fraud Detection System for High and Low Voltage Electricity Consumers based on Data Mining.* Power Engineering Society, IEEE General Meeting.

Cabral, J. E., Pinto, J., Linares, K., & Pinto, A. (2006). Methodology for Fraud Detection using Rough Sets. *IEEE International Conference on Granular Computing*, 244-249.

Davidson, I. (2002). Evaluation and Effective Management of Non-Technical Losses in Electrical Power Networks. *IEEE Africon*, 473-477.

Depuru, S. S., Wang, L., & Devabhaktuni, V. (2010). A Conceptual Design Using Harmonics to Reduce Pilfering of Electricity. *In the Proceeding of IEEE PES General Meeting.* Minneapolis, Minnesota: IEEE.

Depuru, S. S., Wang, L., & Devabhaktuni, V. (2011). Support Vector Machine Based Data Classification for Detection of Electricity Theft. *Power Systems Conference and Explosion (PSCE)*, 1-8.

Doorduin, W. A., Mouton, H. T., Herman, R., & Beukes, H. J. (2004). Feasibility Study of Electricity Theft Detection using Mobile Remote Check Meters. *IEEE AFRICON 2004*, 373-376.

dos Angelos, E. W., Saavedra, O., Cortes, O., & Souza, A. d. (2011). Detection and Identification of Abnormalities in Customer Consumptions in Power Distribution Systems. *IEEE transcaction on power delivery, 26*(4), 2436-2442.

Elkan, C. (2001). Magical Thinking in Data Mining: Lessons from CoIL Challenge 2000. *In the Proceedings of the seventh ACM SIGKDD international conference on Knowledge Discovery and Data Mining* (pp. 426-431). San Francisco, California: ACM.

ESI_Africa. (2014, July 8). *Electricity theft hits economy hard*. Retrieved January 7, 2015, from ESI Africa: http://www.esi-africa.com/electricity-theft-hits-economy-hard/

Fix, E., & Hodges, J. (1952). *Discrimanatory Analysis, Nonparametric Discrimination: Consistency Properties.* Randolph, Field: Report 4, Project 21-49-004 US Air Force School of Aviation Medicine.

Fradkin, D., & Muchnik, I. (2006). Support Vector Machines for Classification. *Discrete methods in epidemiology, 70*, 13-20.

Golmohammadi, K., & Zaiane, O. R. (2012). Data Mining Applications for Fraud Detection in Securities Market. *In the Proceedings of Intelligence and Security Informatics Conference* (pp. 107-114). IEEE.

Guyon, I., Weston, J., Barnhill, S., & Vapnik, V. (2002). Gene Selection for Cancer Classification using Support Vector Machines. *Machine Learning, 46*(1), 389-422.

Hamid, F., & Sepehri, M. M. (2011). A Data Mining Framework for Detectioing Subscription Fraud in Telecommunication. *Journal on Eginnering Applications of Artificial Intelligence, 24*, 182-194.

Han, J., & Kamber, M. (2006). *Data Mining: Concepts and Techniques.* Morgan Kaufmann Publishers.

Hand, D., & Henley, W. E. (1996). A k-Nearest-Neighbour Classifier for Assessing Consumer Credit Risk. *Journal of the Royal Statistical Society. Series D (The Statistician), 45*(1), 77-95.

Hand, D., & Vinciotti, V. (2003). Choosing k for Two-Class Nearest Neighbour Classifiers with Unbalanced Classes. *Pattern Recognition Letters*, 1555-1562.

Henley, W. E., & Hand, D. J. (1996). A k-Nearest-Neighbour Classifier for Assessing Consumer Credit Risk. *Journal of the Royal Statistical Society. Series D (The Statistician), 45*(1), 77-95.

Hernandez, Y., Arroyo-Figueroa, G., Rodriguez, G., Santos, M., & Escobedo, H. (2015). Towards a Framework to Detect and Prevent Non-technical Losses in Power Disrtibution Based on Data-Mining Techniques and Bayesian Networks. *2015 Fourteen Mexican International Conference on Artificial Intelligence (MICAI)* (pp. 157-161). Cuernavaca: IEEE.

