KWAME NKRUMAH UNIVERSITY OF SCIENCE AND TECHNOLOGY

COLLEGE OF SCIENCE

THE EFFECT OF KERNEL FUNCTIONS ON THE ACCURACY OF A LINEAR-BASED MACHINE LEARNING MODEL

(A Case Study of USD-GHS Exchange Rate Prediction)

A THESIS SUBMITTED TO THE DEPARTMENT OF STATISTICS AND ACTUARIAL SCIENCE, KWAME NKRUMAH UNIVERSITY OF SCIENCE AND TECHNOLOGY IN PARTIAL FULFILMENT OF THE REQUIREMENT FOR THE DEGREE OF BSc. STATISTICS

November 14, 2020

Declaration

We hereby declare that this submission is our own work towards the award of the BSc. degree and that, to the best of our knowledge, it contains no material previously published by another person nor material which has been accepted for the award of any other degree of the University, except where due acknowledgment has been made in the text.

Berko Rosina	.....	.....
Student (6797816)	Signature	Date
Boateng Sheila Kwartemaa	.....	.....
Student (6799016)	Signature	Date
Donkor Frank	.....	.....
Student (6801216)	Signature	Date
Opoku Kennedy	.....	.....
Student (6810716)	Signature	Date

Certified by:

Dr. Gabriel Asare Okyere	.....	.....
Supervisor	Signature	Date

Abstract

According to literature, the performance of a machine learning model depends mostly on parameter tuning. In a linear model such as support vector regression, a good choice of kernel function is appropriate for the accuracy of the model. In this project we investigate the effect of three kernel functions of SVR i.e. **"Linear kernel", "Polynomial kernel"** and the **"Radial basis function kernel"** on the accuracy of exchange rate prices of the US dollar against the Ghana cedis. The results show a close Mean Square Error(MSE) in all the three kernels. Based on the results, the accuracy of the model improves when the choice of kernel is radial basis function. The effect of regularization parameter C and epsilon is also investigated. The conclusion made was that, although kernel functions has an effect on the accuracy of the model, the choice of kernel should be chosen based on the behaviour of the dataset.

Acknowledgement

First and foremost acknowledgement goes to the Almighty God for making it possible for us to live to undertake this study.

We also want to thank our supervisor Dr Gabriel Asare Okyere for his immense contribution and support for this project.

We want to say a big thank you to our parents for granting us the opportunity to pursue our tertiary education, and sponsoring it too.

For the lovely friends who expressed concern and offered help, we say God richly bless you.

Our final thank you goes to Mr.Justice Owusu Agyemang(MPhil), for guiding us throughout the length of this project.

Contents

Declaration . . . . . **iv**

Acknowledgement . . . . . **iv**

List of Tables . . . . . **v**

List of Figures . . . . . **vi**

1 INTRODUCTION **1**

1.1 Background 1

1.2 Problem Statement 2

1.3 Justification 2

1.4 Objective 3

1.5 Methodology 3

1.6 Limitation of the Study 3

1.7 Thesis Organization 4

2 LITERATURE REVIEW **5**

2.1 Introduction 5

2.2 Financial Forecasting Approaches 5

2.2.1 Fundamental Approach 5

2.2.2 Technical Approach 5

2.3 Related Works 6

2.3.1 Kernel Functions 6

2.3.2 Exchange Rate Prediction 7

3 METHODOLOGY **12**
3.1 Machine Learning 12
3.1.1 Supervised Learning 13
3.1.2 Support Vector Machine 14
3.1.3 Support Vector Regression SVR 21
3.2 Data 25
3.3 Data Preprocessing 26
3.4 Parameter Tuning 27
3.5 Evaluation Metrics 27
3.6 Tool 27
3.6.1 Python 28
3.6.2 Numpy 28
3.6.3 Pandas 28
3.6.4 Matplotlib 28
3.6.5 Seaborn 28
3.6.6 Jupyter Notebook 29

4 RESULTS **30**
4.1 Analysis of Kernel functions 31

5 CONCLUSION AND RECOMMENDATION **37**
5.1 Conclusion 37
5.2 Recommendation 37

References **42**

List of Tables

4.1 SVR Error and Confidence 32
4.2 Linear Kernel 35
4.3 Plynomial Kernel 35
4.4 Radial Basis Kernel 36

List of Figures

1.1 Thesis Organization 4

3.1 Supervised Learning Model 13
3.2 Linear Separable 14
3.3 Infinite hyperplanes 15
3.4 Geometric Margin: Distance between a point and a hyperplane . . 16
3.5 Maximum Margin Separating Hyperplane 18
3.6 Non-linearly separable data 19
3.7 Soft Margin 20

4.1 Time series Plot of the data 30
4.2 Difference Plot of the time series data 31
4.3 Metrics performance 32
4.4 Linear Kernel 32
4.5 Polynomial Kernel 33
4.6 Radial Basis Kernel 33
4.7 All Kernel Function 34

Chapter 1

INTRODUCTION

1.1 Background

In the world today, currency exchange rate plays a significant role in controlling the dynamics of the exchange market. And as such an appropriate and accurate prediction of exchange rate is a crucial factor for the success of many business, investors and even a country at large.

