RELATING MATHEMATICS TO MACHINE LEARNING THROUGH ALGORITHM DEVELOPMENT FOR BIG DATA ANALYSIS

BY

DIAMOND TAKUDZWA CHIRISA

(R125828Y)

submitted in partial fulfilment of the bachelor of science in mathematics honours degree (hmath)

DEPARTMENT OF APPLIED MATHEMATICS AND STATISTICS

MIDLANDS STATE UNIVERSITY

GWERU, ZIMBABWE

YEAR: 2017

DEDICATION

I dedicate the work in this project to my family and colleagues without whom this project would not have been a success.

APPROVAL FORM

The undersigned certify that they have read and recommend to Midlands State University for acceptance, a research project entitled: **"DEVELOPING A MACHINE LEARNING ALGORITHM FOR DATA CLASSIFICATION"** submitted by Diamond Takudzwa Chirisa in partial fulfilment of the Faculty of Science and Technology for the Bachelor of Science in Mathematics Honors Degree of Midlands State University.

---------- ----------

SUPERVISOR **DATE**

---------- ----------

CHAIRPERSON **DATE**

DECLARATION

I, Diamond Takudzwa Chirisa, do hereby declare this dissertation is my original research work. This dissertation has never been submitted by anyone to any other educational institution for a Degree and or any other form of research.

Students Signature…… Date……

Supervisors Signature…… Date……

ACKNOWLEDGEMENTS

I acknowledge the supervision of Mr Murewi whose effort and support could not be dispensed. Without him this research would not have been a success. Not only has he helped me in this work, he has also helped me shape my career as far as career paths that relate to such work are concerned. I would also like to thank the Machine Learning community more specifically M Zwitter and M Soklic of University Medical Centre, Institute of Oncology, Ljubljana, Yugoslavia for providing the dataset that were used in this research.

I would also like to thank my sisters and my brother who advised me to go for a mathematics degree over a musical degree. I admit I had and will continue having very pleasurable moments working with mathematics. Without forgetting my colleagues, I also appreciate their company and as well admit that completing my degree would have been such a hard task had it not been for their encouragement and company.

ABSTRACT

Data has increased at an exponential rate and has outpaced our capability to analyze it. However, new ways of data analysis, which thrive in big data such as Machine Learning (ML) have emerged. This study explores Machine Learning by creating a Machine Learning algorithm based on Support Vectors. This was done by converting mathematical formulations into a computer algorithm that was then used for data classification. The algorithm was evaluated and compared to other algorithms. The results of the evaluation show that the algorithm was accurate at binary classification. Comparisons to other algorithms using both the iris and breast cancer datasets show that algorithms based on Support Vectors are generally more accurate at data classification. This means that the approach that was used in this study can be used in businesses to determine whether a person will return loan or not or whether a particular student can finish a degree program or not based on past data. The study also indicated that Support Vector Machines algorithm training require more computing power as data gets bigger. Hence, it suggested use of high performance computing for big data analysis.

TABLE OF CONTENTS

FIGURES

TABLES

LIST OF APPENDICES

LIST OF ACRONYMS.

OOP	Object Oriented Programming
ML	Machine Learning
DS	Data Science
AI	Artificial Intelligence
DL	Deep Learning
SVM	Support Vector Machine
GB	Gradient Boosting
RF	Random Forests
LR	Linear Regression
CART	Classification and Regression Trees
KNN	K-Nearest Neighbors
LDA	Linear Discriminant Analysis
NB	Gaussian Naïve Bayes
NN	Neural Networks
PCA	Principal Component Analysis
MGI	Mckinsey Global Institute
IBM	International Business Machines Corporation
EY	Earnest and Young
CIS	Center for Internet and Society
FPF	Future of Privacy Forum
IDC	International Data Corporation

CHAPTER 1

1.1 INTRODUCTION

1.2 BACKGROUND OF THE STUDY

The amount of digital data in the universe is growing at an exponential rate, doubling every two years, and changing how we live in the world (Tech Insider, 2015). This has been caused by an increased usage of computers as well as mobile devices. Not only can data now accumulate by the usage of computer systems in offices, people are now interacting with systems on the go (McKinsey, 2011). This has created a great need for automated and adaptive ways of analyzing huge amount of data and provide it to system users within a click of a button or a touch of a screen as in modern day devices (McKinsey, 2011). One might really wonder why the term "system" is being the focal point of the discussion, we are living in a digital world where one way or the other, everyone is a user of some system and is as well taking a significant part in the global digital data generating process (Marr, nd.)

Not only has data increased over the past years, our capabilities of analyzing it have also improved. We have Artificial Intelligence (AI) emerging from Computer Science (Marr, 2016). As time went by with the need of analyzing huge amounts of data, AI merged with mathematics to produce fields such as Data Science (DS) which was later subdivided into fields such Machine Learning (ML), Deep Learning, Business Analytics (BA) and many other data analytical fields (Smith, 2006). With the advent of parallel processors, usually GPU's (Graphic processing Unit often from nVidia) the computing power issue has been advanced

to the point where for example, Machine Learning is now possible (Brynjolfsson & Mcafee, 2017).

The increase of data and our capabilities to analyze it has somehow incapacitated and or made old approaches to data analysis (normally described as traditional statistics and or mathematical modelling) irrelevant (Breiman, 2001). This is because of the assumptions they have on data analysis. For example, inferential statistics based its analysis on the fact that populations are too huge to be studied hence need to study samples and infer the results to a bigger population (eds. Lin, Genest, Banks, Molenberghs, & Wang, 2013). Now that organizations have big datasets, which represent huge and or entire populations at hand, such assumptions are no longer valid (Every, 2015). In addition, it now seems absurd to get a sample and infer a population that we already have. Moreover, studying patterns and or relationships in a population is no longer an issue since there are available softwares that have astounding graphical presentations of existing populations' data, which expose any pattern, and or relationship that exist in them. According to Lin et al., (2013), it is now more profitable for organizations to focus more on automated predictions, which is what for example Machine Learning does best.

Big data has become a popular business buzzword today, and it is signaling limitless and exciting possibilities in the ability to keep and or store complex and deep information and mining this information in a sophisticated manner to reveal important and or key insights into transactional patterns and behaviors. From banking and business to agriculture, health and even disaster management, the opportunities to mine and or exploit data to benefit societies and businesses alike appear to be infinite (IDC, 2012).

While big data has presented a great opportunity to improve efficiency and profitability in organizations, it is so sad that only a handful of organizations have started to realize the potential

of big data analytics. Just about 0.5% of all the data that is being collected is analyzed and the percentage is shrinking as more data is collected (Guess, 2015). According to Auld (2017), limited IT budgets and the dearth of skilled resources impede big data and analytics initiatives across organizations in developing countries. Moreover, the slow adoption of big data analytics in schools and or colleges have really created a great scarcity in resources available to help utilize big data that is now available to organizations to provide them with unprecedented insights (Auld 2017). This has made it hard for organizations mostly in developing countries to make better decisions and wiser investments based on the data that is at their disposal. Hence, a great need to explore new approaches to data analysis.

1.3 PROBLEM STATEMENT

The growing increase in usage of computers has created big data. This huge amount of information has remained **untapped** since early analytical approaches that are currently being used by organizations for data analysis mainly focus on sampling and inference. This has largely been as a result of cost, time and resources associated with gathering data from entire populations. In addition, the lack of capable computational capabilities has made it unimaginable to compute gigabytes of datasets in a reasonable amount of time until now. In addition, the fact that in most cases the analysis needs to be automated and adaptive has really made it hard to perform statistical analysis using early statistical approaches. Hence the need to explore new data analytical methods such as Machine Learning.

1.4 JUSTIFICATION OF THE STUDY

Intensive digital data generating processes have created huge amounts of data which when not utilized remains an expense to organizations. This is of course as a result of costs associated with securing the data, cleaning it acquiring more storage for the data only to name a few.

Moreover, getting less accurate and less reliable results by using traditional approaches to data analysis when there exist resources that can provide more accurate and reliable results is not always something organizations would want.

In addition, there are usually expenses that are associated with hiring data analysts to do analysis on the data that could have been analyzed automatically using tools such as Machine Learning.

Adding to that, organizations are failing to realize the value that the data they have has. They are actually deprived of the limitless possibilities that big data analysis can do to transform their operations.

1.5 RESEARCH QUESTIONS

1. How to relate machine learning to the field of mathematics and or statistics?
2. How to develop a machine learning i.e. support vector machine (SVM) algorithm using concepts of vectors and optimization?
3. How to use a machine learning algorithm that is SVM for data classification?
4. How to compare SVM algorithm with other machine learning algorithms?

1.6 AIM OF THE STUDY

To convert mathematical formulations into a Machine Learning computer algorithm through coding and demonstrate how the algorithm can be used for data classification

1.7 OBJECTIVES

- To explore the relationship between mathematics and the field of machine learning
- To develop a Support Vector Machine (SVM) algorithm from mathematical formulations
- To classify data using the algorithm
- To compare the SVM algorithm against other algorithms

1.8 ASSUMPTION

- Machine learning algorithms are developed for big data analysis thus the research assumes that readers will be able to relate the algorithm developed in this study to big data analysis though it will be tested on smaller datasets.
- The research assumes that there exist a dataset that requires binary classification.

1.9 DELIMITATION

- Support Vector Machine is one of the complex and or complicated algorithm in the field of Machine Learning.
- Support Vector Machine takes more processing power during the training of the algorithm. However once the algorithm has been trained, classification will happen instantaneously.

- The datasets that were used as a datasets to test classification for the algorithm developed in this study are not as large as big data can be. However, they are sufficient to demonstrate how the algorithm works.

1.10 SIGNIFICANCE OF THE STUDY

The study is going to expose the relationship between mathematics and Machine Learning in a way that is intended to create interest in big data analysis thus increase number of participants of mathematics and or statistics majors in big data analysis.

In addition, the algorithm that will be developed will be used to classify all forms of data that need binary classifications. It is going to enable the researcher (or anyone interested in using the algorithm) to classify cancer data, iris flowers amongst many other forms of data where classification is required for example whether someone will repay a loan or not, a student will pass or not and so on.

It is going to demonstrate how organizations can benefit from the data they have in making sound decisions thus encouraging them to take advantage of machine learning in creating knowledge from their data.

1.11 DEFINITION OF TERMS

1. **Big data** - extremely large data sets that may be analyzed computationally to reveal patterns, trends, and associations, especially relating to human behavior and interactions.
2. **Machine Learning** - The use of algorithms to analyze and or create knowledge from data
3. **Artificial intelligence** - the theory and development of computer systems able to perform tasks that normally require human intelligence.
4. **Algorithm** - a process or set of rules to be followed in calculations or other problem-solving operations, especially by a computer.
5. **Data Science** - A field of Big Data which seeks to provide meaningful information from large amounts of complex data.
6. **Supervised learning** - Techniques used to learn the relationship between independent attributes and a designated dependent attribute (the label).
7. **Unsupervised learning** - Learning techniques that group instances without a pre-specified dependent attribute. Clustering algorithms are usually unsupervised.
8. **Accuracy** - The rate of correct (incorrect) predictions made by the model over a data set (cf. coverage).
9. **Support Vector Machines –** it is a discriminative classifier formally defined by a separating hyperplane. In other words, given labeled training data (supervised learning), the algorithm outputs an optimal hyperplane which categorizes new examples.
10. **Feature** - The specification of an attribute and its value.

11. **Accuracy** - The rate of correct predictions made by the model over a data set.
12. **Attribute** - A quantity describing an instance.
13. **Classifier -** A mapping from unlabeled instances to (discrete) classes.
14. **Confusion matrix -** A matrix showing the predicted and actual classifications.
15. **Cross-Validation -** A method for estimating the accuracy (or error) of an inducer by dividing the data into *k* mutually exclusive subsets (the ``folds'') of approximately equal size.
16. **Dataset -** A schemas and a set of instances matching the schema.
17. **Feature Vector -** A list of features describing an instance.
18. **Instance -** A single object of the world from which a model will be learned, or on which a model will be used (e.g., for prediction).

CHAPTER 2

LITERATURE REVIEW

2.1 INTRODUCTION

Big data has become a topic an internet user can hardly ignore. Starting from adverts on the web and most mobile applications to those that pop up on web pages in an irritating manner, it is more like the world is now run by big data technologies. For computer programmers and mathematics majors, you do not only experience it or see adverts about other things on the internet; you get to see adverts about big data itself. (Shah 2016; Shmueli 2010; Breiman 2001; Wesserman, 2013) discussed and debated over issues like, 'should machine learning take over traditional statistics' and about the future of statistical modelling. This is what has really motivated the researcher to explore Machine Learning.

This chapter will evaluate all the literature related to the study area in an attempt to show and demonstrate that we actually have big data and we have not reached a point where we can analyze all of the data we have. It will as well relate mathematics to computer science. In addition, it will show literature related to machine learning as well as Support Vector Machines (SVM). The study will explain that these topics are not new while at the same time demonstrating the gaps that are being left behind by the growth of data and our limited capabilities to analyze it.

2.2 BIG DATA

Big data refers to large, dynamic, and disparate volumes of data that is being created by people, machines and tools. It is a type of data that requires new scalable and innovative technology to host, collect and process analytically the huge amounts of information or data in order to derive

real time insights that relates to human behaviors, consumers, performance, productivity management and improved shareholder value (EY, 2014)

It is also referred to as information that is gathered from internet-enabled devices (such as smartphones and tablet), social media, machine data, video, voice recordings, DNA, weather as well as continued logging and preservation of structured and unstructured data (MGI, 2011) . It is characterized by the four Vs which as explained below:

Variety: data comes from varied sources and is being generated and or created by people and or machines

Velocity: data is being generated exponentially in an extremely fast way. The generation is non happens continually

Volume: the amount of data being generated and or created is vast as compared to traditional data sources

Veracity: big data is source comes from varied places, as a result it needs to be tested for veracity or quality

Figure 2.1 below summarizes the 4Vs

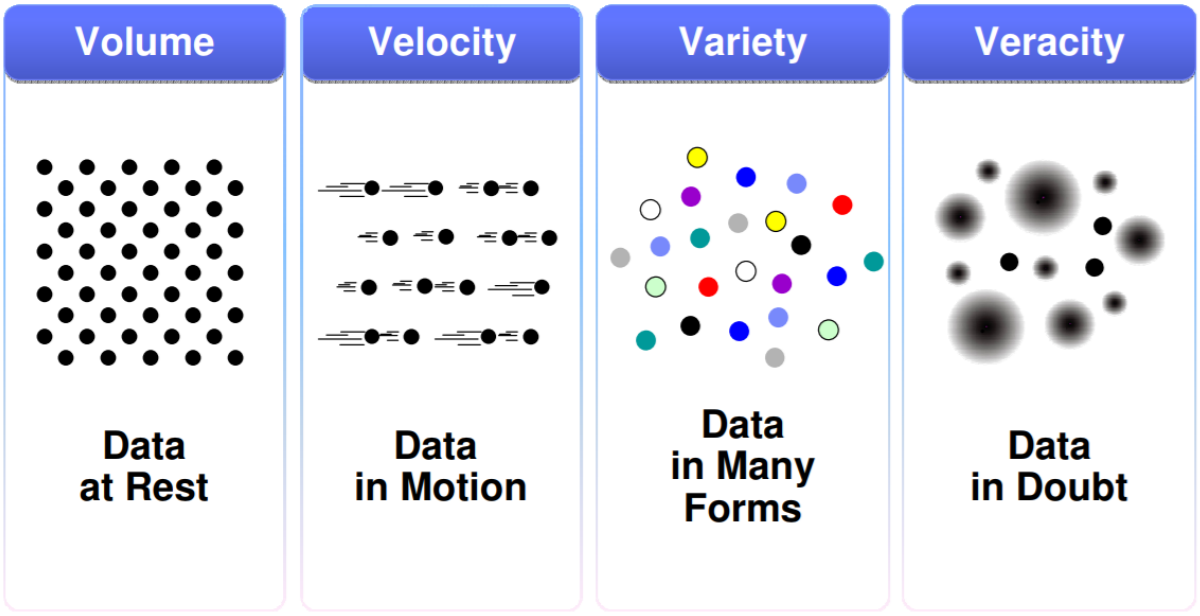


Figure 2.1: 4Vs of big data (IBM, 2012)

Lin, Genest, Banks, Molenberghs & Wang (eds., 2013) noted that data has become very critical to all aspects of human life over the course of the past 30 years. It has changed how we are entertained and educated. It informs the way we experience business, people and the wider world around us. It is evident that data has become the lifeblood of our rapidly growing digital existence. What is now currently available and to come is virtually limitless (Reinsel, Gantz & Rydning, 2017).

2.2.1 EXPONENTIAL INCREASE OF DATA

The amount of data the world is generating is exploding. Organizations and or companies are storing trillions of bytes of information related to their suppliers, customers and operators. There are millions of networked sensors that are embedded in our physical world in devices such as automobiles and mobile phones that create, stores and communicate data. Individuals and multimedia with the use of smartphones and or computers are as well fueling up the exponential growth of data. Big data has become part of every sector and function of the global economy. It is

now more like modern economic activities and or innovations cannot succeed without big data (McKinsey, 2011).

This digital existence as defined by the sum of all data that has been created, replicated and captured at any given time is growing exponentially, and it is well known as “global data sphere”. In the past 10 years, the world has witnesses the transition of analog to digital. What is now currently available and to come is virtually limitless (Reinsel, Gantz & Rydning, 2017).

Illustration of how data has grown over the past years is shown and projections of growth in the following years is shown in fig 2.2 below:

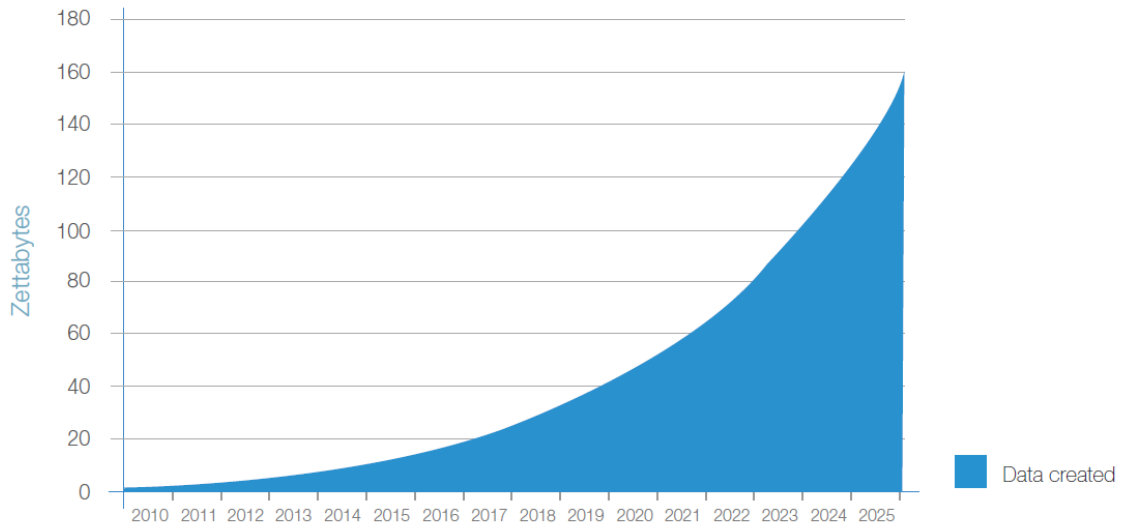


Fig 2.2 Annual Size of the Global Data sphere (IDC, 2017)

2.2.2 0.5% OF DATA ONLY BEING ANALYZED

Of all the data that has been created and continues to be created as you read this document, only a small fraction of it has been explored for analytical value (Rossi, 2015). According to (Guess,

2015) only 0.5% of global data was being analyzed as of 2015. (IDC, 2012) also estimated that by 2020, only 33% of the data will contain information that if analyzed will be inexpensively valuable.