Hsu, C.-W., Chang, C.-C., & Lin, C.-J. (2003). *A Paractical Guide to Support Vector Classification.*

Jaya, B. J., & Tamilselvi, J. J. (2013). Assessment of Fraud Pretentious Business Region Research Articles Using Data Mining Approaches. *International Journal on Computer Science and Engineering, 5*(7), 653-659.

Joachims, T. (1998). Text categorisation with support vector machines: Learning with many relevant features. *In the Proceedings of the European Conference on Machine Learning.* Springer.

Johnson, L. F. (2012). Access to Antiretroviral Treatment in South Africa. *Southern African Journal of HIV Medicine 2004-2011, 13*(1).

Kohavi, R. (1995). A Study of Cross Validation and Bootstrap for Accuracy Estimation and Model Selection. *International Joint Conference on Artificial Intelligence*, (pp. 1137-1143).

Lilly, R. (2014, June 6). *EE publishers*. Retrieved March 18, 2015, from Eskom's energy and revenue loss management: http://www.ee.co.za/article/eskoms-ernegy-revenue-loss-management.html

Michel, P., & Kaliouby, R. E. (2003). Real time facial expression recognition in video using support vector machines. *In the Proceedings of the 5th international conference on Multimodal interfaces*, (pp. 258-264). New York, NY, USA.

Mulak, P., & Talhar, N. (2015, July). Analysis of Distance Measures Using K-Nearest Neighbor Algorithm on KDD Dataset. *International Journal of Sciecne and Research, 4*(7), 2101-2104.

Nagi, J., Keem, S. Y., Tiong, S. K., Ahmed, S. K., & Mohamad, M. (2008, December 1-3). Non-technical loss analysis for detction of electrcity theft using support vector machines. *Power and energy conference, 2008. PECon 2008. IEEE 2nd international*, 907-912.

Nagi, J., Yap, K. S., Tiong, S. K., Ahmed, S. K., & Mohamad, M. (2010). Nontechnical loss detection for metered customers in power utility using support vector machines. *IEEE Transactions on Power Delivery, 25*(2), 1162-8977.

Nizar, A. H., Zhao, D. Y., Zhao, J., & Zhang, P. (2007). A Data Mining Based NTL Analysis Method. *Power Enginnering Society General Meeting 2007 IEEE* (pp. 1-8). Tampa, FL: IEEE.

Pasdar, A., & Mirzakuchaki, S. A. (2007). A Solution to Remote Detecting of Illegal Electricity Usage Based on Smart Metering. *In the Proceedings of the 2nd International Workshop on Soft Computing Applications* (pp. 163-167). Oradea, Romania: IEEE.

PowerNews. (2013, July). *Municipalities,utilities meet in bid to protect dwindling revenues*. Retrieved March 13, 2015, from PowerNews: http:/www.powernews.co.za/news/entry/municipalities-utilities-meet-in-bid-to-protect-dwindling-revenues

PRNewswire. (2015, December 9). *World Loses $89.3 Billion to Electrcity Theft Annualy, $58.7 Billion in Emerging Markets*. Retrieved January 22, 2016, from PRNewswire: http://prnewswire.com/news-realise/world-losses-893-billion-to-electricity-theft-annually-587-billion-in-emerging-markets-300006515.html

Provost, F., Tom, F., & Kohavi, R. (1997). The Case Against Accuracy Estimation for Comparing Induction Algorithms. *In Proceedings of the Fifteenth International Conference on Machnine Learning*, (pp. 445-453).

Ravisankar, P., Ravi, V., Rao, G., & Bose, I. (2011). Detection of Financial Statement Fraud and Feauture Selection using Data Mining Techniques. *Journal on Decision Support Systems, 50*(2), 491-500.

Rennie, J., Shih, L., Teevan, J., & Karger, D. R. (2003). Tackling the Poor Assumptions of Naive Bayes Text Classifiers. *In the Proceedings of the 20th International Conference on Machine Learning*, *3*, pp. 616-623. Washington DC.

Rish, I. (2001). An Empirical Study of the Naive Bayes Classifier. *IJCAI 2001 workshop on empirical methods in artificial intelligence, 3*(22), 41-46.

Stalmans, E., & Irwin, B. (2011). A Framework for DNS Based Detection and Mitigation of Malware Infections on a Network. *Information Security South Africa (ISSA)*, 1-8.