Currency Exchange Rate, also known as forex rate, specifies the value of one country's currency in relation to another. The forex market since it's establishment in the 1970's has experienced a vast growth over the years (Kamruzzaman and Sarker, 2004). It has been one of the leading financial market in the world with an estimate of $6.6 trillion per day as of April 2019.

FOREX rates are influence by a variety of factors including political and economic events, and even the psychological state of individual traders and investors. These factors are highly connected and interact with one another in a very complex manner. Those interactions are very dynamic, erratic and unstable. This complexity makes forex rates prediction difficult (Adetunji et al, 2011).

Trading at the right time with the relatively correct approach can result in large profit however trade based on an incorrect movement can risk big losses (Chander et al, 2015). Using suitable forecasting model and best methods can reduce the effect of the losses and also increase profitability.

Machine learning model in recent years have emerged as a powerful tool in

exchange rate predictions as compared to the traditional time series models. Many past studies (Nanayakkara et al, 2014; Alamili, 2011; Kamruzzaman and Sarker, 2004; Nagpure, 2019; Rojas and Herman, n.d.) suggest that machine learning model such as Random forest, Artificial Neural Network(ANN), and Support Vector Machine(SVM) perform better than traditional time series linear model. The success of these models are largely due to the ability to capture nonlinear features.

In machine learning, and in support vector machine in general, the use of kernels has attracted substantial attention of the research community. Such attention is attributed to the ability of the kernels in mapping data into a high dimensional feature space so that linear machine computational power can be increased (Chiroma et al, 2014). In this project we investigate the effect of kernel functions on the accuracy of support vector regression using the USD/GHS exchange rate.

1.2 Problem Statement

The search for the most approriate machine learning model for a specific application is a challenging task. This is because, to obtain an accurate result depends mostly on parameter tuning. The choice of kernel function affects the performance of SVR hence the need to determine which kernel is significant in achieving a good model accuracy.

1.3 Justification

The world is growing at a fast pace such that decisions and techniques used in the past are rarely applicable to current situations. Technology keeps increasing at an exponential rate and therefore there is the need to keep up with current trends. The finance sector is no exception to this change therefore smart de-

cisions need to be taken by individual and organizations to thrive in the field. Accurate prediction of exchange rate is of high importance in the global market. And as a result, the search for the appropriate parameter of the learning model is needed to ensure accurate results are obtained.

1.4 Objective

The main objective of this project is to investigate the effect of kernel functions of support vector machine on the USD/GHS exchange rate.

The specific objectives are as follows:

1. To investigate the variation of performance with respect to regularization parameter C.

2. To investigate the variation of performance with respect to the epsilon parameter.

1.5 Methodology

In this research we evaluated the effect of three kernel function namely linear, polynomial and radial basis using SVR on the USD/GHS exchange rate prices. Daily exchange rate data for about $5\frac{1}{2}$years was used in this study of which the first 80% was used to build the model and the remaining 20% for evaluating the model. The regularization parameter C and epsilon were varied as well. The performance is evaluated using Root Mean Square Error (RMSE), Mean Square Error(MSE), Mean Absolute Error (MAE) and R^2. The programming language used for the analysis is Python.

1.6 Limitation of the Study

Although factors like inflation and interest rate influence price fluctuations in the forex market, this research only focused on using historical data in building and testing the model.

1.7 Thesis Organization

The thesis report is organised into five main chapters. Chapter one presents the introductory part of the general overview of the study. Chapter two outlines a review on kernel functions and exchange rate prediction. Detailed explanation of the method is given in chapter three. The result and findings of the study is presented in chapter four. Chapter five presents the conclusion and recommendation of the study.

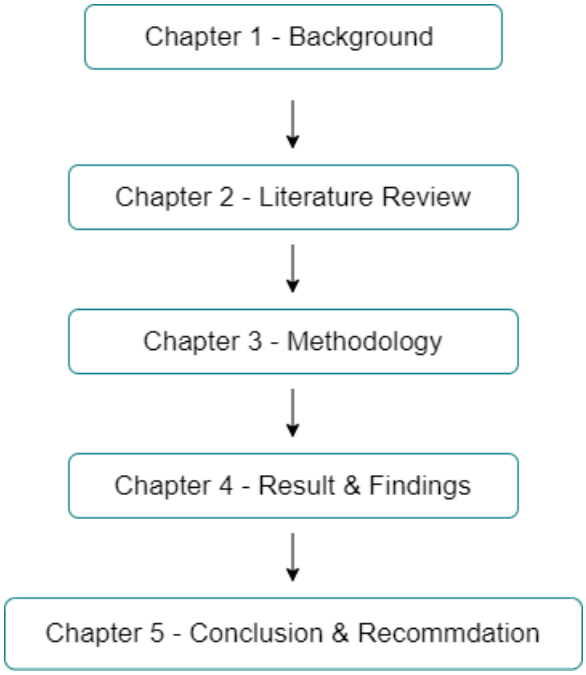

Figure 1.1: Thesis Organization

Chapter 2

LITERATURE REVIEW

2.1 Introduction

This chapter present insights into the two forecasting approaches used in financial forecasting and also reviews related works on various models used in exchange rate prediction across the world.