This is because of limited IT budgets and the dearth of skilled resources that has impeded big data and analytics initiatives across organizations for example in South Africa (IDC, 2017). According to (Rossi, 2015), the slow adoption of big data analytics in schools and or colleges have really created a great scarcity in resources available to help utilize big data that is now available to organizations to provide them with unprecedented insights. This has made it hard for organizations to make better decisions and wiser investments based on the data that is at their disposal. Hence, there is a great need for technologies such as machine learning to be explored.

2.3 MACHINE LEARNING (ML)

According to (eds. Michie, Spiegelhalter & Taylor, 1994) Machine Learning refers to the automated identification of patterns in data. It has been a fertile ground for new statistical and algorithmic developments (Smola & Vishwanathan, 2008). According to (Richert & Coelho, 2013), it is the study of computer algorithms that improve automatically through experience. Machine has produced Applications ranging from data mining programs that discover general rules in large data sets, to information filtering systems that automatically learn users' interests (Mitchel, 1997).

According to (Mitchel 1997, 2), “A computer program is said to learn from experience ‘E’ with respect to some class of tasks ‘T’ and performance measure ‘P’, if its performance at tasks in ‘T’, as measured by ‘P’, improves with experience”.

Machine learning has been used in so many and varied ways. There are a number of cases where mathematical modelling has been used to create ML algorithms that does many things for example: auto correction in computers and or mobile devices, google page ranking, Netflix movies suggestions, credit card fraud detection, stock trading systems, Climate modeling and weather forecasting, Facial recognition, self-driving cars and so forth (Lin et al., 2013).

2.3.1 THE ORIGINS OF ML

With the data becoming widely accessible and in huge datasets, Artificial Intelligence (AI) which is a subset of computers science that deals with the simulation of human intelligence processes in computer systems through learning reasoning and self-correction (Efron, 1991), started incorporating statistical and or mathematical modelling to create knowledge from these huge datasets (Norvig & Russel, 1994). Attempts were made to approach big data with various symbolic methods as well as what is known as “neural networks”. According to (Marr, 2016), these were perceptrons and a number of models which were later found to be the reinventions of Generalized Linear Models (GLM) of the field of statistics. Reasoning based on probability was also employed especially in automating medical diagnostics (datascienceassn.org, nd.).

As time went by with more incorporation of mathematical and or statistical modelling, Machine Learning started realizing that it could do more than what AI was meant for, Thus it became a field on its own (Paulson, 2010)

2.3.2 MACHINE LEARNING VS STATISTICAL MODELLING

Machine learning and statistics are both fields, which focus on creating knowledge from data (Jose, 2016). Machine Learning is all about predictions, supervised learning, and unsupervised learning, but statistics is about sample, population, and hypotheses (Wesserman, 2013). According to Wesserman (2013), Machine Learning is a subfield of Artificial Intelligence which itself is a subfield of computer science while statistical modelling is a subfield of mathematics.

2.3.2.1 TYPES OF DATA THEY DEAL WITH

According to (Srivastava, 2015), Machine Learning tools are capable of learning from trillions of observations one by one. He added that they take advantage of huge datasets that are available and the ability to access them and respond to them in an instance. They make predictions while they learn patterns from data. They have algorithms such as Gradient Boosting (GB) and Random Forest (RF) which are extremely fast in handling big data. ML works very well with data with a wide range of attributes which is as well deep (having many observations). However, statistical modelling is generally used and or applied to smaller data sets with less attributes else they will end up overfitting (Srivastava, 2015).

2.3.2.2 NAMING CONVENTION

Naming for similar operations differs from one field to the other. Table 2.1 illustrates this:

Table 2.1: Machine Learning vs Statistics Naming Convention

Statistics	**Machine Learning**
Estimation	Learning
Classifier	Hypothesis
Data Point	Example/ Instance
Regression	Supervised Learning
Classification	Supervised Learning
Fitting	Learning
Parameters	Weights
Model	Network, graphs
Parameters	Weights
Fitting	Learning
Test set perfomance	generalisation
Regression/classification	Supervised learning
Density estimation clustering	Unsupervised learning

Source (Vidhya, 2015)

2.3.2.2 PREDICTIVE POWER AND HUMAN EFFORT

The lesser the assumptions made in predictive modelling the higher the capabilities of a model to make predictions. (Wassermann, 2010). Machine learning as suggested by its name has minimal human effort. It works on iterations (definition by recursion in mathematics) to find patterns in

data. Since ML does this work on comprehensive data (huge data sets) and independent of any assumption, it has a strong prediction power. On the other hand, statistical models are intense in mathematics and are based on coefficient estimation. They require the modeler (a human being) to have an understanding of the relationship between the variables before using them in a model (Srivastava, 2015)

2.3.3 SUMMARY

It may seem as if ML and statistical modelling are two different branches of knowledge, they are almost similar. The differences between these fields have gone down significantly over the past years. Both of these branches have learned many things from each other and hopes are they will come even closer in the near future (Shah, 2016). It is of no doubt that mathematics and or statistics majors require knowledge of Machine Learning. "Machine Learning offers a plethora of new research areas, new applications areas and new colleagues to work with. Our students now compete with Machine Learning students for jobs. I am optimistic that visionary Statistics departments will embrace this emerging field; those that ignore or eschew Machine Learning do so at their own risk and may find themselves in the rubble of an outdated, antiquated field ". (Wassermann, 2013, 525)

2.4 SUPPORT VECTOR MACHINES (SVM)

SVM is a Machine Learning method derived from mathematical formulations. It is used for pattern recognition and data analysis. It was invented by Vladimir Vapnik of which the current standard incarnation was proposed by both Vladimir Vapnik and Corinna Cortes (Stephanidis & Streitz, 2013). It is a discriminative classifier which is defined by a decision plane or separating hyperplane. This decision plane is what the method then used to classify new examples (Hastie & Zhu, 2006).

According to (Evgeniou, Pontil & Poggio, 2000), the main objective in SVM is to finding the best splitting boundary between data. In a two dimensional space, it can be defined as a line that separates one class from another. SVM works on a vector space, thus the line, which separates data, is in actual sense a hyperplane. The best separating decision boundary or line is the line that contains the widest margin between support vectors of classes in a dataset (Zhong & Fukushima, 2005).

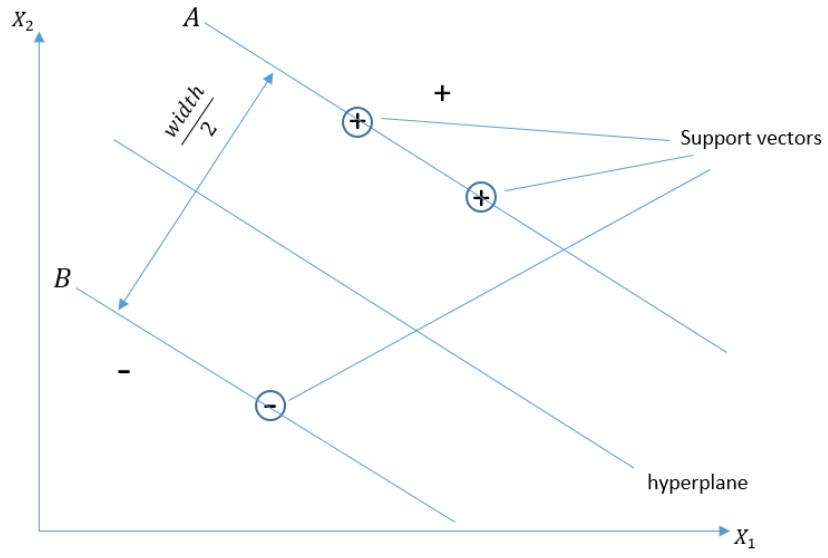


Figure 2.3: Support Vector Machine Classification (Hastie & Zhu, 2006).

The first thing that is needed here is to first find support vectors. Once support vectors have been found, the next thing is to create lines that are separated between each other at a maximum distance. Once support vectors have been found, one would create lines that are maximally separated from each other marking our support vectors. From here, It becomes easy for us to the decision boundary to be found. The total width (w) that marks our support vectors will be divided by 2.

Now if there exist a point to the left of the hyperplane, it would be classified that as black and the one on the other side as red. As shown in fig Fig 2.3

This is how linearly seperable data is classified and or seperated. However in the real world data is rarely linearly seperable. According to (Krebs, 2007), this would require the use of kennels to help in the creation of the decision boundary or hyperplane and kennels are going to be looked at later.

2.4.1 ASSERTIONS BEING MADE TO SVM

A number of assertions are made on SVM. Understanding these constraints is integral to the understanding of the mathematics that will be used in the development of the algorithm in this study.

The first thing that we will look at is the objective of the SVM. The idea here is all centered on taking data that is known and make the SVM training and or fitment our optimization problem that seeks to find the best separating line for the data. The data will be input as a training feature set into the algorithm with its labels or classes (Berwick, 2001) as illustrated in Figure 2.3.

Fig 2.3 shows a Support Vector machine classification in two dimensional space that is why the decision plane is a simple straight line. If an unknown data point is to be put as a prediction feature in the SVM algorithm, the algorithm will check which side of the decision plane the feature is and then make a classification (Evgeniou, Pontil & Poggio, 2000). Here we have a simple line which the classification could be easily solved used linear algebra using equation of a line: $y = mx + c$.

However, there are cases where we might have more than two features (attributes or variables) for example 23 features. This would translate to 23 Dimensions. This means that we need a method that will work in both 2 dimensions and many other dimensions. We now move to a vector space where we now have an unknown feature set denoted by vector $\boldsymbol{u}$, where $\boldsymbol{u} = (\boldsymbol{u_1}, \boldsymbol{u_1}, \ldots \boldsymbol{u_n})$. We now have yet another vector that is perpendicular to the decision boundary denoted by $\boldsymbol{w}$, where $\boldsymbol{w} = (\boldsymbol{w_1}, \boldsymbol{w_2}, \ldots \boldsymbol{w_n})$.The ultimate goal here is to project vector $\boldsymbol{u}$ onto vector $\boldsymbol{w}$ with some bias *b* and then see if the value is above or below zero thus helping us classify new data points. Fig 2.4 illustrates this point.

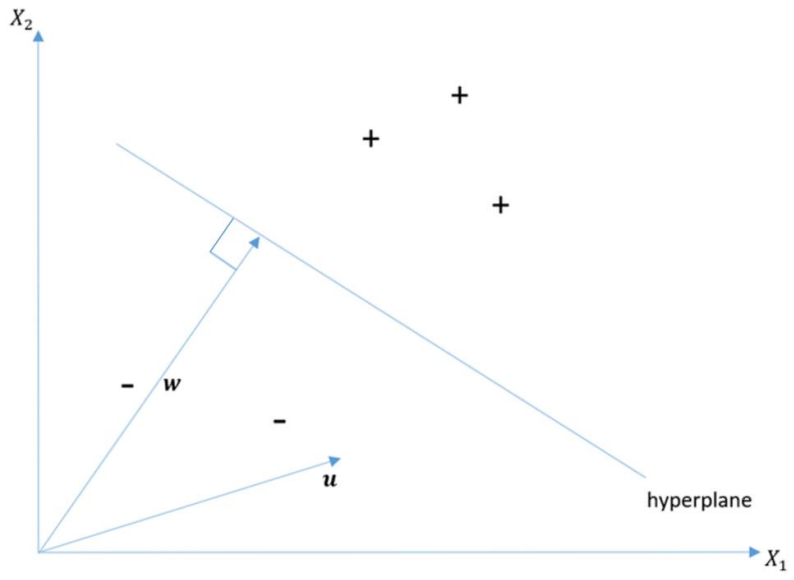


Figure 2.4: Vector $\boldsymbol{w}$ projected onto the hyperplane at a right angle

Now we can project our vector **u** to **w,** add a bias 'b' and then see if any value is above or below 0 (positive or negative) for our classes.

- $\boldsymbol{u}.\boldsymbol{w} + b < 0$ (negative class)
- $\boldsymbol{u}.\boldsymbol{w} + b > 0$ (positive class)

- $\boldsymbol{u}.\boldsymbol{w} + b = 0$ (hyperplane)

Where $\boldsymbol{u}.\boldsymbol{w}$ is the dot product.

The vector $\boldsymbol{u}$ is comprised of X_1 and X_2 but we are yet to find what $\boldsymbol{w}$ and b are comprised of yet. There are infinite possible values of $\boldsymbol{w}$ and b which can satisfy our equation. However, we have a constraint which requires us to get the best separating hyperplane which is the one that has the most width between the data that it separates. This then leads us to the optimization of $\boldsymbol{w}$ and b.

As shown in Figure 2.4, our decision plane goes in between the hyperplane that marks our support vectors $\boldsymbol{x}_{-sv}$ and $\boldsymbol{x}_{+sv}$. Since these support vectors have an impact, constraint values are set on them as illustrated below.

- $\boldsymbol{x}_{-sv}.\boldsymbol{w} + b = -1$ (negative support vector hyperplane)
- $\boldsymbol{x}_{+sv}.\boldsymbol{w} + b = 1$ (positive support vector hyperplane)

Now introducing a class of features Y_i where

$$Y_i = \begin{cases} -1, & -ve\ class \\ 1, & +ve\ class \end{cases} \qquad (*)$$

Now for the positive class ($Y_i = 1$) we have

$$\boldsymbol{x_i}.\boldsymbol{w} + b = 1 \qquad (1)$$

And for the negative class ($Y_i = -1$) we have

$$\boldsymbol{x_i}.\boldsymbol{w} + b = -1 \qquad (2)$$

Multiplying (1) and (2) with Y_i we have

from (1) $Y_i(\boldsymbol{x_i}.\boldsymbol{w} + b) = Y_i(1)$

But $Y_i = 1$ for the $+ve\ class$, so

$Y_i(\boldsymbol{x_i}.\boldsymbol{w} + b) = 1$

$Y_i(\boldsymbol{x_i}.\boldsymbol{w} + b) - 1 = 0 \qquad (3)$

from (2) $Y_i(\boldsymbol{x_i}.\boldsymbol{w} + b) = Y_i(-1)$

But $Y_i = -1$ for & $- ve\ class$

$Y_i(\boldsymbol{x_i}.\boldsymbol{w} + b) = 1$

$Y_i(\boldsymbol{x_i}.\boldsymbol{w} + b) - 1 = 0 \qquad (4)$

$(3) = (4)$

$Y_i(\boldsymbol{x_i}.\boldsymbol{w} + b) - 1 = 0$ for both the $- ve$ and $+ve$ classes

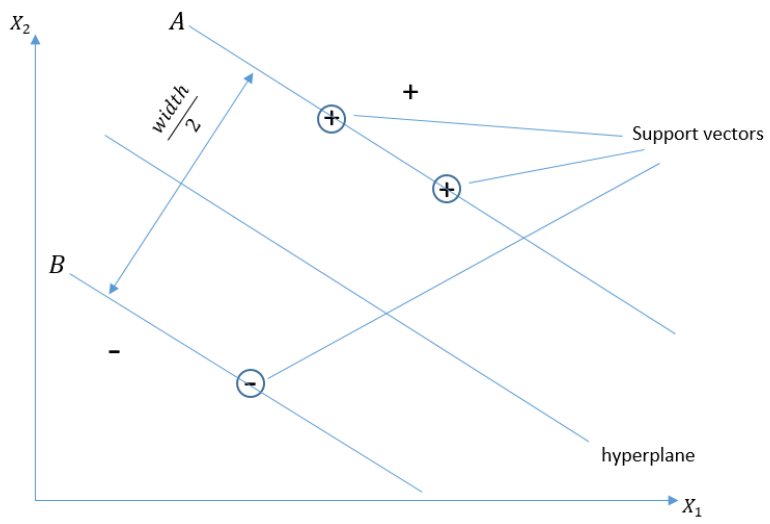


Fig 2.5: The optimization problem

The hyperplane is constructed from $\frac{width}{2}$ as shown in Fig 2.5. So we need to maximize $\frac{width}{2}$

Now $width = (\boldsymbol{X}_+ . \boldsymbol{X}_-) \cdot \frac{\boldsymbol{w}}{|\boldsymbol{w}|}$ where $\boldsymbol{X}_-$ and $\boldsymbol{X}_+$ represent positive and negative support vectors respectively

$Y_i(\boldsymbol{x_i}.\boldsymbol{w} + b) - 1 = 0$

Algebraic operations will lead to

$\boldsymbol{x_{+sv}} = 1 - b$ and $\boldsymbol{x_{-sv}} = 1 + b$

Eventually we get $width = \frac{\boldsymbol{2}}{|\boldsymbol{w}|}$ which is the function that we need to maximize

We want to minimize $|\boldsymbol{w}|$ or $\frac{\boldsymbol{1}}{2}|\boldsymbol{w}|^2$, using Lagrange multipliers

With constraint $Y_i(\boldsymbol{x_i}.\boldsymbol{w} + b) - 1 = 0$

Which all leads to

$$L(w,b) = \frac{1}{2}|w|^2 - \sum_{k=0}^{n} \alpha_i \quad \text{where } \alpha_i = (\alpha_1, \alpha_2, \dots \alpha_n)$$

$$L(w,b) = \frac{1}{2}|w|^2 - \sum_{k=0}^{n} \alpha_i \quad [Y_i(\boldsymbol{x_i}.\boldsymbol{w} + b) - 1] \, , \alpha_i = (\alpha_1, \alpha_2, \dots, \alpha_n)$$

Where $|\boldsymbol{w}|$ needs to be minimized while b is to be maximized

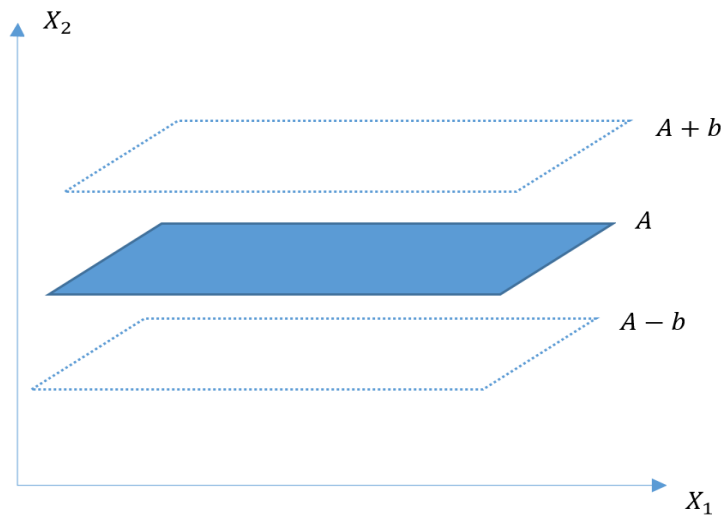


Fig 2.6: Effects of changes to bias b

Now $\boldsymbol{x_i}.\boldsymbol{w} + b$ is more or less like $mx + c$ where c is the y-intercept. Modification of b moves the plane up or down as shown in fig 2.4.1.4

Back to our Lagrange, we have

$$\frac{\partial L}{\partial w} = w - \sum \alpha_i \, Y_i \, X_i$$

$$\frac{\partial L}{\partial b} = \sum_{k=0}^{n} \alpha_i Y_i = 0$$

$$L = \sum_i \alpha_i - \frac{1}{2} \sum_{ij}^{n} \alpha_i \alpha_j Y_i Y_j \cdot (X_i \cdot X_j)$$ (which is a quadratic problem)

2.4.2 KERNELS

In cases where data is not linearly separable, SVM uses kernels, which are functions that take input vectors in a vector space and returns the dot product of those vectors (Krebs, 2007). If there exist some data, then $\mathbf{x}, \mathbf{z} \in X$ and a mapping

$$\phi : X \to \Re^N$$

Then

$$k(\mathbf{x}, \mathbf{z}) = \langle \phi(\mathbf{x}), \phi(\mathbf{z}) \rangle$$

is a kernel function (Krebs, 2007).