Steinbach, M., Vipin, K., & Tan, P.-N. (2006). *Introduction to Data Mining.* Boston, USA: Pearson Education India.

T&DWorldMagazine. (2015, February 12). *HOME > SMART GRID > INDIA TO SPEND $21.6 BILLION ON SMART GRID INFRASTRUCTURE BY 2025*. Retrieved June 30, 2016, from Transimission and Distribution World: http://tdworld.com/smart-grid/india-spend-216-billion-smart-grid-infrastructure-2025

Tamilselvi, J. J., & Jaya, B. J. (2013). Assesment of Fraud Pretentious Business Region Research Articles using Data Mining Approaches. *International Journal on Computer Science and Engineering, 5*(7), 653-659.

van der Merwe, D. W., & Engelbrecht, A. P. (2003). Data Clustering using Particles Swarm Optimisation. *Evolutionary Computation, 2003. CEC'03. The 2003 Congress on. 1*, pp. 215-220. IEEE.

van der Walt, C., & Barnard, E. (2006). Data Characteristics that dtermine Classifier Performance. *SAIEE Africa Research Journal, 98*(3), 87-83.

Vapnik, V. N. (1999). An overview of statistical learning theory. *IEEE transactions on neural networks, 10*(5), 988-999.

Viaene, S., Derrig, R. A., & Dedene, G. (2004). A Case Study of Applying Boosting Naive Bayes to Claim Fraud Diagnosis. *IEEE Transactions on Knowledge and Data Engineering, 16*(5), 612-620.

Weinberger, Q. K., Blitzer, J., & Saul, K. L. (2009). Distance Metric Learning for Large Margin Nearest Neighbor Classification. *The Journal of Machnine LEarning Research, 10*, 207-244.

Zhao, J., Dong, Z. Y., & Li, X. (2007). An improved Naive Bayesian classifier with advanced discretization method. *International Journal of Intelligent Systems Technologies and Application, 3*(3-4), 241-256.

APPENDICES

APPENDIX A: SVM CLASSIFICATION MATLAB CODE

```
clear
clc
%% In this program a SVM classifier is trained and tested using given data
set

%Read the data from microsoft excel file
filename = 'Sample_with_labels.xlsx'; % Excel filename
X = xlsread(filename,'B2:Y3157'); %X is a 3156 by 24 matrix
y = xlsread(filename,'AA2:AA3157'); %y contain class labels
[d,n] = size(X); % d is number of observations and n is number of variables

%NORMALISE X
XN = zeros(d,n); % XN is a matrix containing normalised values of X
    for j = 1:d
        XN(j,:) = (X(j,:)-min(X(j,:)))/range(X(j,:));
    end

%Use random subsampling to obtain various accuracy estimmates
p = 500; %p is the number of iterations
specificity = zeros(p,1);
sensitivity = zeros(p,1);
error_rate = zeros(p,1);
for i = 1:p

% Create indices for training and test sets: 2/3 training: 1/3 test
[train_indices, test_indices] = crossvalind('HoldOut', d,1/3);

%Create training and test sets
training_set = XN(train_indices,:);
training_classlabels = y(train_indices);
test_set = XN(test_indices,:);
test_classlabels = y(test_indices);

%Train the SVM model using training sets.
svm_mdl =
svmtrain(training_set,training_classlabels,'Kernel_Function','rbf','autosca
le',false,'rbf_sigma',1.3542,'boxconstraint',2.1639);

%Classify the test set.
predicted_classlabels = svmclassify(svm_mdl,test_set);

%Check classification perfomance
cp =
classperf(test_classlabels,predicted_classlabels,'Positive',1,'Negative',0)
;

specificity(i) = cp.Specificity;
sensitivity(i) = cp.Sensitivity;
error_rate(i) = cp.ErrorRate;

end
```