2.2 Financial Forecasting Approaches

A forecast represents an expectation about a future value. The expectation is constructed using information or dataset selected by the forecaster. Based on the dataset used by the forecaster, there are two pure approaches to financial forecasting.

2.2.1 Fundamental Approach

Fundamental approach is concerned with a range of data regarded as economic, financial and social variables that determines variable such as exchange rates. These includes news sentiment, inflation rate, interest rates, gross domestic product (GDP), power purchasing parity (PPP), etc.

2.2.2 Technical Approach

Technical approach in general rely on past price trends, patterns and waves as reasonable predictors. Technical analysis looks for the repetition of specific price patterns and then use them to generate future prices.

2.3 Related Works

2.3.1 Kernel Functions

Chiroma et al (2014) compared the performance of kernel functions for support vector machine on crude oil price data. The analysis of variance was used for validating the results. The findings emanated from the research indicated that the performance of the wave kernel function was statistically significantly better than the radial basis function, polynomial, exponential, and sigmoid kernel functions. However the computational efficiency of the wave activation function was poor compared with the other kernel functions in the study.

Shrawan et al (2013) in their research presented the effects of interaction between various Kernel functions and different feature selection techniques for improving the learning capability of support vector machine(SVM) in detecting email spams. The interaction of four Kernel functions of SVM i.e. "Normalised Polynomial Kernel (NP)", "Polynomial Kernel (PK)", "Radial Basis Function Kernel (RBF)", and "Pearson VII FunctionBased Universal Kernel (PUK)" with three feature selection techniques i.e. "Gain Ratio (GR)", "Chi-Squared (χ^2), and "Latent Semantic Indexing (LSI)" were tested on the "Enron Email Data Set". The results reveal that some of the proposed model shows poor performance for high dimensional data and vice versa.

Yekkehkhany et al (2014) developed a framework based on Support Vector Machines (SVM) for crop classification using polarimetric features extracted from multi-temporal Synthetic Aperture Radar (SAR) imageries. Several kernel functions are employed and compared in this study for mapping the input space to higher Hilbert dimension space. The method is applied to several UAVSAR L-band SAR images acquired over an agricultural area near Winnipeg, Manitoba, Canada. The experimental tests show an SVM classifier with

RBF kernel for three dates of data increases the Overall Accuracy (OA) to up to 3% in comparison to using linear kernel function, and up to 1% in comparison to a 3^{rd} degree polynomial kernel function.

kamruzzaman et al (n.d) in their paper investigated the effect of different kernel functions,namely linear,polynomial,radial basic and spline on prediction measured by several widely used performance metrics. The effects of regulation parameter is also studied. The prediction of six different foreign currency exchange rates against Australian dollar was performed and analyzed.

2.3.2 Exchange Rate Prediction

Goyal et al (2019) in their report; 'Machine Learning for Statistical Arbitrage: Using News Media to Predict Currency Exchange Rates' explored the application of machine learning for predicting bilateral Foreign Exchange Rates utilizing sentiment from news articles and prominent macroeconomic indicators. Variables such as exchange rate, gross domestic product (GDP), power purchasing parity (PPP) and 2000 articles per year from 1981-2016 were used to train the models. They also relied on utilizing the Latent Dirichlet Allocation (LDA) algorithm to cluster articles into topics and further understand the semantic meanings behind each topic and applying them to foreign exchange rates. Four different regression models were deployed in this report. Among all models, random forest regressor produced the least average error of 7.8%. At the end of the report the conclusion made was that although news article sentiment itself does not seem to outweigh established global macroeconomic indicators like GDP and PPP, it does offer major feature importance for certain countries.

Bofa et al (2019) employed artificial neural network to model the daily exchange rate between the Ghana cedis and the US dollar and forecasted future rates using the mode ensemble operator. Hidden nodes from 1 to 10 were observed to select the best model needed for the forecast. Among them,the

model with 1 hidden node was selected as it is shown to have the least cross-validated MSE. Again the forecasted rate showed an upward trend for the period of 123 days. As a result, the Cedi may depreciate compared to the Dollar for some time but may be stable within the range of 5.50 to 5.60 Ghana cedis for the forecasted period, in the absence of any structural changes. Based on the data obtained, the MLP (MSE=0.0056) network slightly outperform the ELM LASSO (MSE=0.0065) network model in forecasting the daily rate of the Ghana Cedi to the US Dollar. Finally, they advised the Ghanaian general public to limit their sense of taste for foreign goods as it has an adverse impact on the local currency value.