The main idea here is to map data on a feature space into high dimensional vector space which can be linearly separable. In this case, a hyperplane will then be used to create a boundary that separates one class of data from another. This is as illustrated in figure 2.7 below.

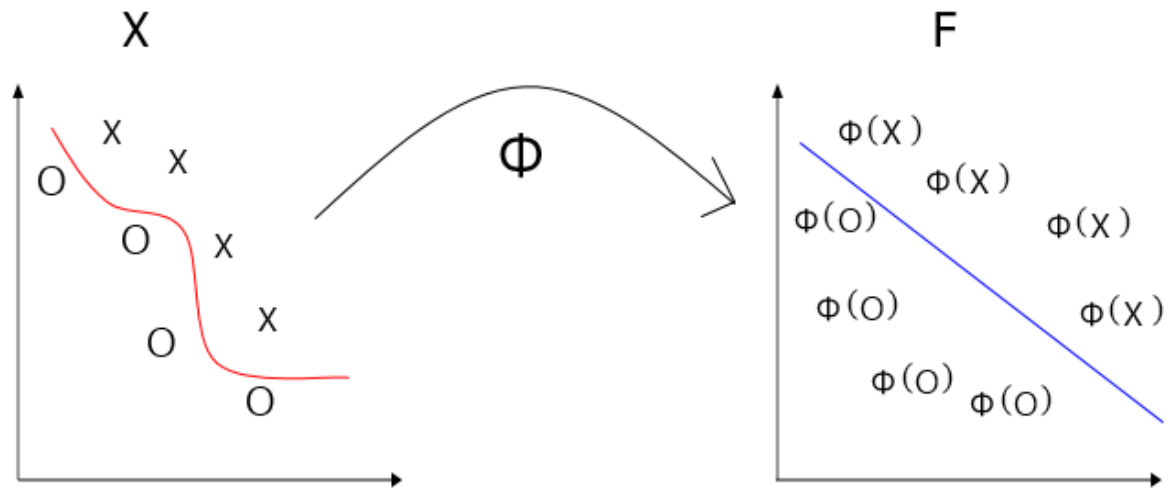


Figure 2.7: Mapping **X** to higher dimensions Vector space **F** where data is linearly separable (Krebs, 2007)

Consider a two-dimensional input space

$$X \subseteq \Re^2$$

which has a feature map that is defined by:

$$\phi : \mathbf{x} = (x_1, x_2) \mapsto \phi(\mathbf{x}) = (x_1^2, x_2^2, \sqrt{2}x_1x_2) \in F = \Re^3$$

Considering the inner product in the feature space, we have:

$$\begin{aligned} \langle \phi(\mathbf{x}), \phi(\mathbf{z}) \rangle &= \langle (x_1^2, x_2^2, \sqrt{2}x_1x_2), (z_1^2, z_2^2, \sqrt{2}z_1z_2) \rangle \\ &= x_1^2z_1^2 + x_2^2z_2^2 + 2x_1x_2z_1z_2 \ = (x_1z_1 + x_2z_2)^2 \\ &= \langle \mathbf{x}, \mathbf{z} \rangle^2 \end{aligned}$$

A good example of such mapping is as shown below in Figure 2.4.2.1

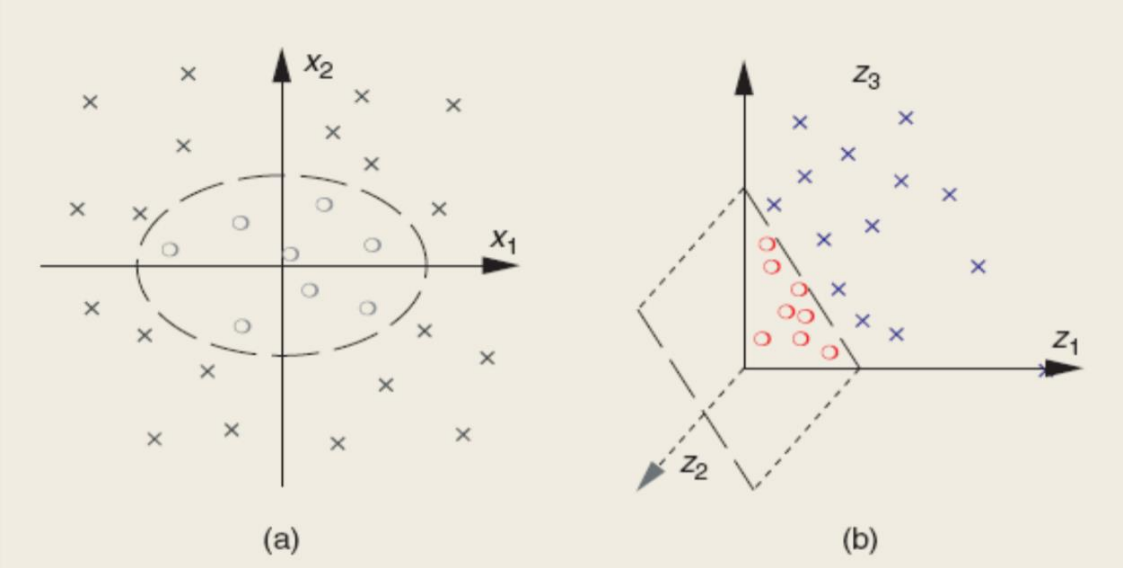


Figure 2.8: Kernel Mapping (Krebs, 2007)

Figure 2.8 shows effect of mapping

$$\phi(x_1, x_2) = (x_1^2, \sqrt{2}x_1x_2, x_2^2)$$

Where (a) is input space ***X*** and (b) is feature space ***H***. Now

$$k(\mathbf{x}, \mathbf{z}) = \langle \mathbf{x}, \mathbf{z} \rangle^2$$

But $k(\mathbf{x},\mathbf{z})$ is as well a kernel that computes inner product of the mapping

$$\psi(\mathbf{x}) = (x_1^2, x_2^2, x_1x_2, x_2x_1) \in F = \Re^4$$

Which shows that a given kernel function is not unique to a given feature space.

2.4.2.1 APPLICATION OF KERNELS TO SUPPORT VECTORS

The solution of a dual problem gives:

$$\hat{\mathbf{w}} = \sum_{i=1}^{n} \hat{\alpha}_i \, y_i \mathbf{x}_i$$

Decision boundary is given by:

$$\hat{\mathbf{w}}^T \mathbf{x} + w_0 = \sum_{i \in SV} \hat{\alpha}_i \, y_i (\mathbf{x}_i^T \mathbf{x}) + w_0$$

The decision is given by:

$$\hat{y} = \text{sign} \left[\sum_{i \in SV} \hat{\alpha}_i \, y_i (\mathbf{x}_i^T \mathbf{x}) + w_0 \right]$$

Decision for mapping to a feature space is given by:

$$\hat{y} = \text{sign} \left[\sum_{i \in SV} \hat{\alpha}_i \, y_i (\phi(\mathbf{x}) \cdot \phi(\mathbf{x}_i)) + w_0 \right]$$

2.5 CONVEX OPTIMIZATION (CVXOPT)

CVXOPT is an open source package that does convex optimization in the python programming environment. It makes use of kernels to perform optimization on nonlinearly separable data (CVXOPT, 2014). Its main purpose is making development of algorithms and or softwares for convex optimization an easy task.

2.5 PYTHON

Python is a general-purpose programming language. It can be used to develop just about anything. Professionally, it is very good for data analysis, backend web development, artificial intelligence, Machine Learning and scientific computing. It has gained more acceptance with both data scientists and scientist communities (Cesarani, 2016).

Python is easy to understand, flexible, scalable and has a very big community. It has been used as introductory programming languages for science and data analysis in a number of universities especially in the United States of America (Guo, 2014).

2.7 GRADIENT DESCENT

Gradient descent is an iterative optimization method for finding the minimum of a function, It is a very popular method in the field of machine learning since Machine Learning is concerned with highest accuracy or minimization of error rate given set of training data (Gordon, 2010). It is used to find the minimum error by minimizing a set cost function.

If we have,

$f : \mathbb{R}^n \to \mathbb{R}$

which is differentiable and convex and we want to solve

$$\min_{x \in \mathbb{R}^n} f(x),$$

i.e., finding x* such that f(x*) = min f(x)

Gradient descent:

choose initial

$x^{(0)} \in \mathbb{R}^n$

Repeat

$$x^{(k)} = x^{(k-1)} - t_k \cdot \nabla f(x^{(k-1)}), \quad k = 1, 2, 3, \ldots$$

Stop at some point. At each iteration, consider the expansion

$$f(y) \approx f(x) + \nabla f(x)^T (y - x) + \frac{1}{2t}\|y - x\|^2$$

Quadratic approximation. Replacing

$\nabla^2 f(x)$ by $\frac{1}{t} I$

$$f(x) + \nabla f(x)^T (y - x)$$

Which is linear approximation of *f*

$$\frac{1}{2t}\|y - x\|^2$$

Proximity term to *x* with weight 1/(2*t*)

Choosing the next point

$$y = x^+$$

to maximize the quadratic approximation

$$x^+ = x - t\nabla f(x)$$

CHAPTER 3

3.1 INTRODUCTION

The purpose of this chapter is to give an investigator more if not enough information to replicate our study. It is going to be structured in such a way that it becomes easy for the reader to identify all the parts of the research and how they work together to address the central research question in the study. Moreover it is going to look at the type of data that has been used, model development, algorithm training as well as classification. Mathematical and theoretical aspects of the SVM has already been explained in Chapter 3 of the study. Here, the researcher delved deep into the programming implementation of the mathematical formulation shown in the chapter in context. At the end of the development process, the algorithm will then be used for data classification.

It is of prime importance to note that this research is primarily focused on demonstrating how machine-learning algorithms are developed from mathematical formulations as a way of demystifying machine learning and encouraging mathematics and or statistics students to adopt the technology thus helping in closing the gap between the growth of data and our capacity to analyze it. It will not be enough if we end on demystifying ML, the algorithm will as well be evaluated to see its performance and then compared to other algorithms in Chapter 4 of this study.

3.2 DATA TYPE

The type of data that is being used to test the algorithm created in this study are the iris and breast cancer dataset.

3.2.1 IRIS DATASET

"The Iris dataset is a dataset from the 1930s and is one of the first modern examples of statistical classification" (Richert & Coelho, 2013). It is perhaps one of the best known datasets to be found in the pattern recognition literature.

The dataset has been referenced by a number of authors (Duda & Hart 1973; Gates, 1972) and cited in a number of papers (Zhong & Fukushima, 2005; Kotsiantis & Pintelas, 2005; Dy & Broodley, 2004) only to name a few.

Here we have Iris flowers which have multiple species that can be identified by their morphology. Today the genomic signatures now defines the species. However in the 1930s before DNA had been identified (Richert & Coelho, 2013). The following attributes determines the classification of the species as either:

- Sepal length
- Sepal width
- Petal length
- Petal width

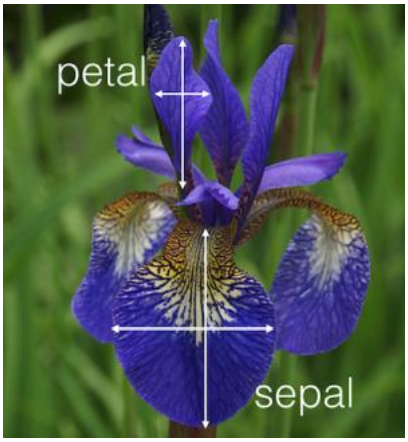


Fig 3.1: Iris Flower (UCI, n.d.a)

The measurements from our data will be referred to as features. For each plant, the species is recorded. The question now will be, given measurements of a plant can we make a good classification based on the measurements? This is a supervised learning or classification problem. Which means that given labeled examples, we can then come up with a rule that will then be applied to other examples. (Richert & Coelho, 2013)

The dataset that will be used for classification of Iris flowers is an open data dataset that was created by R.A. Fisher and was uploaded on UCI Machine learning repository. Its details are as follows:

Table 3.1: Iris dataset Summary

class	Instances	Missing Values
Iris Setosa	50	Non
Iris Vericolor	50	Non
Irisa Virginica	50	Non

source: (UCI, n.d.a)

Table 3.2: Iris Dataset Attributes

#	Attribute	Unit of measurement	values
1	Sepal length	Cm	float
2	Sepal width	Cm	Float
3	Petal length	Cm	Float
4	Petal width	Cm	Float
5	Class	cm	Iris Setosa
			Iris Vericolour
			Iris Virginica

Source: source: (UCI, n.d.a)

The data set contains 3 classes that has 50 instances each. Each class refers to a type of a plant. One class (Iris Setosa) is linearly separable from the other 2 classes (Iris Vericoloyr and Iris Virginica); the latter (Iris Vericolour and Iris Virginica) are NOT linearly separable from each other.

3.2.2 BREAST CANCER DATASET

This dataset is an open data dataset available at Machine Learning Repository. The dataset was created by was created by Wolberg (1991) from University of Wisconsin Hospitals in USA. Samples arrived periodically while they were recorded and added to the dataset. The dataset has 699 records as shown in Table 2.8.1.

Table 3.3 Breast Cancer Dataset samples

Group	Instance	Date Added
1	367	January 1989
2	70	October 1989
3	31	February 1980
4	17	April 1990
5	48	August 1990
6	49	Updated January 1991
7	31	June 1991
8	86	November 1991
Total	699	As of donated database on15 July 1992

Source (UCI, n.d.b)

The breast cancer dataset contains records about breast tumors which are classified as either benign (non-cancerous) or malignant (cancerous). The features in the dataset were computed form a digitalized image of a needle aspirate (FNI) of a breast mass. The features describes and or relates the characteristics of cell nuclei that are present in the image (UCI, n.d.b). Details about the attributes are as shown in Table 3.2.

Table 3.4 Breast Cancer Data Set Attribute Information

Attribute	Data Type	Range
Sample code number	string	
Clump Thickness	float	1-10
Uniformity of Cell Size	float	1-10
Marginal Adhesion	float	1-10
Single Epithelial Cell Size	float	1-10
Bare Nuclei	float	1-10
Bland Chromatin	float	1-10
Normal Nucleoli	float	1-10
Mitoses	float	1-10
Class	integer	2 for benign, 4 for malignant

Source: (UCI, n.d.b)

3.3 POPULATION

While demystification of machine learning by relating it to mathematical and or statistical modelling is directed at mathematics and statistics majors. It is as well meant for organizations that have big data at their disposal and have not yet started taking advantage of machine learning in deducing informed decisions from it. This is all directed at increasing participants to big data analysis thus helping in closing the gap between the growth of data and our ability to analyze it.

The algorithm that is developed in this chapter will work on any situation that requires binary classification. However, it is important to note that some parameters would need to be modified to suit new datasets.

3.4 PROCEDURE

This section discusses the procedure that was taken to meet the objectives of our study thus demystifying machine learning by relating it to mathematics while in the process creating an algorithm which will then be used to classify data.

The logical design of the algorithm is summarized in the flow chat in Figure 3.2 below:

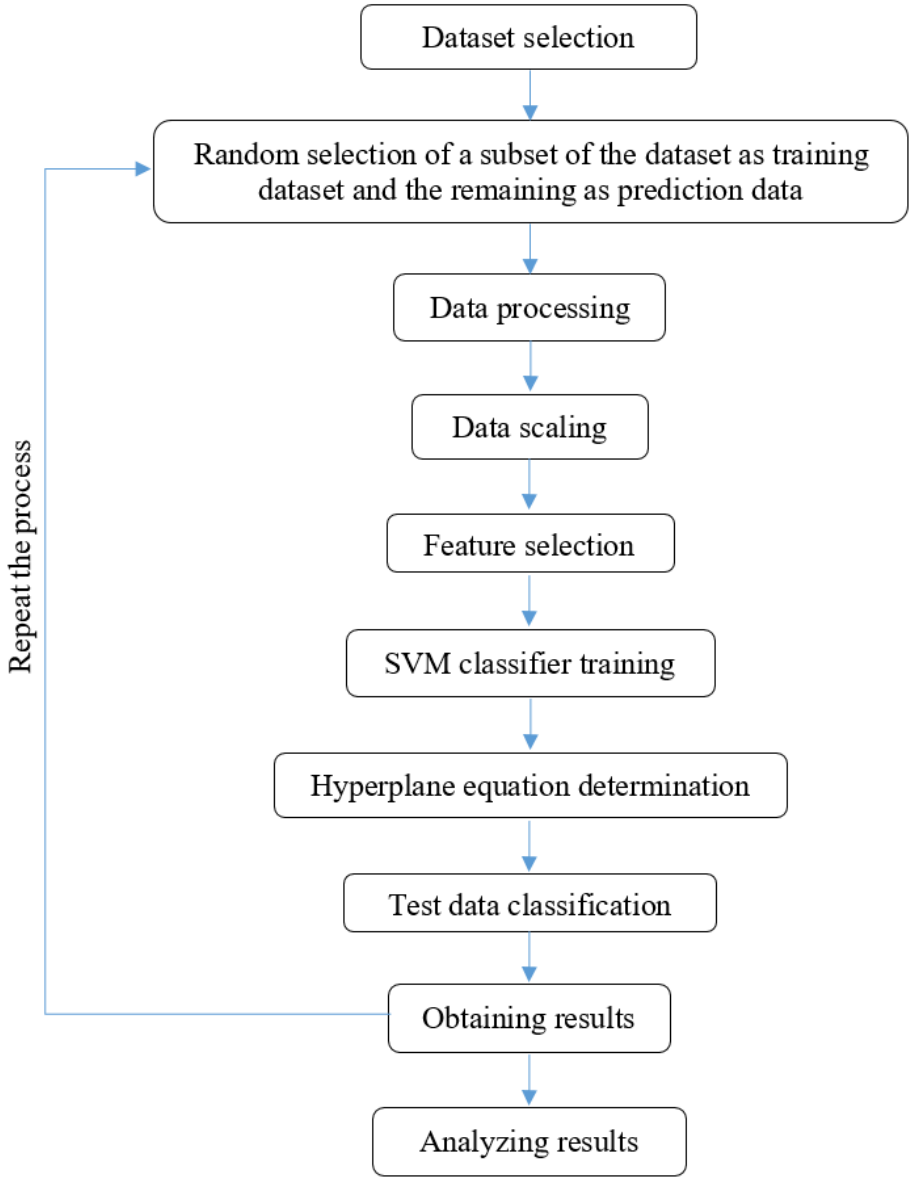


Figure 3.2 Flow chat representation of the SVM algorithm design logic

3.4.1 TOOLS USED

The algorithm in this study was built using python tools for visual studio 2017, the latest version of python, which was available on the time of writing was used for the development of the algorithm. Matplot lib was used for graphical representation of the support vector classification.