SVM PARAMETER OPTIMISATION CODE

```
clear
clc

function yfit = ...
    crossfun(xtrain,ytrain,xtest,rbf_sigma,boxconstraint)

% Train the model on xtrain, ytrain,
% and get predictions of class of xtest
svmStruct = svmtrain(xtrain,ytrain,'Kernel_Function','rbf',...
   'rbf_sigma',rbf_sigma,'boxconstraint',boxconstraint);
yfit = svmclassify(svmStruct,xtest);


%Read the data from microsoft excel file
filename = 'Sample_with_labels.xlsx'; % Excel filename
X = xlsread(filename,'B2:Y3157'); %X is a 3156 by 24 matrix
y = xlsread(filename,'AA2:AA3157'); %y contain class labels
[d,n] = size(X); % d is number of observations and n is number of variables

%NORMALISE X
XN = zeros(d,n); % XN is a matrix containing normalised values of X
    for j = 1:d
        XN(j,:) = (X(j,:)-min(X(j,:)))/range(X(j,:));
    end

%Set up a partition for cross validation. This step causes
%the cross validation to be fixed. Without this step, the cross
%validation is random,so a minimization procedure
%can find a spurious local minimum
c = cvpartition(d,'kfold',10);



%Set up a function that takes an input z=[rbf_sigma,boxconstraint], and
returns the cross-validation value of exp(z). The reason to take exp(z) is
twofold:
%1. rbf_sigma and boxconstraint must be positive.
%2. You should look at points spaced approximately exponentially apart.

%This function handle computes the cross validation at parameters
exp([rbf_sigma,boxconstraint]):
minfn = @(z)crossval('mcr',XN,y,'Predfun', ...
    @(xtrain,ytrain,xtest)crossfun(xtrain,ytrain,...
    xtest,exp(z(1)),exp(z(2))),'partition',c);

%Search for the best parameters [rbf_sigma,boxconstraint] with fminsearch,
setting looser tolerances than the defaults.
opts = optimset('TolX',5e-4,'TolFun',5e-4);
[searchmin fval] = fminsearch(minfn,randn(2,1),opts)
z = exp(searchmin)
```

APPENDIX B: NB CLASSIFICATION MATLAB CODE

```
clear
clc
%% In this program a Naive Bayes classifier is trained and tested using
given data set

%Read the data from microsoft excel file
filename = 'Sample_with_labels.xlsx'; % Excel filename
X = xlsread(filename,'B2:Y3157'); %X is a 3156 by 24 matrix
[~,true_labels,~] = xlsread(filename,'Z2:Z3157'); %
[d,n] = size(X); % d is number of observations and n is number of variables

%NORMALISE X
XN = zeros(d,n); % XN is a matrix containing normalised values of X
    for j = 1:d
        XN(j,:) = (X(j,:)-min(X(j,:)))/range(X(j,:));
    end

%Use random subsampling to obtain various accuracy estimmates
p = 500; %p is the number of iterations
specificity = zeros(p,1);
sensitivity = zeros(p,1);
error_rate = zeros(p,1);
for i = 1:p

% Create indices for training and test sets: 2/3 training; 1/3 test

[train_indices, test_indices] = crossvalind('HoldOut', d,1/3);

%Create training and test sets
training_set = XN(train_indices,:);
train_labels = true_labels(train_indices);
test_set = XN(test_indices,:);
test_labels = true_labels(test_indices);

%Train the Naive Bayesian classifier
nb_mdl = NaiveBayes.fit(training_set, train_labels);

%Predict class labels of the test set
predicted_labels = predict(nb_mdl,test_set);

%Check classification perfomance
cp =
classperf(test_labels,predicted_labels,'Positive','Fraudulent','Negative','
Clean');

%Compute posterior probalilities for each predicted class
post = posterior(nb_mdl,test_set);

specificity(i) = cp.Specificity;
sensitivity(i) = cp.Sensitivity;
error_rate(i) = cp.ErrorRate;

end
```