Nagpure (2019) adopted different deep learning model such as Support Vector Regressor (SVR), Artificial Neural Network (ANN), Long Short-Term Memory (LSTM), Neural Network with Hidden Layers to predict the exchange rate between worlds top traded currencies from daily data of January 2011 to December 2018. The result demonstrates the appropriateness of the Deep learning using SVR, ANN, LSTM and MLP-Neural Networks to the issue of multi-currency exchange rate prediction. The outcomes were profoundly promising; results showed that the average accuracy of the predicting model exceeds 99% .

He and Shen (2007) in their paper; 'Bootstrap Methods for Foreign Currency Exchange Rates Prediction', investigated the use of bootstrap methods for time-series prediction. however unlike the traditional single model like neural network, or support vector machine based timeseries prediction, they proposed the use of bootstrap methods to construct multiple learning models, and then use a combination function to combine the output of each model for the final predicted output. Neural network model is used as the base learning algorithm. Six major foreign currency exchange rates including Australia Dollars (AUD), British Pounds (GBP), Canadian Dollars (CAD), European Euros (EUR), Japanese Yen (JPY) and Swiss Francs (CHF) were used for the predic-

tion (base currency is US dollar). Simulations on exchange rate data from January 01, 2003 to December 27, 2006 on both daily prediction and weekly prediction indicated that the proposed method can significantly improve the forecasting performance compared to the traditional single neural network based approach.

Sekar et al. (2017) in their thesis analyzed the claim that Recurrent Neural Network(RNN) model with Empirical Mode Decomposition(EMD) is more accurate in forecasting than Autoregressive Integrated Moving Average (ARIMA). Daily data from April-2010 to March -2017 is used in the study. Upon evaluation using root mean square error(RMSE), RNN+EMD gave smaller error value as compared to ARIMA. However, an observation made from the study was that ARIMA model cannot handle larger dataset regardless of how the data is trained.

Yasir (2019) tested the validity of a deep-learning-based model over three currency exchange rates, which are Pak Rupee to US dollar(PKR/USD), British pound sterling to US dollar(GBP/USD), and HongKong Dollar to US dollar (HKD/USD). The proposed model also incorporated the ongoing social and political event sentiments in predicting the exchange rate. Moreover, since the currency market is heavily dependent upon highly volatile factors, gold and crude oil prices were also considered for exchange rate forecasting. The studies also showed the importance of incorporating investor sentiment of local and foreign macro-level events for accurate forecasting of the exchange rate. Approximately 5.9 million tweets were processed to extract major events sentiment. The results indicated that the deep-learning based model is a better predictor of foreign currency exchange rate in comparison with statistical techniques normally employed for prediction. The results also presented evidence that the exchange rates of all the three countries are more exposed to events happening in the US.

Abreu et al (2018) proposed a hybrid system using machine learning techniques together with genetic algorithms to investigate the claim that technical analysis is used to discover investment opportunities. In this work, an architecture for automatic feature selection is proposed to optimize the cross-validated performance estimation of a Naive Bayes model using a genetic algorithm. The proposed architecture improves the return on investment of the unoptimized system from 0.43% to 10.29% in the validation set. The features selected and the model decision boundary are visualized using the algorithm t-Distributed Stochastic Neighbor embedding.

Rojas and Herman (n.d.) in their report, used machine learning methods to forecast the US Dollar against the Mexican Peso exchange rate. An innovative framework was used to find the best possible performance. Firstly, a market variable dataset and a fundamental dataset was used to train ML algorithms. Secondly, binary classification experiments and continuous target experiments were conducted to produce an output of binary long/short signal on which simple trading strategy was executed. The results suggest that continuous target prediction outperforms binary classification not only in terms of accuracy, but also in terms of specificity and sensitivity. The result obtained also indicates that SVM produce the best result in the binary classification case and Ridge regression in the continuous target case, both in terms of accuracy and cumulative profits but the fundamentals dataset yields poor results.

Lin et al (2006) present a hybrid model for predicting the occurrence of currency crises by using the neuro-fuzzy modeling approach. The model integrated the learning ability of neural network with the inference mechanism of fuzzy logic. The empirical results indicated that the proposed neuro-fuzzy model leads to a better prediction of crises. Significantly, the model can also construct a reliable causal relationship among the variables through the obtained knowledge base. Compared to the traditionally used techniques such as logit,

the proposed model can thus lead to a somewhat more prescriptive modeling approach towards finding ways to prevent currency crises.