3.4.2 DEVELOPING THE ALGORITHM

Firstly, necessary imports or modules were imported such as matplotlib for graphical presentations and numpy for numeric operations or number manipulations.

After this, a data dictionary was created. This dictionary stored training data for the algorithm in their respective classes. The iris dataset was regrouped into either setosa or non-setosa(versicolor and verginica) since the researcher was interested in testing for a binary classification). Thus he ended up having 2 classes, vertosa with 50 instances and non vertosa with 100 features. One class was stored as negative class (setosa) and the other as positive class (non-setosa that is versicolor and virginica) as shown by the code below.

```
154     negative_class = []
155     positive_class = []
156
157     df = pd.read_csv("iris_dataset.csv")
158     for i, row in df.iterrows():
159
160         if row['class'] == 'Iris-setosa':
161             if count_neg <= 500: #this helps us change sample sizes for our negative class
162                 negative_class.append([float(row['sepal_length']), float(row['sepal_width'])])
163         else:
164             if count_pos <= 500: #this helps us change sample sizes for our positive class
165                 positive_class.append([float(row['sepal_length']), float(row['sepal_width'])])
166         if i == 2000: #150 this helps us change total sample size
167             print('[' + str(row['sepal_length']) + ',' + str(row['sepal_width']) + ']')
168             break
169
170     data_dictionary = {-1:np.array(negative_class),
171                        1:np.array(positive_class)}
```

Code to change sample sizes and evaluating the performance of the algorithm is shown by conditional if statements (definition by cases in mathematics) in line 161, 164 and 166

A class Named "SupportVectorMachine" was created following concepts of Object Oriented Programming (OOP). This is the class that dealt with all mathematical operations associated with the SVM. Code that visualize data was hosted in this class. Colors for the classification was as well defined here in a dictionary for example red for positive class and blue for the negative class data points as illustrated below:

```
9   class SupportVectorMachine:
10      def __init__(_self, visualization=True):
11          _self.visualization = visualization
12          _self.colors = {1:'r',-1:'b'}
13          if _self.visualization:
14              _self.fig = plt.figure()
15              _self.ax = _self.fig.add_subplot(1,1,1)
```

After this, methods that does training, prediction and fitting of the data into the model were created. The fit method was used to train the SVM algorithm. Gradient decent method was used for solving our optimization problem that is for finding the minimum value of ||**w**||.

Prediction, training and fitting methods were created. Minimum value of ||**w**|| was calculated using the gradient descend method and values of each descend stored in an optimization dictionary name "opt_dictionary". As the algorithm was stepping down the **w** vector using the gradient descent method, it tested whether the vector was still in the constraint function. It searched for the minimum $\|\boldsymbol{w}\|$ while at the same time maximum value of b that satisfy our equation $(\boldsymbol{Y_i(x_i.w + b) - 1 = 0})$ and the value was stored in the "opt dictionary" as well. The dictionary was in the key value pair form that is {||w||:[w,b]}. When the algorithm was done looping finding

the values of **w** within our constraint function, it then selected the smallest key in the "opt_dictionay" that is the one with the smallest $\|\boldsymbol{w}\|$.

After this, the algorithm then checked all possible values of **w** using transformations defined by the "transforms" object. That is if the algorithm had found (3,4) to be the value of **w** that gives minimum $\|\boldsymbol{w}\|$, other possible values of the vector **w** such as (3,-4), (-3,4), (-3,-4) will then be tested in our equation $(\boldsymbol{Y_i(x_i.w + b) - 1 = 0})$ since they all give the same magnitude of **w**.

The algorithm started the stepping from a middle value in the training dataset. It started stepping using bigger steps comparing the previous value from the preceding one. If it found that the preceding value had a lower $\|\boldsymbol{w}\|$ and still satisfies our equation $(\boldsymbol{Y_i(x_i.w + b) - 1 = 0})$ it continues until it reaches a point where it detected that the values it now started to get were becoming larger that the value of $\|\boldsymbol{w}\|$which were satisfying our equation. Here it detected that it had reached a point closer to the minimum value of $\|\boldsymbol{w}\|$ so it then changed stepping and started using smaller steps towards the minimum $\|\boldsymbol{w}\|$ until it had reached the closest point to minimum $\|\boldsymbol{w}\|$ and or $\|\boldsymbol{w}\|$ thus solving the optimization problem for the SVM.

This was all done by the following code:

```
16      # train svm
17      def fit_svm(_self, data):
18          _self.data = data
19          # ----------{ ||w||: [w,b] }-------------
20          _opt_dict = {}
21
22          _transforms = [[1,1],
23                         [1,-1],
24                         [-1,1],
25                         [-1,-1]]
```

Array of transforms (line 22 – 25)

```
27          _all_data = []
28          for yi in _self.data:
29              for featureset in _self.data[yi]:
30                  for feature in featureset:
31                      _all_data.append(feature)
32
33          _self.max_feature_value = max(_all_data)
34          _self.min_feature_value = min(_all_data)
35          _all_data = None
```

Determining range of feature set to get a starting point for gradient descent (line 27 - 35)

```
39          _step_sizes = [_self.max_feature_value * 0.1,
40                         _self.max_feature_value * 0.01,
41                         # more accurate but takes a lot of processing power
42                         _self.max_feature_value * 0.001,
43                         ]
```

Gradient descent steps for ***w*** (line 39 – 42).

```
45          # expensive iteration or recursion
46          _b_range_multiple = 2
47          # we dont have to take very small of steps of b as we do with w
48          _b_multiple = 5
49          _the_latest_optimum = _self.max_feature_value * 10
```

Gradient Descent step values for *b* (Line 45 – 49)

Line 39 - 49 are objects that were used to determine the gradient steps for **w** and *b* through definition by recursion or loops. The steps of *b* were larger to minimize the amount of time the algorithm takes on optimization while ***w*** had smaller steps since its magnitude was of prime importance in the optimization function. Here, variables that helped in making steps (using

recursion or loops) with *b* were then set as illustrated below. The steps for *b* were larger since the algorithms was mostly concerned with the values of $\boldsymbol{w}$ for the optimization problem.

The idea with the gradient descent was to step down the vector finding the minimum and maximum $\boldsymbol{w}$ which satisfies the equation ($Y_i(\boldsymbol{x_i}.\boldsymbol{w} + b) - 1 = 0$) as illustrated in figure 3.2 below. Here the algorithm started at the medium point in our dataset and then add steps starting with larger one. It compared it the output satisfied our equation and if so it continues with the steps until it reached a point where **w** was becoming bigger. It then start moving backwards using smaller steps and repeated the process until it reached the optimum value of $\boldsymbol{w}$.

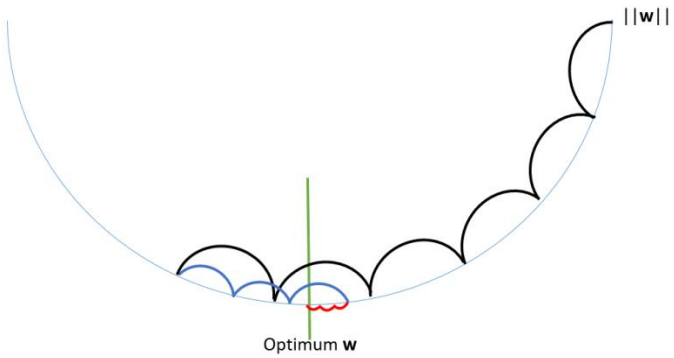


Figure 3.3: Gradient descent process taken by the SVM algorithm

Values of *b* were as well iterated through. This was done using constant steps. The size of the steps was broken down to smaller ones but the results were the same though it took more processing power making the algorithm inefficient. Hence, the value of steps for b were left as a constant value.

After iterations through each transformation and testing the values against constraint requirements. Feature sets that failed to meet the constraints in our data set were ignored. Once all the steps were

taken, the optimization dictionary was sorted and contained key pair values in the form (||***w***|| : [***w***,*b*]). This is illustrated in Lines 16 - 90 below (Blank Lines were skipped):

```
16      # train svm
17      def fit_svm(_self, data):
18          _self.data = data
19          # ----------{ ||w||: [w,b] }-------------
20          _opt_dict = {}
21          _transforms = [[1,1],
22                         [1,-1],
23                         [-1,1],
24                         [-1,-1]]
25          _all_data = []
26          for yi in _self.data:
27              for featureset in _self.data[yi]:
28                  for feature in featureset:
29                      _all_data.append(feature)
30
31          _self.max_feature_value = max(_all_data)
32          _self.min_feature_value = min(_all_data)
33          _all_data = None
34          # remember support vectors yi(xi.w+b) = 1
35          _step_sizes = [_self.max_feature_value * 0.1,
36                         _self.max_feature_value * 0.01,
37                         # more accurate but takes a lot of processing power
38                         _self.max_feature_value * 0.001,]
39          # expensive iteration or recursion
40          _b_range_multiple = 2
41          # we dont have to take very small of steps of b as we do with w
42          _b_multiple = 5
43          _the_latest_optimum = _self.max_feature_value * 10
44
45          for step in _step_sizes:
46              w = np.array([_the_latest_optimum,_the_latest_optimum])
47              # we can do this because convex
48              current_time = datetime.datetime.now()
49              is_optimized = False
50              while not is_optimized:
51                  #continue
52                  for b in np.arange(-1 * (_self.max_feature_value * _b_range_multiple),
53                                     _self.max_feature_value * _b_range_multiple,
54                                     step * _b_multiple):
55                      for transformation in _transforms:
56                          w_t = w * transformation
57                          found_option = True
58
59                          for i in _self.data:
60                              for xi in _self.data[i]:
```

```
63                      for i in _self.data:
64                          for xi in _self.data[i]:
65                              yi = i
66                              if not yi * (np.dot(w_t,xi) + b) >= 1:
67                                  found_option = False
68                      if found_option:
69                          _opt_dict[np.linalg.norm(w_t)] = [w_t,b]
70                  if w[0] < 0:
71                      is_optimized = True
72                      time_span = (datetime.datetime.now() - current_time).total_seconds() / 60.0
73                      print('Optimized a step. it took ' + str(time_span) + ' minutes')
74                  else:
75                      w = w - step
76
77              _norms = sorted([n for n in _opt_dict])
78              #formatt of our dictionary ||w|| : [w,b]
79              try:
80                  _opt_choice = _opt_dict[_norms[0]]
81                  _self.w = _opt_choice[0]
82                  _self.b = _opt_choice[1]
83                  _the_latest_optimum = _opt_choice[0][0] + step * 2
84              except BaseException as e:
85                   #logger.error('Error occured @: ' + str(e))
86                   print('Error occured @: ' + str(e))
87
88          for i in _self.data:
89              for xi in _self.data[i]:
90                  yi = i
```

Fit method (line 16 – 90)

After the training method, the prediction method was created. Inorder for the predictions to be visualized, code that handles visualization was added as well. Visualization for predictions were done by plotting stars on the respective classes for the data. Colors for the classes were set automatically to respective classes. This is illustrated in Lines 93 – 99 below:

```
93      def predict_svm(_self,features):
94          # sign( x.w+b ) determines the classification
95          classification_data = np.sign(np.dot(np.array(features),_self.w) + _self.b)
96          if classification_data != 0 and _self.visualization:
97              _self.ax.scatter(features[0], features[1], s=200, marker='*',
98                               c=_self.colors[classification_data])
99          return classification_data
```

A method for visualization of support vectors, features (training data) and the hyperplanes (for positive class, negative class and the decision boundary) was created. This was done by finding

two points from each of the negative and positive classes. Since ***w*** and *b* were found from the optimization function, algebra was used to create a function that returns a value for the second feature that was then used to create the hyperplanes. This is illustrated in Lines 101 – 103 below.

```
101     def visualize_svm(_self):
102         [[_self.ax.scatter(x[0],x[1],s=100,color=_self.colors[i])
103           for x in data_dictionary[i]] for i in data_dictionary]
```

Hyperplanes were visualized in lines 113 - 133 as shown below

```
113         def hyperplane(x,w,b,v):
114             return (-w[0] * x - b + v) / w[1]
115 
116         _datarange = (_self.min_feature_value * 0.9,_self.max_feature_value * 1.1)
117         hyperplane_min_x = _datarange[0]
118         hyperplane_max_x = _datarange[1]
119 
120         # (w.x+b) = 1 for our positive support vector hyperplane
121         positive_sv_1 = hyperplane(hyperplane_min_x, _self.w, _self.b, 1)
122         positive_sv_2 = hyperplane(hyperplane_max_x, _self.w, _self.b, 1)
123         _self.ax.plot([hyperplane_min_x,hyperplane_max_x],[positive_sv_1,positive_sv_2], 'k')
124 
125         # (w.x+b) = -1 for our negative support vector hyperplane
126         negative_sv_1 = hyperplane(hyperplane_min_x, _self.w, _self.b, -1)
127         negative_sv_2 = hyperplane(hyperplane_max_x, _self.w, _self.b, -1)
128         _self.ax.plot([hyperplane_min_x,hyperplane_max_x],[negative_sv_1,negative_sv_2], 'k')
129 
130         # (w.x+b) = 0 for our decision plane
131         db1 = hyperplane(hyperplane_min_x, _self.w, _self.b, 0)
132         db2 = hyperplane(hyperplane_max_x, _self.w, _self.b, 0)
133         _self.ax.plot([hyperplane_min_x,hyperplane_max_x],[db1,db2], 'y--')
```

3.4.3 EVALUATING ALGORITHMS

Here the researcher created models of the data and estimated their accuracy on unseen data. The following steps are the steps the researcher took to achieve this.

1. Dividing the dataset into training data and testing or prediction data.
2. Setting up the test harness to use the 10-fold cross validation method.
3. Building five different ML models to predict species of our iris dataset
4. Selecting the ML model that best suits the data.

Another python class was created and encapsulated/separated from the 'SupportVectorMachine' class following concepts of OOP.

3.4.3.1 VALIDATION DATASET

For the researcher to know if the model that was built was good, statistical methods to estimate the accuracy of the model in predicting unknown data points were used. This was done by holding back some data for algorithm training and some for prediction. The data that was used for training was used to estimate accuracy of the models.

The iris dataset was split into two groups and or classes. 80 percent of the data was used to train the models while the remaining 20 percent was used to validate the models. This is shown in lines 45 – 52 below:

```
45  # Splitting out data for validation
46  my_array = irirs_dataset.values
47  X = my_array[:,0:4]
48  Y = my_array[:,4]
49  validation_size = 0.20
50  seed = 7
51  X_train_data, X_validation_data, Y_train_data, Y_validation_data = model_selection.train_test_split(
52      X, Y, test_size=validation_size, random_state=seed)
```

3.4.3.2 TEST HARNESS

10-fold cross validation was used to estimate accuracy of the models algorithms. The iris dataset was split into 10 parts of which 9 were used as training data whilst the remaining 1 was used for training combinations of training splits. Training was done on 10 parts and testing on 1 for all combinations of the train-test splits.

3.4.3.3 BUILD MODELS

Six algorithms were evaluated. The algorithms were as follows:

- Linear Discriminant Analysis (LDA)
- Gaussian Naive Bayes (GNB)
- Logistic Regression (LR)
- K-Nearest Neighbors (KNN)
- Classification and Regression Trees (CART)
- Support Vector Machines (SVM)

The researcher decided to mix simple linear models(LDA and LR) and nonlinear models(CART,KNN,GNB and SVM). The random number seed was reset before each run to ensure that the evaluation of each algorithm was performed using the same data splits as shown from line 54 – 74 below. This ensured that the results become directly comparable.

```
54  # Testing option and evaluation metrics
55  _seed = 7
56  _scoring = 'accuracy'
57
58  _models = []
59  _models.append(('LDA', LinearDiscriminantAnalysis()))
60  _models.append(('CART', DecisionTreeClassifier()))
61  _models.append(('LR', LogisticRegression()))
62  _models.append(('NB', GaussianNB()))
63  _models.append(('KNN', KNeighborsClassifier()))
64  _models.append(('SVM', SVC()))
65  # evaluating models
66  _results = []
67  _names = []
68  for _name, _model in _models:
69      kfold = model_selection.KFold(n_splits=10, random_state=seed)
70      _cv_results = model_selection.cross_val_score(_model, X_train, Y_train, cv=kfold, scoring=_scoring)
71      _results.append(cv_results)
72      _names.append(name)
73      _message = "%s: %f (%f)" % (_name, cv_results.mean(), cv_results.std())
74      print(_message)
```

A plot of the model evaluation was created and the spread as well as the mean accuracy of the models was compared. A population of accuracy was then found from evaluating each model 10 times (10 fold cross validation) as shown in Lines 77 - 83 below:

```
77  # Comparing algorithms
78  _fig = plt.figure()
79  _fig.suptitle('Algorithm Comparison')
80  _ax = fig.add_subplot(111)
81  plt.boxplot(results)
82  _ax.set_xticklabels(_names)
83  plt.show()
```

3.4.3.4 PREDICTIONS

To find the accuracy of predictions, the researcher looked at each of the six algorithms independently. Validation set was kept so that overfitting to the training data set or a data leak will be easy to identify. This was to avoid overly optimistic results.