APPENDIX C: KNN CLASSIFICATION MATLAB CODE

```
clear
clc

%READ THE DATA FROM EXCEL FILE
filename = 'Sample_with_labels.xlsx';
X = xlsread(filename,'B2:Y3157'); %X is a 3156 by 24 matrix
true_labels = xlsread(filename,'AA2:AA3157'); % True class labels
[d,n] = size(X); % d is number of observations and n is number of variables

%NORMALISE THE DATA USING MIN-MAX NORMALISATION
D = zeros(d,n); % D is the matrix containing normalised values of X
    for j = 1:d
        D(j,:) = (X(j,:)-min(X(j,:)))/range(X(j,:));
    end

p = 500; %number of iterations
sensitivity = zeros(p,1);
specificity = zeros(p,1);
error_rate = zeros(p,1);

    for i = 1:p

%PARTITION D INTO TRAINING AND TEST SETS
[train_indices,test_indices] = crossvalind('HoldOut',d,(1/3));
training_set = D(train_indices,:);
test_set = D(test_indices,:);
training_labels = true_labels(train_indices);

%CLASSIFICATION USING K NEAREST NEIGHBOUR APPROACH
predicted_labels =
knnclassify(test_set,training_set,training_labels,21,'cityblock');

%CHECK CLASSIFICATION THE PERFOMANCE
test_labels = true_labels(test_indices);
 %Positive label correspond to "fraudulent" and "negative" to clean
cp = classperf(test_labels,predicted_labels,'Positive',2,'Negative',1);

sensitivity(i) = cp.Sensitivity;
specificity(i) = cp.Specificity;
error_rate(i) = cp.ErrorRate;
    end
```

APPENDIX D: MATLAB CODE FOR ALL THE CLASSIFIER

```
clear
clc

%READ THE DATA FROM EXCEL FILE
filename = 'Sample_with_labels.xlsx';
raw_data = xlsread(filename,'B2:Y3157'); %X is a 3156 by 24 matrix
true_labels = xlsread(filename,'AA2:AA3157'); % True class labels
[d,n] = size(raw_data); % d is number of observations and n is number of
variables


%NORMALISE THE DATA USING MIN-MAX NORMALISATION
D = zeros(d,n); % D is the matrix containing normalised values of X
    for j = 1:d
        D(j,:) = (raw_data(j,:)-min(raw_data(j,:)))/range(raw_data(j,:));
    end

    p = 500; %number of iterations
    knn_sensitivity = zeros(p,1);
    svm_sensitivity = zeros(p,1);
    nb_sensitivity = zeros(p,1);

    knn_specificity = zeros(p,1);
    svm_specificity = zeros(p,1);
    nb_specificity = zeros(p,1);

    knn_error_rate = zeros(p,1);
    svm_error_rate = zeros(p,1);
    nb_error_rate = zeros(p,1);

for i = 1:p

    %PARTITION D INTO TRAINING AND TEST SETS
    [train_indices,test_indices] = crossvalind('HoldOut',d,(1/3));
    training_set = D(train_indices,:);
    test_set = D(test_indices,:);
    training_labels = true_labels(train_indices);
    test_labels = true_labels(test_indices);

    %OBTAIN PREDICTED CLASSES FROM EACH MODEL
    %knn with k = 21 and using sum of absolute differences
    knn =
knnclassify(test_set,training_set,training_labels,21,'cityblock');

    %svm with optimum parameters of gamma = 1.3542 and C = 2.1639
    svm_structure =
svmtrain(training_set,training_labels,'autoscale',false,...,
        'kernel_function','rbf','rbf_sigma',1.3542,'boxconstraint',2.1639);
    svm = svmclassify(svm_structure,test_set);

    %nb with posterior probabilities, post
    nb = predict(NaiveBayes.fit(training_set,training_labels),test_set);
    post =
posterior(NaiveBayes.fit(training_set,training_labels),test_set);

    %COMPARE PERFOMANCE OF THE MODELS
```

```
    cp_knn = classperf(test_labels,knn,'Positive',2,'Negative',1);
    cp_svm = classperf(test_labels,svm,'Positive',2,'Negative',1);
    cp_nb = classperf(test_labels,nb,'Positive',2,'Negative',1);

    %CALCULATE PERFORMANCE MEASURES
    knn_sensitivity(i) = cp_knn.Sensitivity;
    svm_sensitivity(i) = cp_svm.Sensitivity;
    nb_sensitivity(i) = cp_nb.Sensitivity;

    knn_specificity(i) = cp_knn.Specificity;
    svm_specificity(i) = cp_svm.Specificity;
    nb_specificity(i) = cp_nb.Specificity;

    knn_error_rate(i) = cp_knn.ErrorRate;
    svm_error_rate(i) = cp_svm.ErrorRate;
    nb_error_rate(i) = cp_nb.ErrorRate;

end
```