Chander et al (2015) present five different neural network system for foreign exchange rate prediction. The authors observe the performance of the proposed method for Pound Sterling (PS), United States Dollar (USD), EURO and Japanese Yen (JYEN) against Indian Rupee (INR) using their historical data. The authors also evaluated the forecasting performance of the proposed system by using statistical metric. From the results, it is confirmed that the new approach provided an improve technique to forecast foreign exchange rate. Among the five models, Levenberg-Marquardt based model outperforms other models and attains comparable results. It also demonstrates the power of the proposed approach and produces more accurate prediction. In conclusion, the proposed scheme can improve the forecasting performance significantly when measured on three commonly used metrics.

Ahmad et al (n.d) said in their report that SVM perform better than ANN. Supporting their claims the authors stated that, in SVM data is not overfitted as compared to ANN which does not overfit data unless ANN cross-validation is applied.

Nanayakkara et al (2014) stated that ANN based model performs better when compared with the Generalized Auto-Regressive Conditional Heteroskedasticity (GARCH) model in predicting the exchange.

Chapter 3

METHODOLOGY

3.1 Machine Learning

Machine learning is a method of data analysis that automates analytical model building, based on the idea that, the system can learn from data, identify patterns and make decisions with minimal human intervention. Machine learning is basically about extracting knowledge from data. It is a research field at the intersection of statistics, artificial intelligence, and computer science and is also known as predictive analytics or statistical learning (Machine Learning: What it is and why it matters, n.d).

The application of machine learning methods has in recent years become ubiquitous in everyday life. From automatic recommendations of friends and ads on social media site like Facebook and Linkedln to which movies to watch, what food to order, or which products to purchase. Vision and language processing, as well as forcasting things like stock market trends and weather, are all some applications of machine learning algorithm (Müller and Guido, 2016, p.1).

Outside from these applications, ML is increasingly gaining attention in forex trading. With majority of ML model today focus on making predictions, the technology has lend itself to supporting the one area where finance must make its best informed prediction; forecasting. There are three main types of ML algorithm namely Supervised learning, Unsupervised learning and Reinforcement learning. In this study, supervised learning algorithm was considered.

3.1.1 Supervised Learning

Supervised machine learning algorithm is design to learn by examples. It is called "supervised" learning because it originates from the idea that, training this type of algorithm can be thought of as teacher supervising the whole process (Browlee, 2019). With supervised learning algorithm, the training data will consist of both input and output(target variable). During the training, the algorithm will search for patterns in the data that correlate with the desired outputs. After training, the machine is provided with an unseen input so that the algorithm can produce a correct outcome or output based on the prior training data. Some examples are Support Vector Machine(SVM), Decision tree, Radom forest and K-Nearest neighbour. Under supervised learning, support vector machine regression will be analyzed and deployed on the cedi-dollar currency rate.

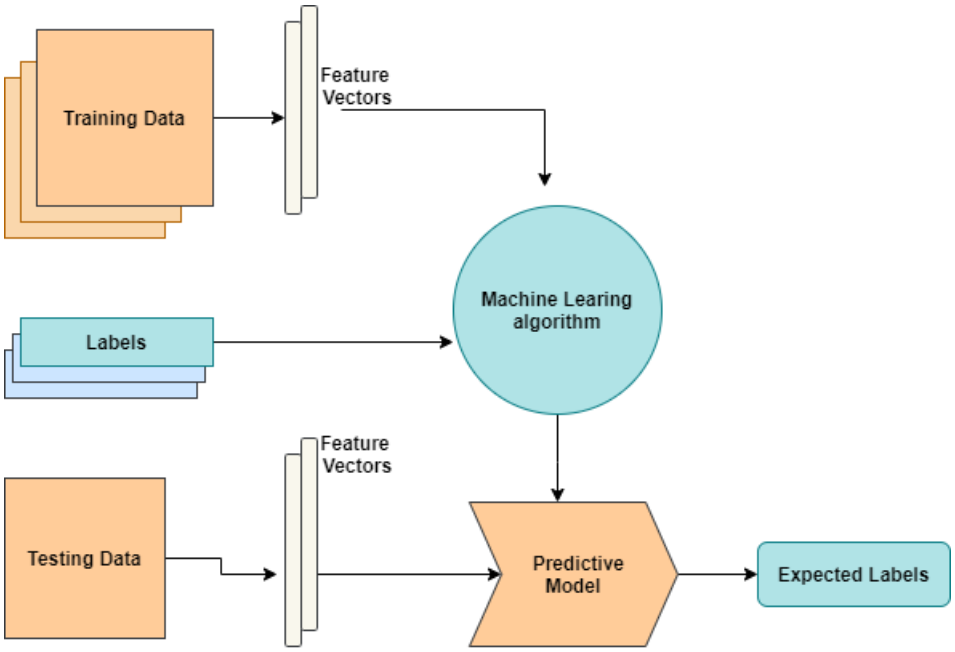

Figure 3.1: Supervised Learning Model

3.1.2 Support Vector Machine

Support vector machine is a supervised linear model for classification and regression problems. SVM originated from Vapnik's statistcal learning theory. The idea of SVM is simple, the algorithm creates a line or a hyperplane which separates the data into classes. In this project we will implement the regression aspect of SVM but before we shall consider the mathematical formulation of the classification, the maximal margin.