This gave the researcher an independent final check on the accuracy of the best model that suits the iris dataset. A validation set was kept so that in case a slip is made during training for example overfitting to the training set or data leak, it will result in optimistic results. Lines 118 – 124 below shows an example that was used to check accuracy of the KNN model and or algorithm.

```
118  #K-Nearest Neibhors
119  _knn = KNeighborsClassifier()
120  _knn.fit(X_train_data, Y_train_data)
121  _predictions = _knn.predict(X_validation)
122  print(accuracy_score(Y_validation, _predictions))
123  print(confusion_matrix(Y_validation, _predictions))
124  print(classification_report(Y_validation, _predictions))
```

3.5 DATA ANALYSIS AND INTERPRETATION

The Support Vector Machine algorithm was used to analyze the iris data. Data was graphically represented on our vector space using the algorithm created in this chapter. Descriptive statistics of the data were presented in graphs. Matplotlib module was used for displaying graphical presentations which in turn showed our support vectors, hyperplanes, data points as well as predictions and or classifications of unknown data points.

Moreover, the efficiency of the code was looked at and its accuracy in classification. The algorithm was tested by modifying properties in the optimization function and see how they affected the accuracy of our classification. These properties include gradient descent step sizes, training data size as well as the number of steps which were made on each gradient descent. In addition, classification results were then compared with other ML algorithms (mentioned in this chapter). The results of comparisons were then analyzed.

Chapter 4 will present all this with graphical representations accompanied with descriptive summaries. This is to ensure it becomes easy to follow for the readers.

3.6 SUMMARY

We have seen how mathematical computations were converted into a software algorithm. While it may look as if the code has nothing similar to mathematics (the programming syntax as opposed to mathematical symbols and formulations). It is important to know that Machine Learning algorithms are developed from mathematical and or statistical modelling; in fact, Machine Learning is all about optimization and all forms of optimization are machine learning (Bennet & Parrado-Hernandez, 2006). Hence remains a duty for mathematics and or statistics majors to engage in Machine Learning algorithm development and help close the gap between the growth of data and our capability to analyze it.

CHAPTER 4

DATA PRESENTATION AND ANALYSIS OF RESULTS

4.1 INTRODUCTION

This chapter focuses on the analysis and interpretation of results. The researcher used the algorithm that was developed in this study to together with matplotlib to generate and create graphs that will be used for analysis in this chapter. The chapter start by looking at linearly separable data analysis (the iris dataset) and then goes on to evaluate classification of non-linearly separable dataset (breast cancer dataset) using cvxopt.

4.2 DESCRIPTIVE STATISTICS OF THE IRIS DATASET

The researcher started by showing descriptive statistics about the iris dataset and then went on evaluating the algorithm using the dataset. In addition, the algorithm was then used for iris data classification and lastly evaluation of other algorithms was then done and their results compared to the SVM algorithm. Other algorithms which were evaluated on the iris data set are as follows:

- Logistic regression(LR)
- Classification and Regression Trees(CART)
- K-Nearest Neigbhors (KNN)
- Linear Discriminant Analysis (LDA)
- Gaussian Naïve Bayes (NB)

4.2.1 EXPLORATORY DATA ANALYSIS

This section shows descriptive information about the iris dataset as shown in figures below:

Table 4.1 Iris Dataset Statistical Summary

	Sepal length	Sepal width	Petal length	Petal width
count	150	150	150	150
mean	5.843333	3.054000	3.758667	1.198667
std	0.828066	0.433594	1.764420	0.763161
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

Table 4.1 is a statistical summary of the dataset. Sepal-length has the highest mean implying that generally has larger values that any other attribute in our dataset. Followed by petal length sepal width and petal width. This clearly shows that sepals and bigger than petals of the iris flower.

Table 4.2 Iris Dataset Class Distribution

class	Setosa	versicolor	virginica
size	50	50	50

Table 4.2 shows the size of each class. That is we have 50 records from each of the three classes of our iris flower.

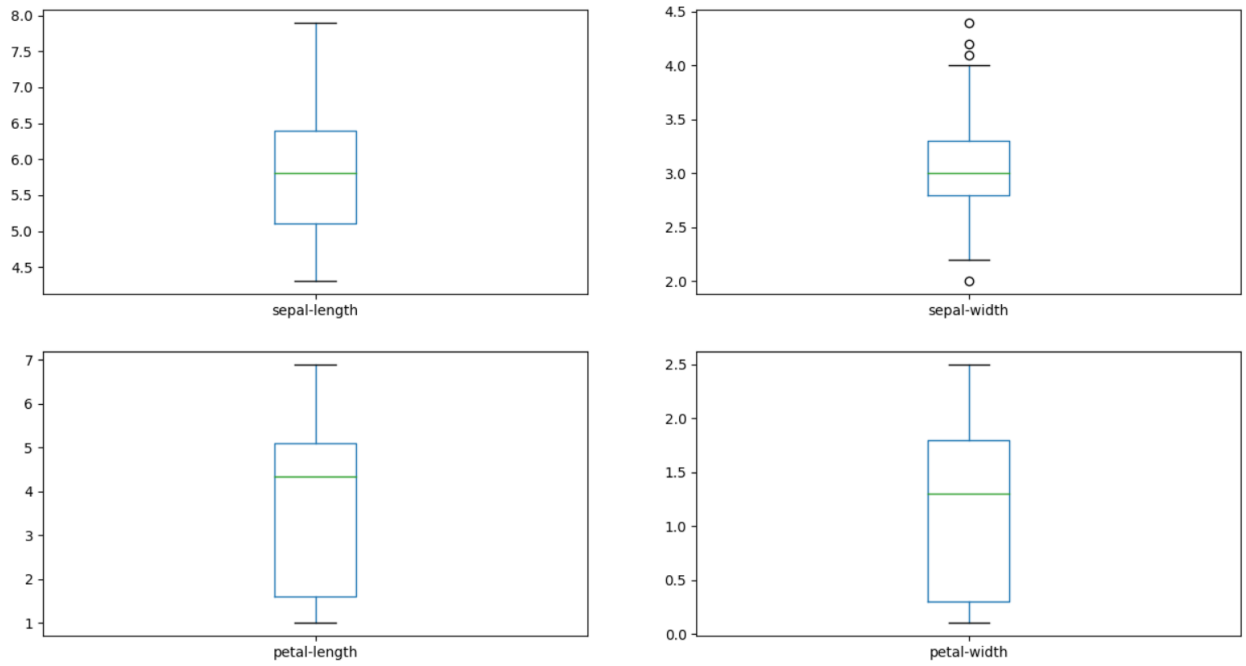


Figure 4.1 Iris Dataset Univariate plot: Box and whisker

Figure 4.1 shows the distribution of the iris data with respect to the attributes in the dataset. As shown in Figure 4.1, sepal length is skewed to the left, sepal length is almost skewed to the center, petal length is skewed to the left and petal length is skewed to the left as well.

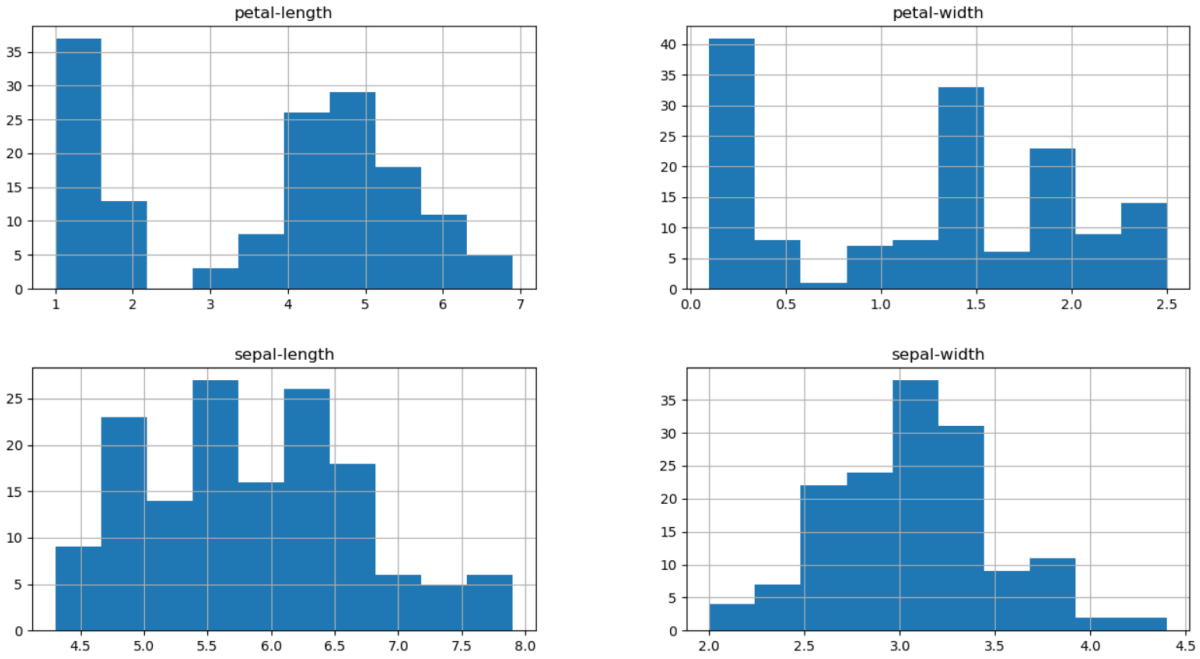


Figure 4.2 Iris Dataset Univariate Plot: Histogram

Figure 4.2 shows histograms representing the distribution of the iris dataset attributes for the three classes that is setosa, versicolor and virginica.

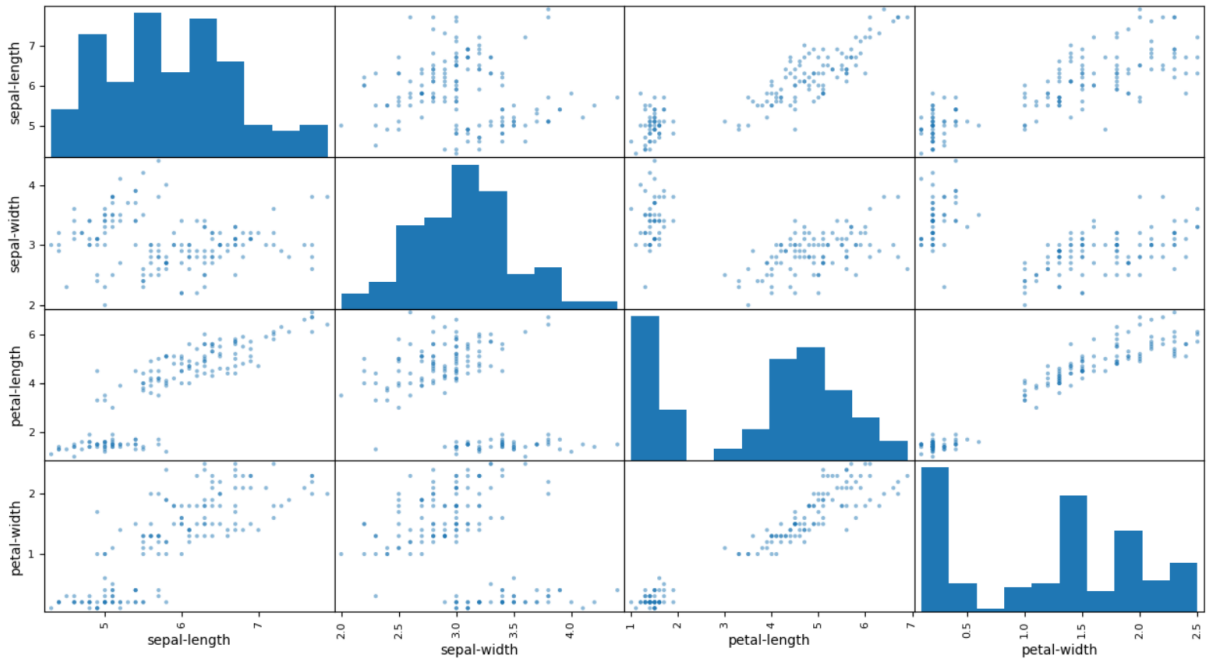


Figure 4.3: Iris Dataset Multivariate Plot

4.2.2 ALGORITHM EVALUATION

The results in this section show the performance of the algorithm that was developed in chapter 3. It evaluates the algorithm by making changes to gradient descent stepping as well as modification of the sample size. Results after the mentioned operations are shown in this section.

Here the researcher focused on linear binary classification of the data. So the original dataset which has 3 classes was divided into 2 classes. Our dataset (iris dataset) had one class (setosa) which was linearly separable from the other two (verginica and versicolor) as described in chapter two of the study. So the duty of the algorithm here was to linearly separate the setosa class from the other two classes (verginica and versicolor). Performance, accuracy and overfitting results are as shown below (Figure 4.4).

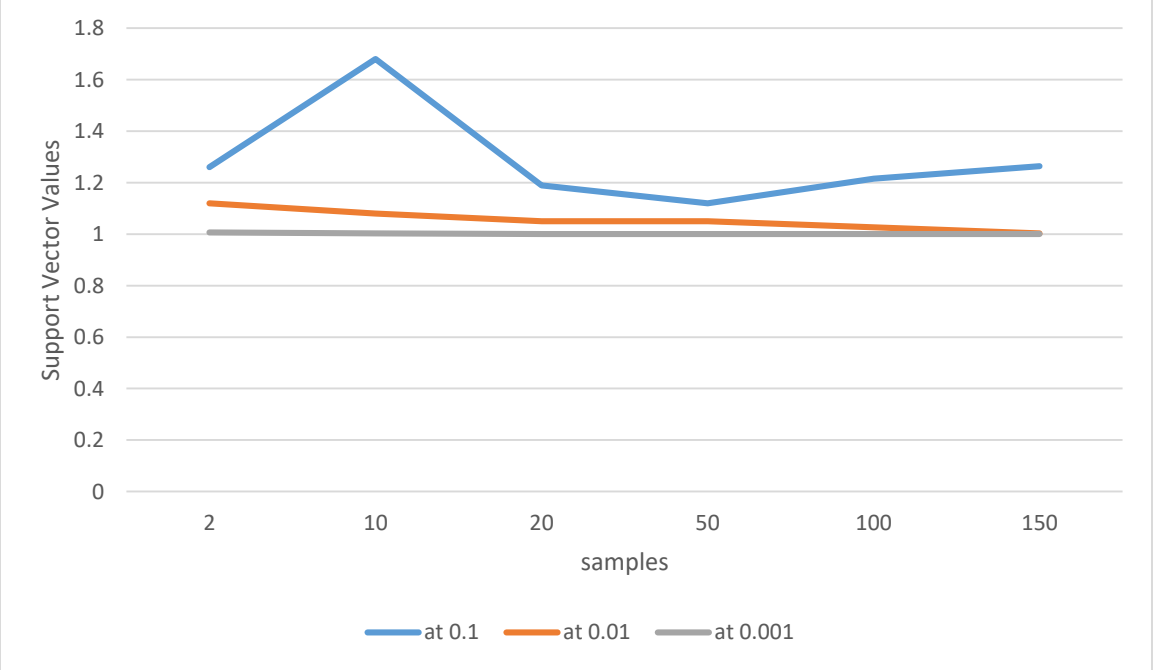


Figure 4.4 Algorithm Classification Accuracy using Iris Dataset

Figure 4.4 shows classification accuracy. Support vectors were given by $Y_i(\boldsymbol{x_i}.\boldsymbol{w} + b) = 1$ where Y_i represented our class. A feature (data point) in our feature space which gave us a value approximately 1 was our support vector. Not necessarily equal to 1since these values depended on values of **w** and **b** which were obtained through optimization. Figure 4.4 shows the accuracy of support vectors according to sample sizes as well as gradient descent steps. The x-axis represent sample sizes while the y-axis represent the values of support vector closest to one for each gradient stepping (see appendix 1 form more information). As shown in figure 4.3.1.1, the smallest gradient steps produced more accurate values of support vectors for all the sample sizes (that is 2, 10, 20, 50, 100 and 150). Gradient steps at 0.01 gets more accurate as the size of the training data increased while gradient steps at 0.1 were not consistent. This shows that if the algorithm is using small steps for example at 0.001, the accuracy of support vectors did not depend on the size of the training data though it did at larger steps.

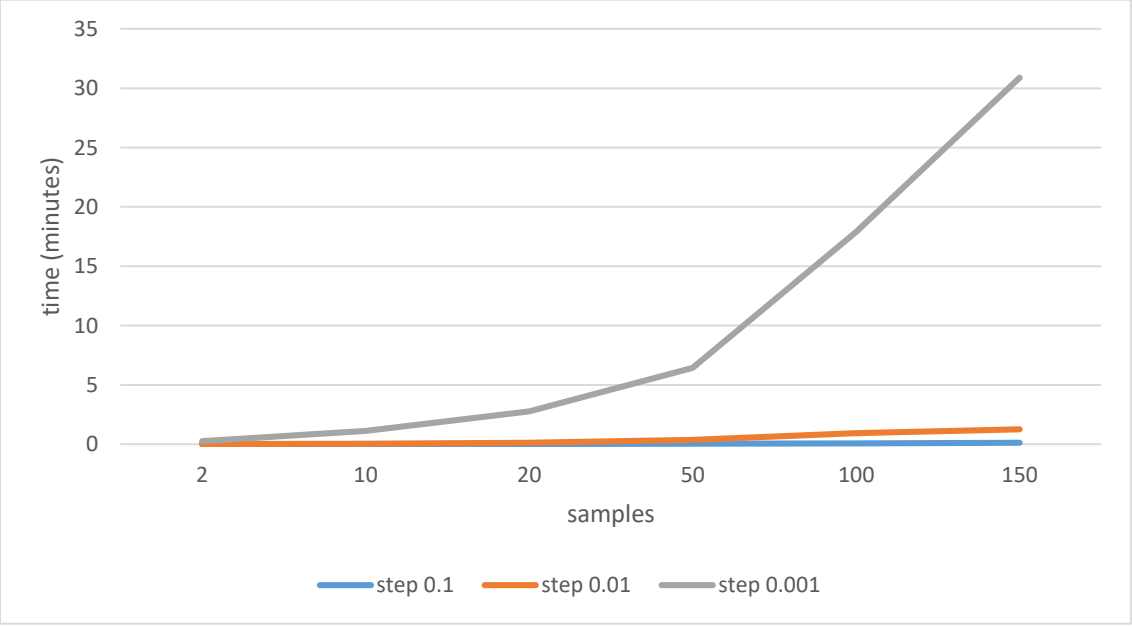


Figure 4.5 Algorithm Processing Time at Different Gradient Steps and Samples

The processing time of the computation or optimization was increasing as data was increasing. This is because the points in which the algorithm had to go through finding an optimum **w** were increasing. The algorithm hence had to go through many points in loops (recursion) to find the optimum **w**. However, the time was almost constant and far less for smaller steps i.e at 0.1 and 0.01 as shown in Figure 4.3.1.2. Big and powerful machines could lessen the processing time taken for optimizing our **w** on even smaller gradient descent steps ensuring that we get more accurate classification. However, larger steps for example 0.01 produced more accurate as the training data increased as shown in Figure 4.3.1.1 while at the same time taking reasonable amount of time to complete the optimization of **w** and *b* as shown in Figure 4.5.

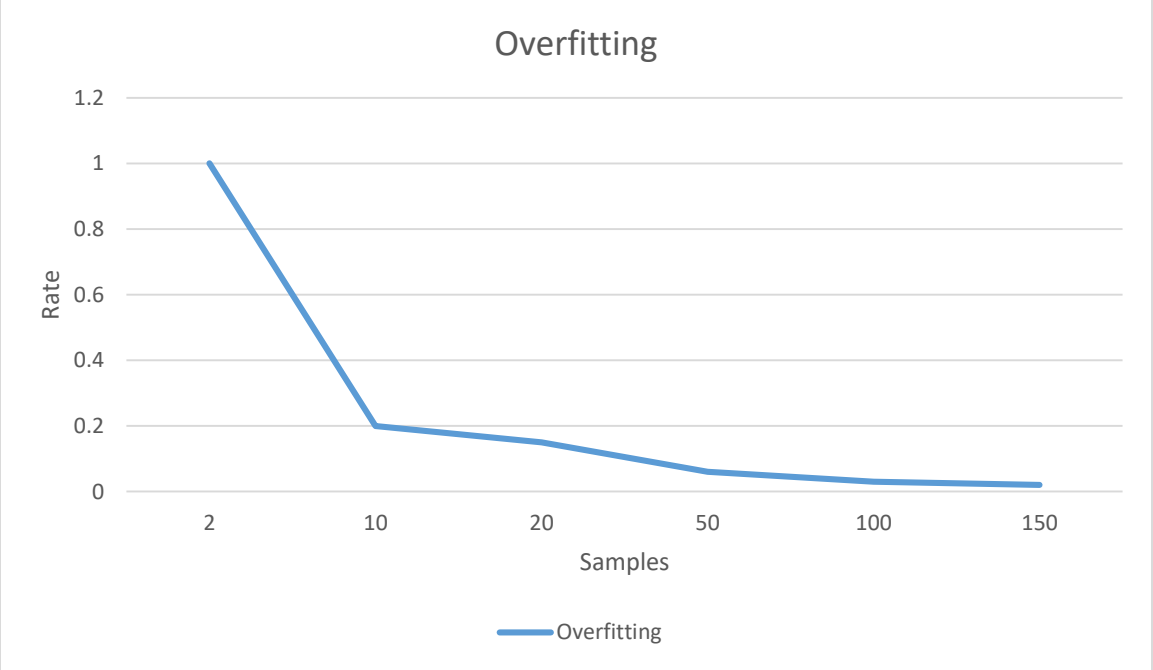


Figure 4.6 Test for Overfitting on iris Dataset

Overfitting rate = $\frac{Number\ of\ Support\ Vectors}{Size\ of\ training\ data}$

As training dataset was being increased, overfitting rate decreased as illustrated in Figure 4.6 above. Which implies an inverse relationship between the sample size and our overfitting rate.

4.2.3 CLASSIFICATION ON IRIS DATSET

Here the algorithm that was developed in this study was used for iris dataset classification. The dataset had some features removed (10 instances from each class) from it and then plugged back into the algorithm for data classification. Here (Figure 4.3.2.1) the model is presented with all features without any prediction just the classification of the training data and then presented after some data points have been removed and used as a prediction data in Figure 4.3.2.2.

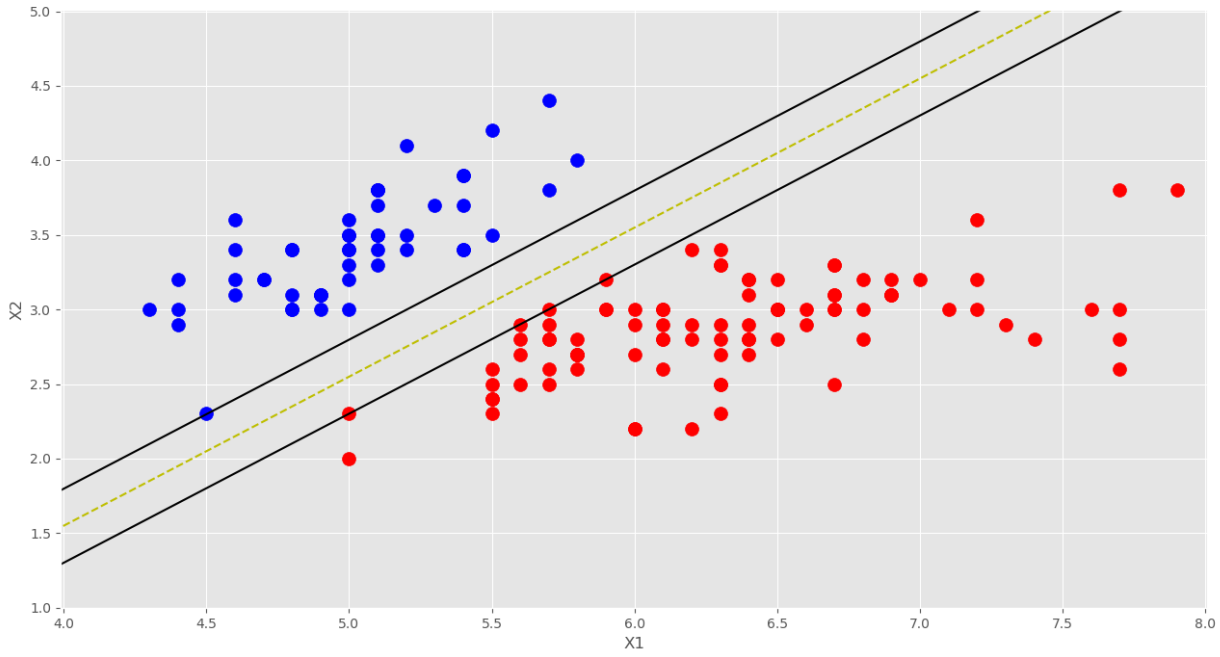


Figure 4.7 Classification of Iris Dataset Before Prediction

Figure 4.7 shows the basis of classification that has been created by the algorithm. The algorithm created the decision boundary (doted line), plotted all data points on, and colored with respective

colors. Blue representing the setosa class and red representing the non-setosa class which is the versicolor and virginica classes. The algorithm created as well the support vector planes which as shown by a continuous line for both the green (negative class) and the red (positive class).

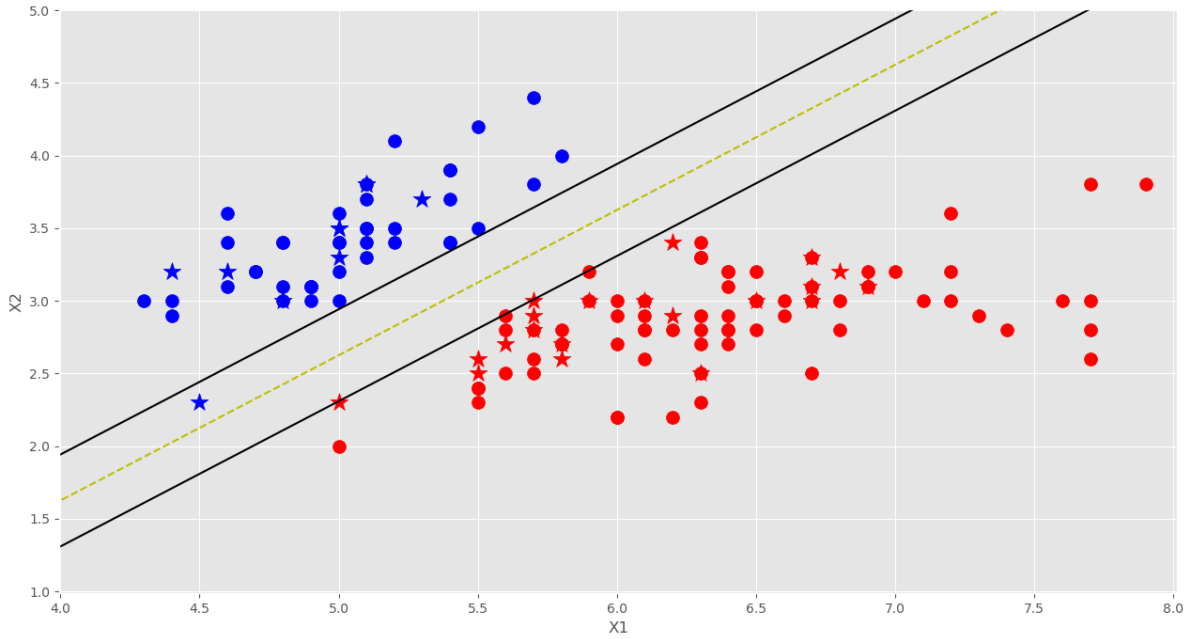


Figure 4.8 Classification of Iris Dataset after Sample Data was Removed and Prediction Done

The iris dataset was divided into two classes, training data and prediction data. 10 instances or samples from each of our three classes were set aside as prediction data and the remaining used as training data for the algorithm. This left us with 120 training instances with 40 from each class and 30 prediction instances with 10 from each class. The algorithm was trained with 120 instances and established decision boundaries as shown in Figure 4.3.2.1 but here with a smaller training data set. Later on, data was injected into the algorithm as prediction data and the results are shown in figure 4.8. The setosa class was plotted in blue dots (training data) and blue starts (prediction data) while the non-setosa (versicolor and virginica) classes were in red dots (training data) and red stars for prediction data.

As shown in Figure 4.8 the algorithm correctly predicted data that was set as prediction data. However, the data point that is on (4.5, 2.3) was used as prediction data which was once a training data point in figure 4.3.2.2. This point was a support vector. Since it was removed from the training data, the algorithm had to look for other support vector(s) for the red class and hence shifted up the hyperplane for the red class support vectors. Since the hyperplane is in the middle of the red class support vectors and the blue class support vectors it as well shifted upwards by half the distance the hyperplane for the red class shifted. Thus, we ended up having data point (4.5, 2.3) slightly below the negative class support vectors hyperplane but above the decision boundary and still belonging to the right class (that is the blue class or setosa class).

4.2.4 EVALUATING OTHER ALGORITHMS AND COMPARING THEM TO THE SVM USING IRIS DATASET

Here comparison of the SVM was done to many other algorithms such as Logistic regression(LR), Classification and Regression Trees(CART), K-Nearest Neigbhors (KNN), Linear Discriminant Analysis (LDA) and Gaussian Naïve Bayes (NB).

4.2.1 ACCURACY COMPARISON

Accuracy of the SVM algorithm was compared to other algorithms on the iris dataset and results are as shown below:

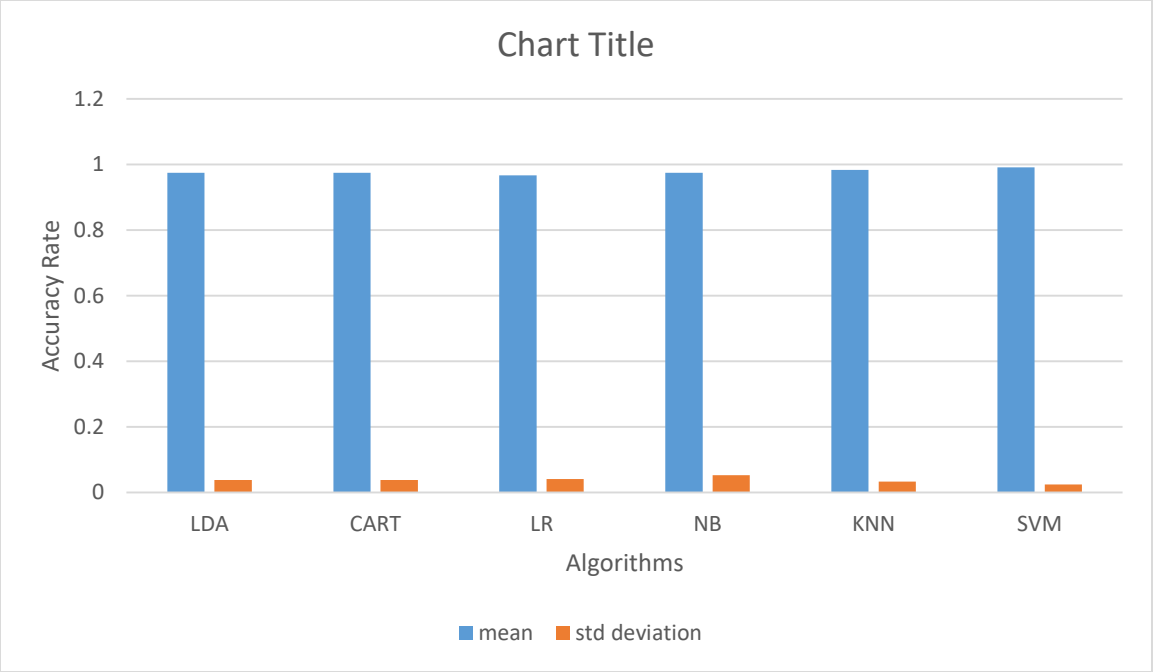


Figure 4.9 Algorithms Accuracy Comparison on Iris dataset

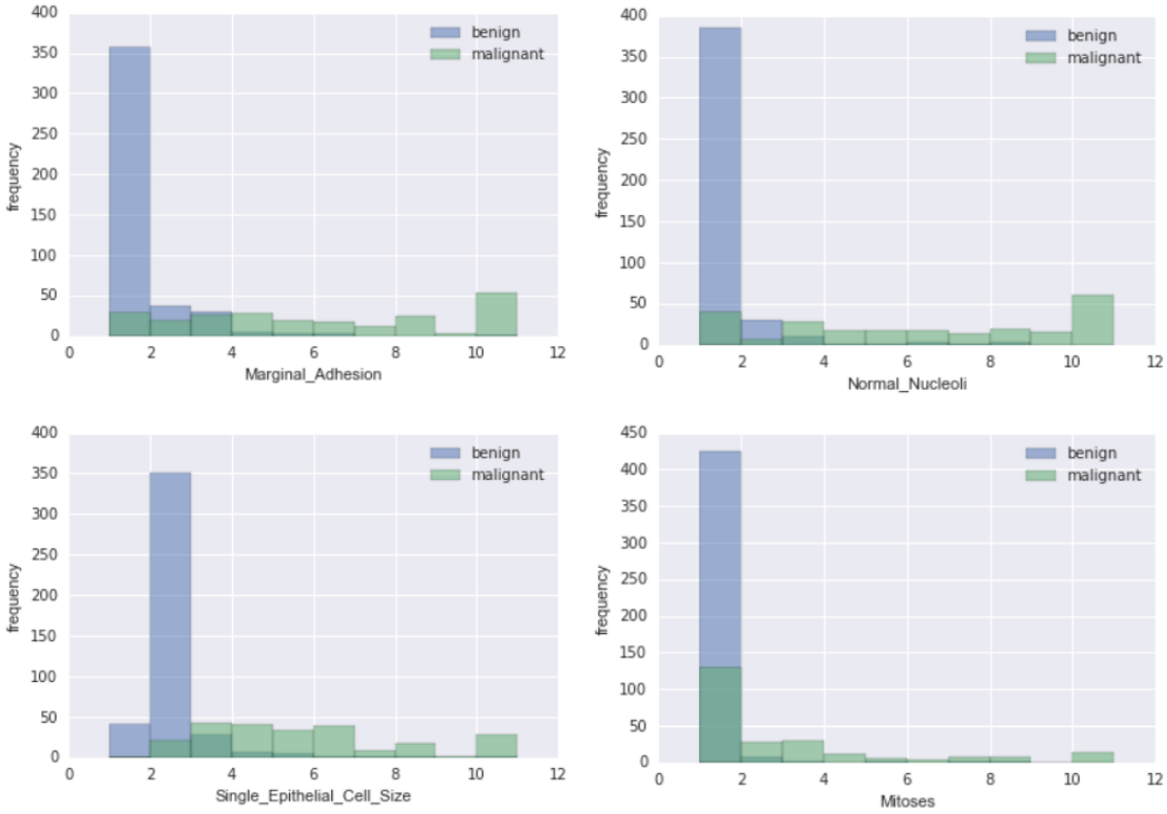
Here 6 models were compared estimating their accuracies. The 10 fold cross validation method was used where the total number of the instances in the dataset divided the ratio of the number of correctly predicted instances. The SVM proved to be more accurate with 99.1667% accuracy followed by KNN which has 98.3333% accuracy in classifying our iris dataset as shown in Figure 4.9.

4.3 BREAST CANCER DATASET

Breast cancer dataset was as well classified using the algorithm developed in chapter 3 together with the cxvopt module which handled the optimization problem using kernels. This section will start by showing a descriptive summary of the dataset and goes on to classify it into either benign or malignant classes.

4.3.1 EXPLORATORY DATA ANALYSIS ON BREAST CANCER DATASET

This section shows exploratory data analysis of the breast cancer dataset. It attempts to describe the breast cancer dataset.



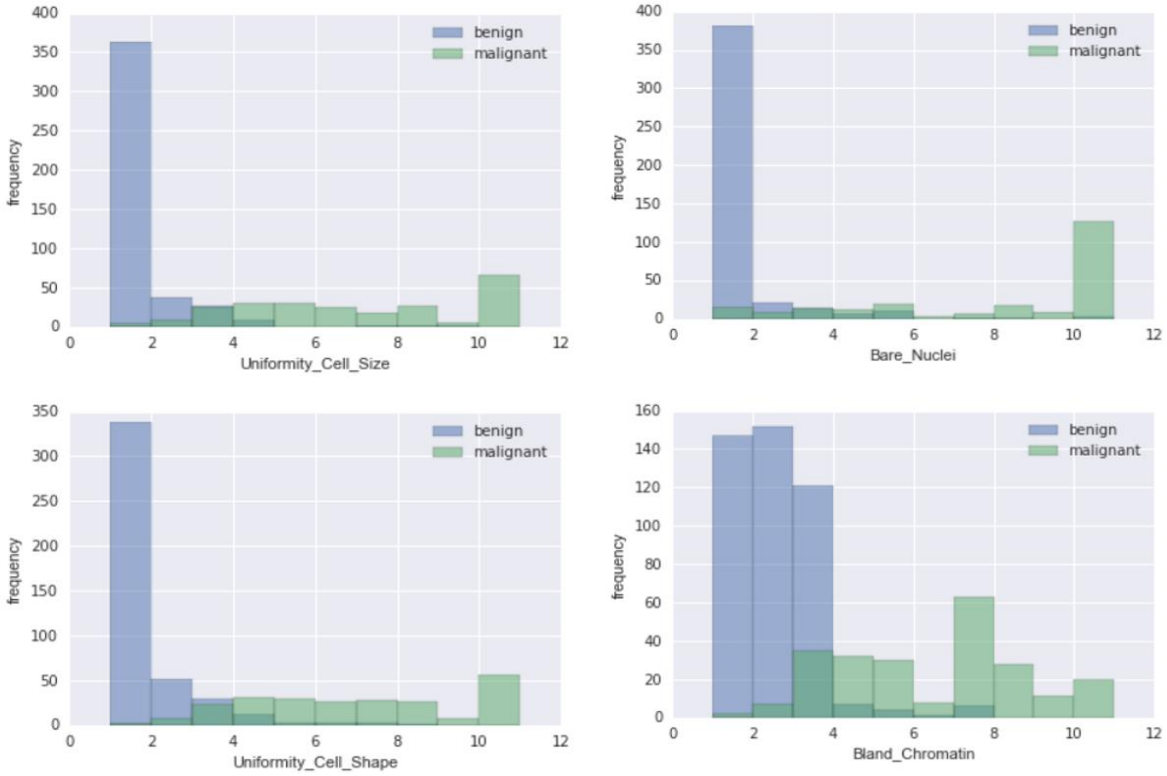


Figure 4.10: Breast Cancer Dataset Attribute Distribution

Figure 4.10 shows the distribution of attribute values for both our classes that is the benign and malignant classes. As illustrated in Figure 4.10, the benign class is skewed to the left (towards lower score that is from 1-3 regardless of type of features while malignant cells have a score ranging from 1-10 and slightly skewed to the right (that is higher scores).