Architecture Support Vector Machine: Classification

Given a training data set (x_1, y_i) for $i = 1, ..., n$ where $x_i \in \mathbb{R}^d$ and $y_i \in \{-1, 1\}$ SVM tries to find a hyperplane that separates the data into classes .

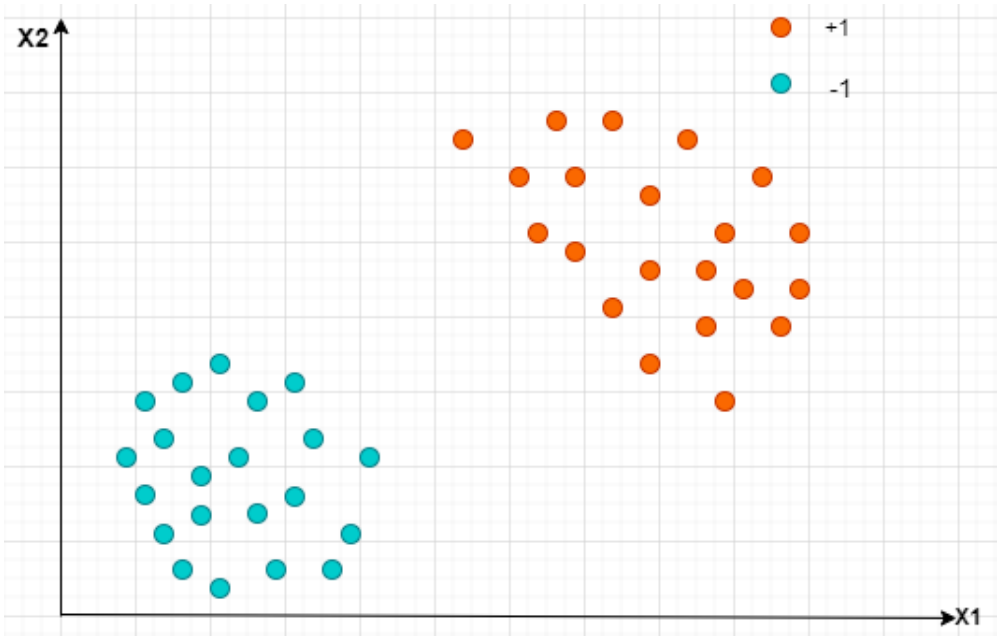

Figure 3.2: Linear Separable

The equation of a hyperplane can be written as $w^T x + b = 0$, where $w^T x$ is the dot product of the two vectors (w, x). Hence for **classification** we can define a hypothesis function g such that

$$g(x) = \begin{cases} +1 & \text{if } w^T x + b > 0 \\ -1 & \text{if } w^T x + b < 0 \end{cases} \tag{3.1}$$

which is equivalent to

$$g(x) = y(w^T x_i + b)$$

Now if we can find the w, b corresponding to the hyperplane that separates the data, then we can have a decision function for classifying the data set. However in cases like this, we can have an infinitely many hyperplanes, that can separate the two classes.

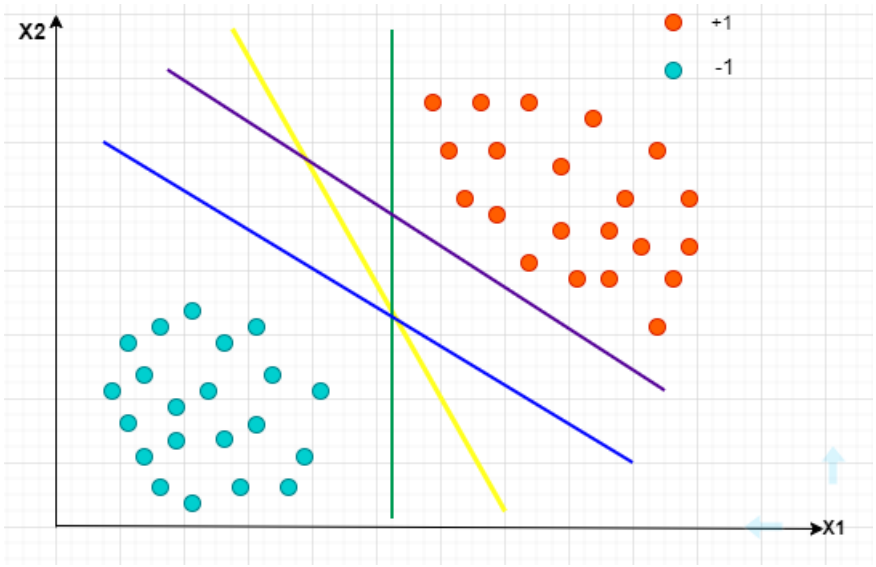

Figure 3.3: Infinite hyperplanes

Now consider figure 3.3, all the colored lines seem to separate the data. However, we are looking for the best line that seprate the classes of data in such a way that, if we have a new point x^*, we can say with high confidence that, x^* belong to one of the two classes. In order words, we can say that our goal is to find a more generalized separator.

According to the SVM algorithm, finding the best separating hyperplane is the same as finding the hyperplane that maximizes the margin. The distance

between the hyperplane and the points closest to the hyperplane from both classes is computed. These points are called support vector. The hyperplane for which the margin is maximum is the optimal hyperplane.

Margins play a key role in finding the optimal hyperplane. Figure 3.4 illustrate how to find the margin between a point and the hyperplane. The margin in question here is **m**, which is the perpendicular distance from x_1 to the hyperplane H_0. The vector point x_1 is the support vector, r is the difference between vector x_0, x_1 thus,

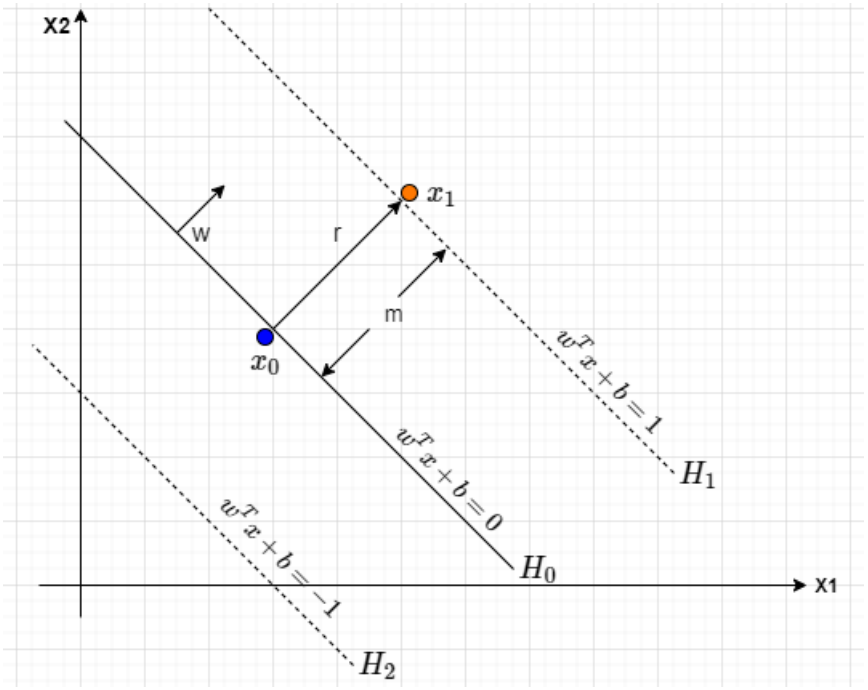

Figure 3.4: Geometric Margin: Distance between a point and a hyperplane

$r = x_1 - x_0$. The vector r is in the same direction as vector **w**, and so they share the same unit vector $\frac{w}{||w||}$, hence we multiply m by the unit vector of r, we can define r as $r = m\frac{w}{||w||}$. By substitution we get

$$x_0 = x_1 - m\frac{w}{||w||} \tag{3.2}$$

Now the point x_0 is on the hyperplane H_0 which implies that it satisfies the

equation of the hyperplane hence we have :

$$\begin{aligned} w^T x_0 + b &= 0 \\ w^T(x_1 - m\frac{w}{||w||}) + b &= 0 \\ w^T x_1 - m\frac{w \cdot w}{||w||} + b &= 0 \\ w^T x_1 - m\frac{||w||^2}{||w||} + b &= 0 \\ w^T x_1 - m||w|| + b &= 0 \\ m &= \frac{w^T x_1 + b}{||w||} \end{aligned} \tag{3.3}$$

In the same way if x_1 is on the hyperplane H_2 then,

$$m = -\frac{w^T x_1 + b}{||w||} \tag{3.4}$$

In general the margin is:

$$m = y\frac{(w^T x_1 + b)}{||w||} \tag{3.5}$$

The margin in search here is the geometric margin. By definition if H is an optimal hyperplane that separate the data (x_i, y_i) for $i = 1, ..., n$ then, the geometric margin of this optimal hyperplane is

$$\min_i \quad d(x_i, H),$$

the distance from the hyperplane H to the closest data point. The optimal separating hyperplane is the one that correctly classifies all data while being farthest away from the data points. In this respect, it is said to be the hyperplane that maximizes the margin (Lauer, 2014).