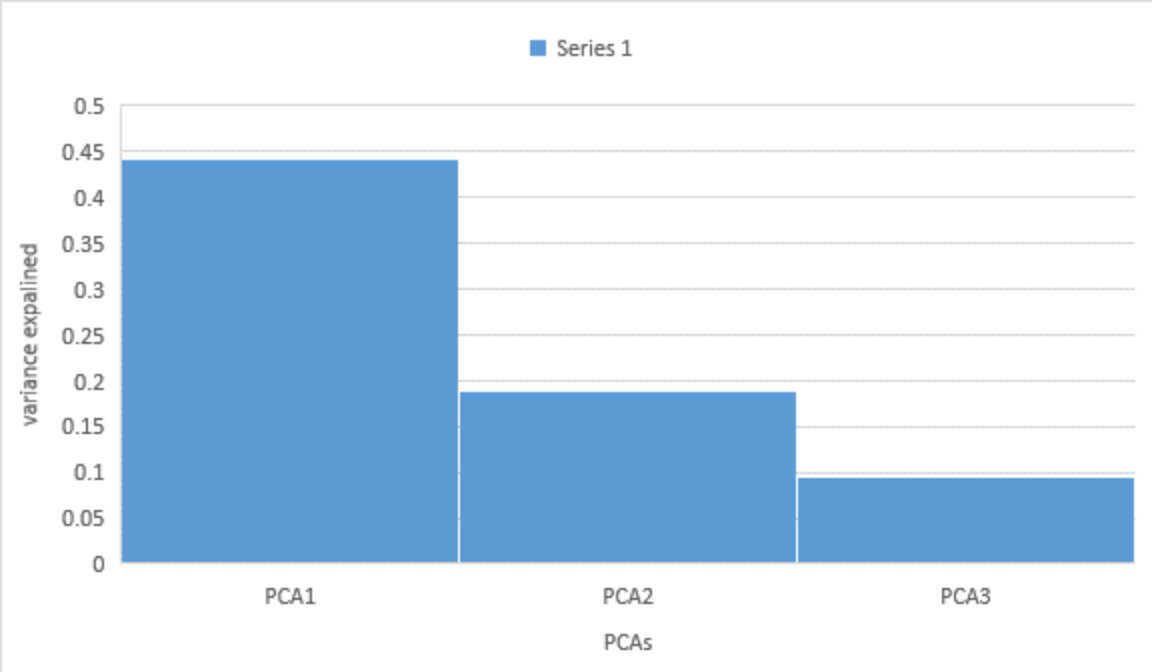


Figure 4.11: Breast Cancer Dataset Principal Component Analysis

Figure 4.11 shows results for Principal Component Analysis (PCA) for the breast cancer dataset. The data was decomposed and broken down from 30 dimensions into 3 dimensions. This was done so that it could be visualized. PCA was used to find a pattern making sure that the variation in the dataset is returned. Since some variance may be lost, PCA was used to try to explain as much of the variance as possible. A shown in figure 4.11, the last dimension or principal component (PCA3) retains 9.4 % of the variance. Adding more dimensions will not result in huge changes hence the data was reduced to 3-dimensions without affecting much of its structure. Adding up the three PCAs in figure 4.11 gave 0.7264 implying 72.64% of the variance was retained.

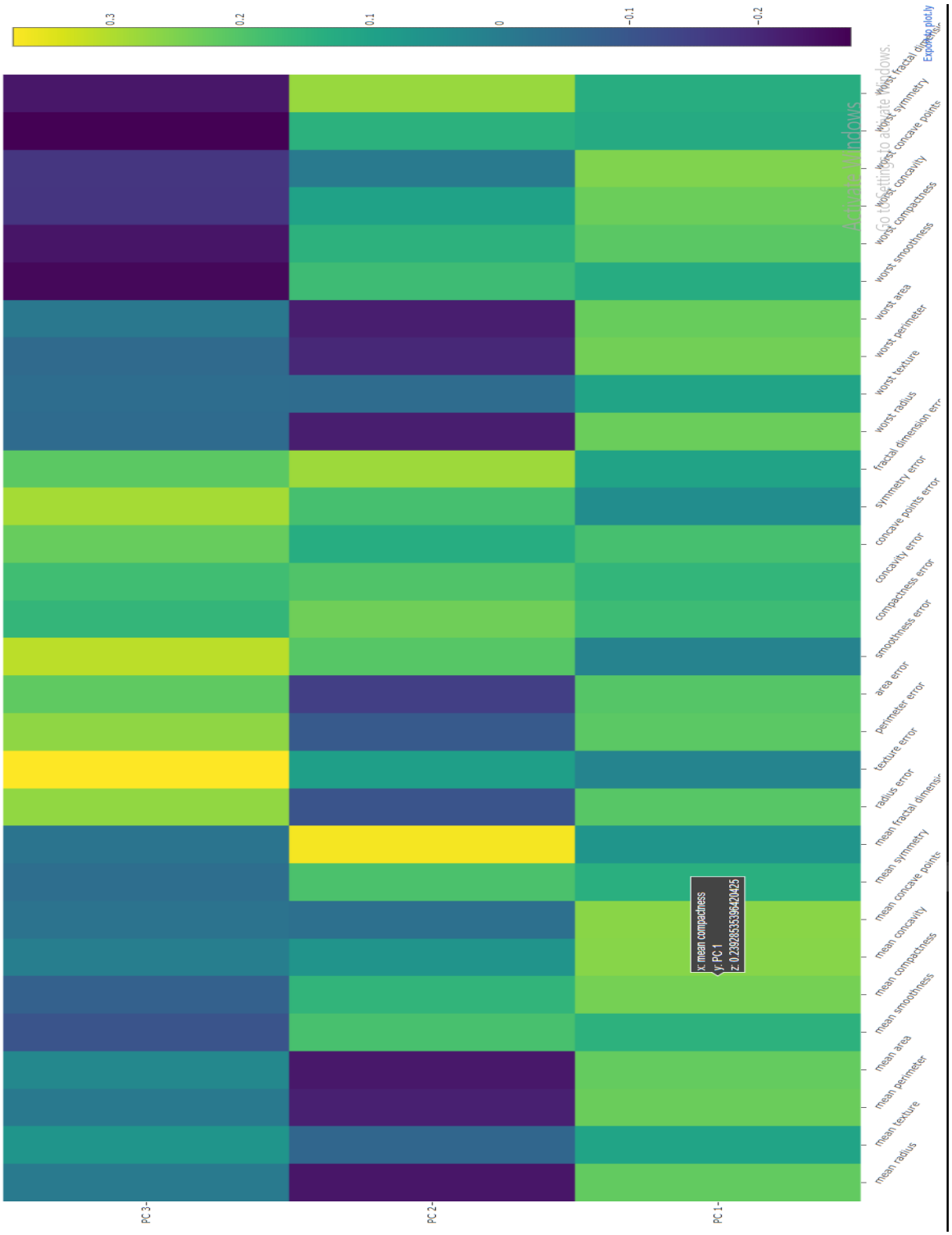


Figure 4.12: Heat Map on Breast Cancer Dataset

In figure 4.3.1.4, a heat map was created using the heat map function available in python. It was created by supplying the principal component as z-axis. The x-axis was the feature and the y-axis the principal component.

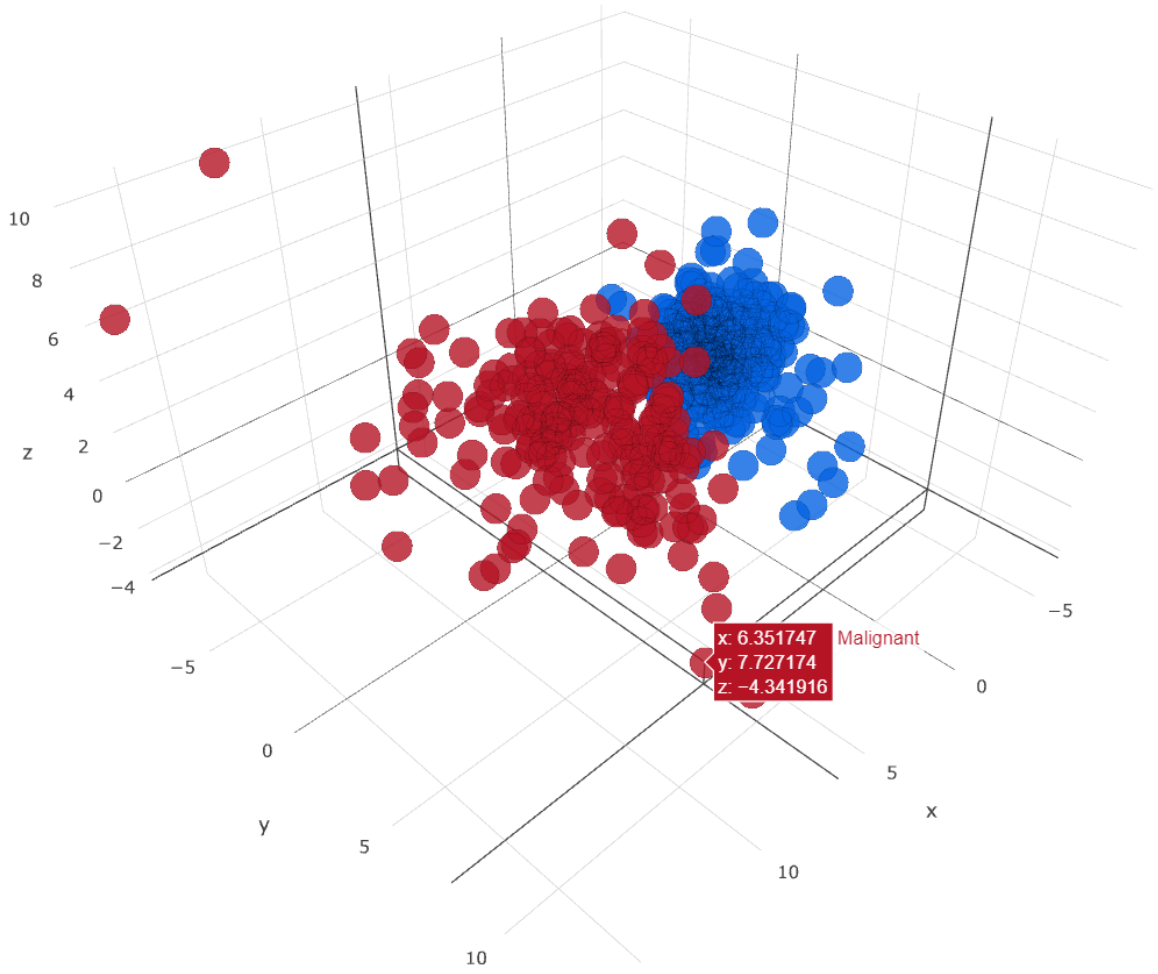


Figure 4.13: 3-D Scatter Plot for Breast Cancer Dataset

In figure 4.13, the dataset was divided into its respective classes. That is benign and malignant classes. This was done by adding the original dataset into the transformed dataset. Thus the labels for the classes was determines e.g. benign or malignant. Visualizations were created using

Scatter3d function. An additional styling was added so that the classes can be distinguished by colors on the 3-D vector space as illustrated in Figure 4.13. The iplot function was then used to visualize the dataset. It is clear that the iris dataset is not linearly separable as shown in Figure 4.13.

4.3.2 CLASSIFYING THE BREAST CANCER DATASET

This section shows results of breast cancer dataset classification. The dataset was further reduced to 2-D using PCA as shown in Figure 4.14

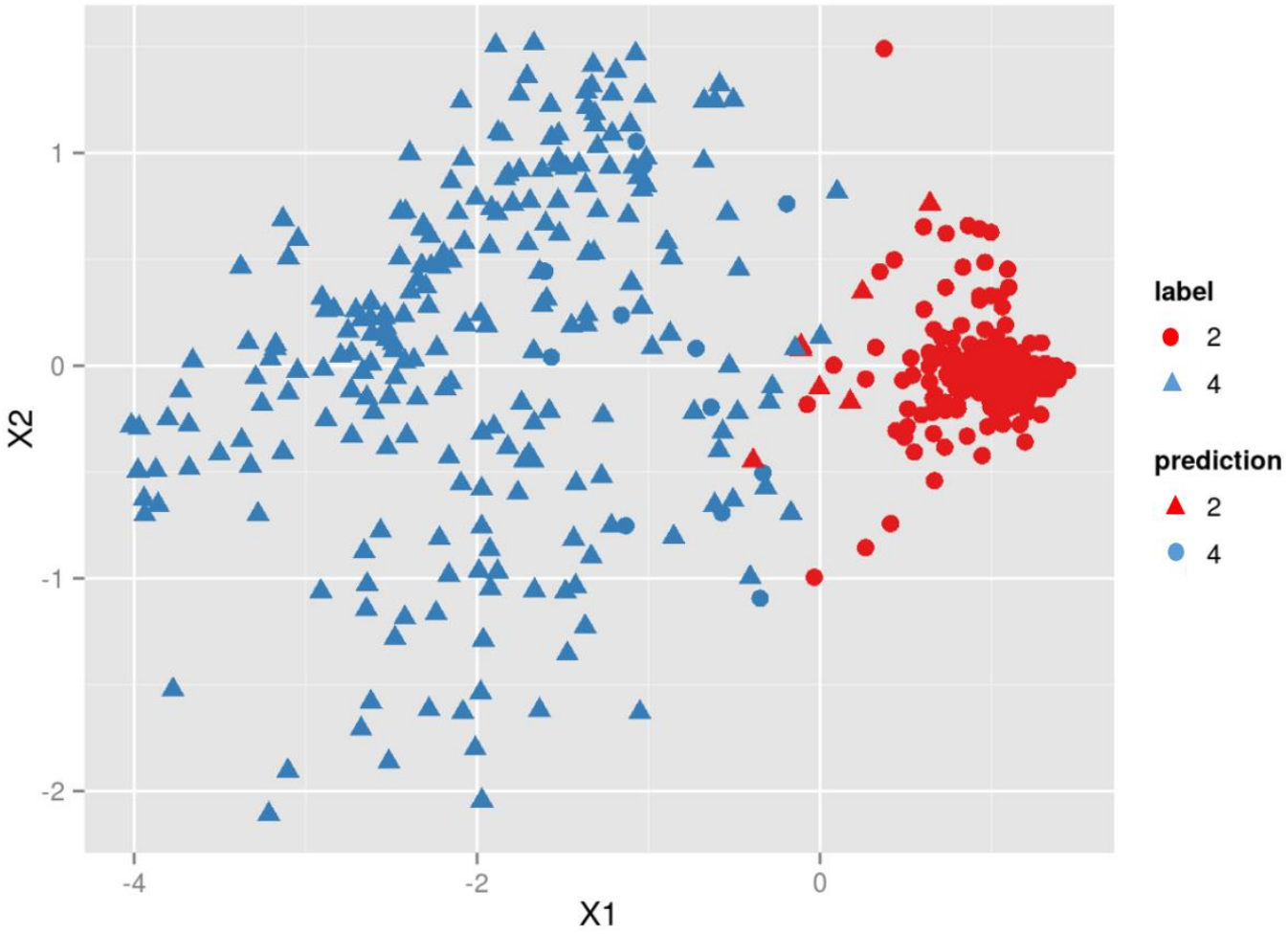


Figure 4.14: Breast Cancer Dataset Classification Results

The breast cancer dataset was divided into two classes. One class used as training data and the other as prediction data. Ten instances (sample size) were removed from each class and were used as prediction data while the remaining data was used to train the algorithm. Colors and shapes were used to visually display our data. Red color was used for the benign class while the blue class was used for the malignant class. For the benign class (in red) circles were used as training data labels while triangles were uses as prediction data labels. For the malignant class, triangles were used as training data labels while circles were used as prediction data labels as shown in figure 4.14.

2 represents benign and 4 represent malignant. The algorithm managed to correctly classify the prediction data for the breast cancer dataset using the cvxopt optimization method.

4.3.3 ALGORITHM EVALUATION

Three other algorithms were used and their accuracy compared to the accuracy provided by the SVM algorithm. These algorithms and or models are Linear Discriminant (LD), Neural Networks (NN) and K-Nearest Neighbors (KNN). This was done by separating the breast cancer dataset into two groups. One group was used as Training data while the other was used as prediction data. The dataset had 699 instances for both the classes of which 458 instances were for the benign class while the rest (241 instances) were for the malignant class.

The researcher wanted to use 20 instances as testing, classification or prediction size. So to create this testing class the following computations were considered by determining the ratio of the classes to the dataset as follows:

$$\frac{\text{class size}}{\text{dataset size}}$$

$\frac{458}{699} = 0.655221745$ (for the benign class)

$\frac{241}{699} = 0.344778254$ (for the malignant class)

Now to get number of instances that are supposed to be removed from each class to give us 20 instances following computations were taken:

$0.655221745 \text{ x } 20 = 13$ (for benign class)

And the rest (which is 7) for the malignant class.

This was done for maintaining the ratio of class sizes so not to cause bias during the raining of the algorithm especially by taking same number of testing data from classes of different sizes. The figure 4.16 below summarizes the classes we end up having for our cross validation:

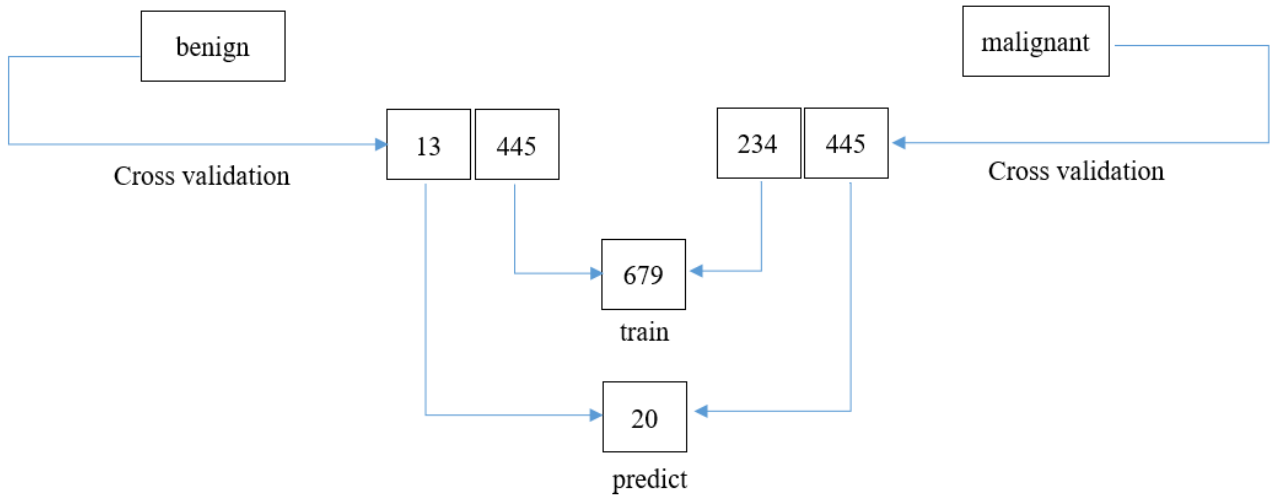


Figure 4.15: Training and Testing Data Allocation

Cross validation (Hart, Duda & Stork, 2001) was used for other classification algorithms (mentioned in this section) together with our SVM. Cross validation was used so that the researcher

would have an average result for all (10) the accuracy calculations. This meant that the researcher had a more realistic value thus avoiding peak results of the classifications. Cross validation was repeated for choosing test dataset before any classification procedure. Results of the accuracy findings are summarized in the Figure 4.17 below.

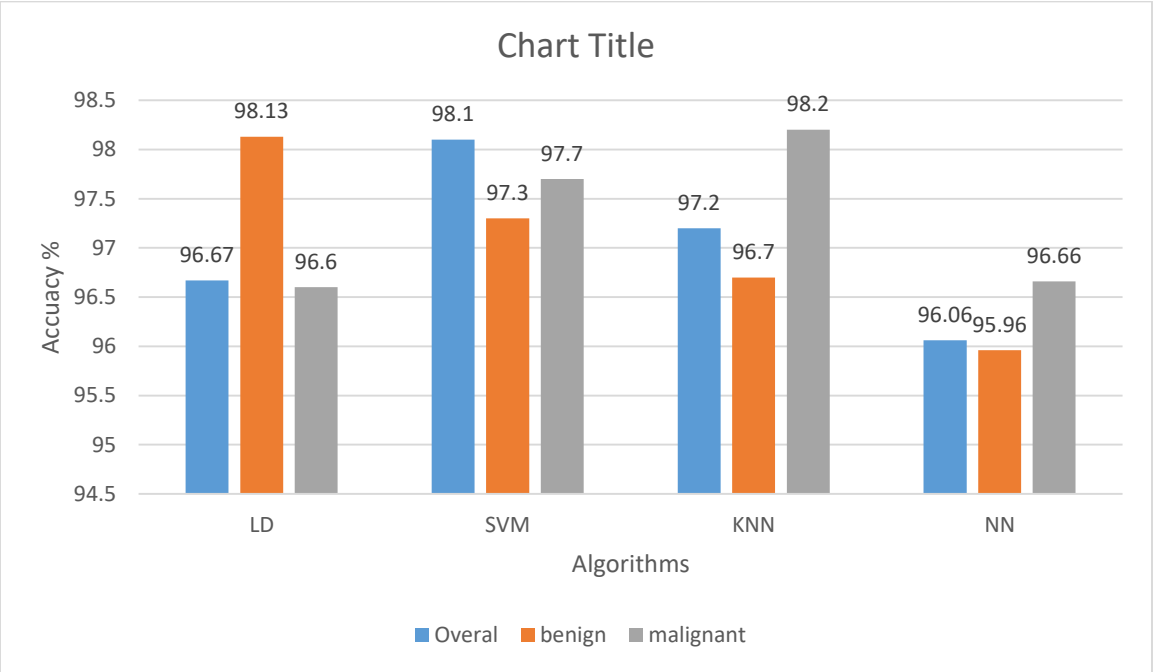


Figure 4.16: Breast Cancer Classification Accuracy Comparisons

Figure 4.3.2.3 shows that SVM algorithm has the highest overall accuracy 98.1%. This shows that it is the best model that suits classification for our breast cancer dataset. It however has lower percentages for the classification of respective classes as compared to other algorithms. However, it has constantly high accuracy power for both the benign and malignant classes. This is evident enough to show that SVM modelling is the most appropriate modelling for data classification.

4.4 CONCLUSION

The researcher presented, interpreted and analyzed the findings of the study in this chapter. A number of other algorithms were also looked at and determined how they best fit the iris data set as well as the breast cancer dataset. SVM algorithm proved to be more accurate in the classification of both the datasets. Of particular importance to the study was the evaluation of the algorithm developed in chapter 3. It was proven that the more the data we use the more the accuracy of our results or classification as in the case of our study. However, it was clearly shown that SVM optimization problem takes time to be solved as data increases. However, with bigger and more powerful machines, the time is not as high as compared to smaller computers such as the one that was used in this research. The good thing however is that the algorithm only consumes a bit of time during training. Once training has been done, predictions will then happen instantaneously since all decision boundary rules would have been established. The following chapter will present conclusions to the study as well as recommendations.

CHAPTER 5

CONCLUSIONS AND RECOMMENDATIONS

5.1 INTRODUCTION

This chapter concludes the research. Here, a summary of the research is presented and findings interpreted and discussed. Of particular importance to the study was the exploration of machine learning in an attempt to discover if existing mathematical formulations and or statistical modelling can be used by mathematics and statistics majors as well as organisations to create algorithms for big data analysis.

Literature review indicated that Machine learning has its deep roots in mathematics, mathematical optimization to be specific. Thus, the exclusion of mathematics and or statistics majors in big data analysis more specifically using ML could really be the reason why the growth of data is outpacing our capabilities to analyze it (Tech Insider, 2009).

The study could not just end in showing literature that relates Computer science which itself is a field that introduced Machine Learning since it is a scientific study. The researcher went on to develop an algorithm for data classification. This was as well zeroing in on exploring machine learning in light of both mathematics and statistics since the researcher focused more on translating mathematical formulations into a machine-learning algorithm through. The developed algorithm was then used for data classification.

Here is a quick recap of the objectives of the study:

- To demystify machine learning by relating it to mathematics and or statistics
- To develop a Support Vector Machine algorithm from mathematical formulations

- To classify data using the algorithm

5.2 CONCLUSIONS

The algorithm developed in this study will be able to classify breast tumors as either benign or malignant given digitalized image of a needle aspirate of a (FNA) of a breast mass as shown in the study. Moreover it will be able to classify iris flowers as either setosa or non-setosa (virginica or versicolor).

In addition, it is evident that machine-learning algorithms are built by combining mathematical formulations especially those that are related to optimization. The study also shows that SVM are very accurate at data classification than other algorithms such as KNN, NB, CART, LDA, NN and LR. The more the data, the higher the accuracy of SVMs. However, SVMs tend to take more time during algorithm training especially at small gradient steps. In addition, SVMs do not only classify data into two groups only, they classify data by separating one class data from the other(s) one at a time.

5.3 RECOMMENDATIONS

Based on the findings of the study, here is a list of recommendations made by the researcher:

- The study recommends engagement of Machine Learning algorithm development as a way of increasing participation of big data analysis thus help close the gap between the growth of data and our capacity to analyze it.
- The study recommends use of high performance computing for big data analysis
- The study recommends usage of open data datasets for the purposes of learning big data analysis as well as testing Machine Learning algorithms
- The study recommends the use of the algorithm development approach in businesses problems such as determining people who are most likely to repay loans thus avoiding lending money to bad debtors.

5.4 AREAS FOR FURTHER STUDY

The researcher proposes that a further study to be conducted to evaluate big data technologies such as Machine Learning and see how we can use our mathematical and or statistical knowledge to develop and or use machine learning algorithms.

5.5 REFERENCES

Auld, A., 2017, 'The Problem in Big Data: Lack of Skills', viewed 14 August 2017, from https://www.computing.co.uk/ctg/opinion/3013853/the-big-problem-in-big-data-a-lack-of-skills-skills

Bennet, K.P. & Parrado-Hernandez, E. 2007, 'The Interplay of Optimization and Machine Learning Research' Review, vol. 1, no. 1, pp. 1.

Berwick, R., (nd.)'An Idiot's guide to Support vector machines (SVMs)', viewed on 20 August 2017, from http://web.mit.edu/6.034/wwwbob/svm-notes-long-08.pdf

Breiman, L., 2001, Statistical Modeling: The Two Cultures, *Statistical Science* 16: 3, 199–231.

Cesarani, K., 2016, 'Why Learn Python', viewed 14 December 2016, from http://www.bestprogramminglanguagefor.me/why-learn-python

CVXOPT, nd., 'Convex Optimisation', n.d., viwed on 11 August 2017, from http://cvxopt.org/#

Datascienceassn.org, nd., 'Intoduction to Machine Learning', viewed on 4 February 2017', from http://www.datascienceassn.org/sites/default/files/Introduction%20to%20Machine%20Learning.pdf

Efron, B. and Tibshirani, R. (1991). Statistical analysis in the computer age, Science 253: 390–395.

Evgeniou, T., Pontil, M. & Poggio, T., (2000), Regularization networks and support vector machines, Advances in Computational Mathematics 13(1): 1–50.

EY, 2014, 'Big Data: Changing the way we compete and operate', viewed 3 January 2017, from http://www.ey.com/Publication/vwLUAssets/EY_-

_Big_data:_changing_the_way_businesses_operate/%24FILE/EY-Insights-on-GRC-Big-data.pdf

Gates, G.W. (1972) "The Reduced Nearest Neighbor Rule". IEEE Transactions on Information Theory, May 1972, 431-433.

Gordon, A. (1999). Classification (2nd edition), Chapman and Hall/CRC Press, London.

Guess, A., 2015, 'Only 5% of all data is currently being analyzed', viewed 9 July 2017, from http://www.dataversity.net/only-0-5-of-all-data-is-currently-analyzed

Guo, P., 2014, 'Python is Now the Most Pupolar Introductory Teaching Language at Top U.S. Universities', viewed 18 January 2017, from https://m.cacm.acm.org/blogs/blog-cacm/176450-python-is-now-the-most-popular-introductory-teaching-language-at-top-u-s-universities/fulltext

Hastie, T. (ed.), Tibshirani, R. (ed.) & Fiedman, J. (ed.), 2008, The Elements of Statistical Learning, Stanford, California

Hastie, T. and Zhu, J. (2006). Discussion of "Support vector machines with applications" by Javier Moguerza and Alberto Munoz, Statistical Science 21(3): 352–357.

Jose, U., 2016, Python for Probability, Statistics, and Machine Learning, Springer, Switzerland

Hastie, T. Tibshirani, R. and Friedman, J. (1991). The Elements of Statistical Learning, 2nd edition.

Hastie, T., Rosset, S., Tibshirani, R. and Zhu, J. (2004). The entire regularization path for the support vector machine, Journal of Machine Learning Research 5: 1391–1415. Hospitals, Madison

IBM, 2012, 'Fueling Change with Big Data Growth and Analytics', viewed 3 March 2017, from

IDC, 2012, 'The Digital universe in 2020', viewed on 10 September 2017, from https://www.emc.com/collateral/analyst-reports/idc-the-digital-universe-in-2020.pdf

IDC, 2017, 'The state of Big Data Analytics in South Africa', viewed 20 August 2017, from https://www.idc.com/getdoc.jsp?containerId=CEMA42083917

Krebs, D., 2007, 'Introduction to Kernel Methods', viewed on 21 March 2017, from https://people.cs.pitt.edu/~milos/courses/cs3750-Fall2007/lectures/class-kernels.pdf

Lin, X. (ed.), Genest, C. (ed.), Banks, D.L., (ed.), Molenberghs, D.W.S., (ed.) & Wang, J., 2013, The past, Present and Future of Statistical Science, CRC Press, NY

Marr, B. 2016, 'A short History of Machine Learning Every Manager Should Know', viewed 16 April 2017, from https://www.forbes.com/sites/bernadmarr/2016/02/19/a-short-history-of-machine-learning-every-manager-should-read/amp/

Marr, B, nd.,'What is Big Data: A Simple Explanation for Everyone', viewed, 13 May 2017, from https://www.bernardmarr.com/default.asp?ContentID=766

MGI, 2011, 'Big data: The next frontier for innovation, competition, and productivity', viewed on 18 August 2017, from https://bigdatawg.nist.gov/pdf/MGI_big_data_full_report.pdf

Michie, D., Spiegelhalter, D. and Taylor, C. (eds) (1994). Machine Learning, Neural and Statistical Classification, Ellis Horwood Series in Artificial Intelligence, Ellis Horwood

Mitchel, T. (1997), Machine Learning, McGraw Hill, NY

Norvig, P. & Russel, S.J., 1994, Artificial Intelligence: A Modern Approach Artificial Intelligence, Prentice Hall, N.J

Paulson, H.G., 2010, Computer Science students Need Adequate Mathematical Background, viewed on 10 February 2017 from http://users.math.uoc.gr/~ictm2/Proceedings/pap398.pdf)

Richert, W., & Coelho, L.P., 2013, Building Machine Learning Systems in Python, Packt Publishing Ltd, Birningham, UK

Rossi, B., 2015, 'How to address the talent shortage in big data', viewed 9 April 2017, from http://www.information-age.com/how-to-adress-talent-shortage-big-data-123459664/

Shah, A., 2016, 'Machine Learning vs Statistics', viewed on 1 April 2017, from https://www.kdnuggets.com/2016/11/machine-learning-vs-statistics.html

Shmueli, G., 2010, To Explain or to Predict? *Statistical Science*. pp. 289–310. DOI: 10.1214/10-STS330

Smith, C., 2006, 'The History of Artificial Intelligence', viewed on 20 August 2017, from https://courses.cs.washington.edu/courses/csep590/06au/projects/history-ai.pdf

Smola, A. & Vishwanathan, S.V.N., (2008), Intoduction To Machine Learning, Cambridge University Press , Cambridge

Streitz, N. (ed.) & Stephanidis, C. (ed.), 2013, Distributed, Ambient, and Pervasive Interactions, Springer, NY

Srivastava, T., 2015, 'What is the Difference Between Machine Learning and Statistical Modelling', viewed 23 January 2017, from https://www.analyticsvidhya.com/blog/2015/07/difference-machine-learning-statistical-modeling/

UCI, n.d.a, 'Iris Data Set', viewed 3 July 2017, from https://archive.ics.uci.edu/ml/datasets/iris

UCI, n.d.b, 'Breast Cancer Wisconsin (Diagnostic) Data Set', viewed 10 March 2017, from https://archive.ics.uci.edu/ml/datasets/breast+cancer+wisconsin+%diagnostic%29

Wesserman, L., (nd.) 'The Rise Of Machines', viewed 4 April 2017, from http://www.stat.cmu.edu/~larry/Wasserman.pdf

Wolberg, W. H., 1991. *Wisconsin breast cancer database,* University of Wisconsin

Zhong, P. & Fukushima, M., 2005, A Regularized Nonsmooth Newton Method for Multi-class Support Vector Machines, viewed 21 December 2016, from http://rexa.info/paper/81ad8e5b8306aee758f09fd5c1caa8a23c63c2d6

APPENDICES

APPENDIX 1: SVM TRAINING RESULTS ON IRIS DATASET

Training data sample	steps	Step values	Time Taken (minutes)	Lowest Support vector Values	Overfitting SVs/sample
2	1	0.1	0.0028770333333333334	1.82	1
				1.26	
	2	0.1	0.0029605333333333336	1.19	
		0.01	0.010793633333333334	1.12	
	3	0.1	0.0030025666666666667	1.0262	
		0.01	0.010810333333333333	1.0066	
		0.001	0.2501607166666667		
10	1	0.1	0.009710683333333333	1.96	0.2
				1.68	
	2	0.1	0.00799865	1.08	
		0.01	0.0434908	1.176	
	3	0.1	0.009853	1.0024	
		0.01	0.0466445	1.0178	
		0.001	1.1214577833333335		
20	1	0.1	0.020373116666666666	1.19	3/20 = 0.15
				1.68	
	2	0.1	0.01483575	1.05	
		0.01	0.11224455	1.4	
	3	0.1	0.016205716666666665	1.001	
		0.01	0.11090921666666667	1.015	
		0.001	2.766640333333333		
50	1	0.1	0.03445661666666667	1.12	3/50 = 0.06
				1.12	

	2	0.1	0.040087083333333336	1.05	
		0.01	0.3512348	1.05	
	3	0.1	0.03514883333333333	1.001	
		0.01	0.3088650333333333	1.001	
		0.001	6.418629933333334		
100	1	0.1	0.06814051666666666	1.216	3/100 = 0.03
				1.216	
	2	0.1	0.06612153333333333	1.026	
		0.01	0.9193962	1.026	
	3	0.1	0.10463033333333334	1.01612	
		0.01	0.7185570333333333	1.01612	
		0.001	17.90286943333333	1.001	
150	1	0.1		1.264	3/150 = 0.02
				1.264	
	2	0.1	0.1397314	1.0033	
		0.01	1.2433983666666668	1.0033	
	3	0.1	0.298786564	1.2043	
		0.01	2.788767576565	1.02378	
		0.001	30.89779854445554	1.001	

APPENDIX 2: BREAST CANCER DATASET SUMMARY

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	mean fractal dimension
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	0.2419	0.07871
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	0.1812	0.05667
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	0.2069	0.05999
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	0.2597	0.09744
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	0.1809	0.05883

worst radius	worst texture	worst perimeter	worst area	worst smoothness	worst compactness	worst concavity	worst concave points	worst symmetry	worst fractal dimension
25.38	17.33	184.60	2019.0	0.1622	0.6656	0.7119	0.2654	0.4601	0.11890
24.99	23.41	158.80	1956.0	0.1238	0.1866	0.2416	0.1860	0.2750	0.08902
23.57	25.53	152.50	1709.0	0.1444	0.4245	0.4504	0.2430	0.3613	0.08758
14.91	26.50	98.87	567.7	0.2098	0.8663	0.6869	0.2575	0.6638	0.17300
22.54	16.67	152.20	1575.0	0.1374	0.2050	0.4000	0.1625	0.2364	0.07678