$$\underset{w,b}{argmax} \quad min \quad y\frac{(w^T x + b)}{||w||} \tag{3.6}$$

From equation 3.4, $w^T x_1 + b = 1$ when, x_1 is on hyperplane H_1. By substituting into equation 3.5, the margin $m = \frac{1}{||w||}$. In the same way when x_1 is on the

hyperplane H_2 the margin is $\frac{1}{||w||}$. Combining the two margins, the total margin becomes

$$\frac{2}{||w||} \tag{3.7}$$

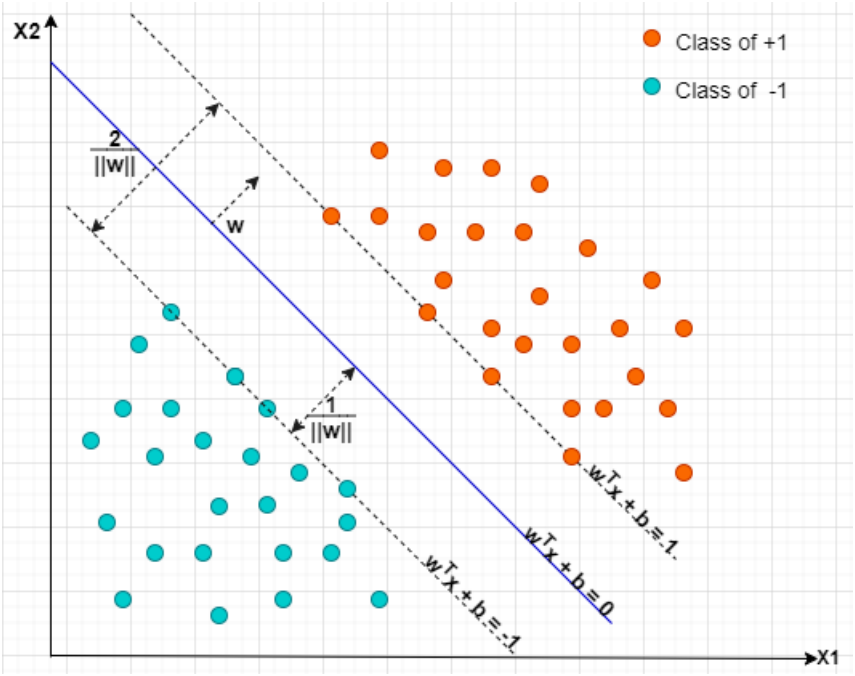

Figure 3.5: Maximum Margin Separating Hyperplane

As stated earlier, optimal separating hyperplane is the one that maximizes the margin hence,

$$max \quad \frac{2}{||w||} \tag{3.8}$$

however, one can easily see that, the smaller the value of w, the larger the margin becomes. And so to find the optimal hyperplane one has to solve the following quadratic programming problem

$$\min_{w,b} \quad \frac{1}{2}||w||^2 \tag{3.9}$$

under the constraints of inequality type

$$y_i(w^T x_i + b) \geq 1, \quad i = 1, 2, ..., n \tag{3.10}$$

Linearly Inseparable

The procedure developed so far requires the data to be linearly separable. However instances where there is an outlier or data is nonlinear, the optimization problem proposed above fail to perform (Kowalczyk, 2017). SVM algorithm introduce the concept of soft margin and kernel functions in this case.

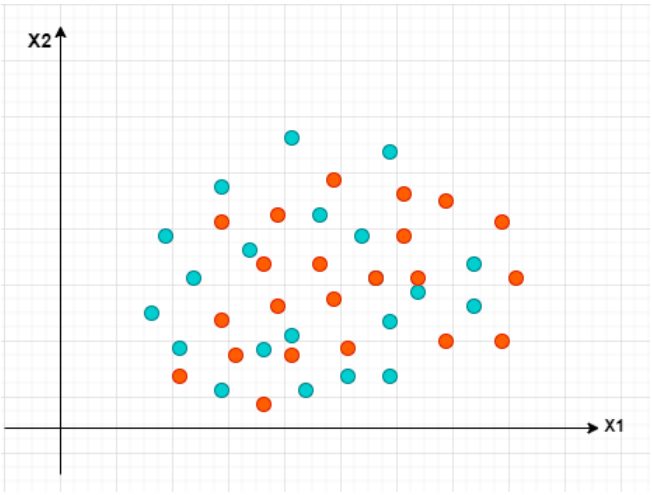

Figure 3.6: Non-linearly separable data

From Figure 3.6, it is clear that there is no specific linear decision boundary that can perfectly separate the data, i.e, the data is not linearly separable.

Soft Margin SVM

Soft margin formulation is based on a simple premise; allow SVM to make a certain number of mistakes and still keep the margin as wide as possible so that other points can still be classified correctly. By modification, the objective function becomes

$$\frac{1}{2}||w||^2 + \text{C}(\#\text{ of mistakes}) \tag{3.11}$$

The constant C is a regularization parameter. This parameter controls how SVM should handle errors i.e deciding the trade-off between maximizing the margin and minimizing the mistakes.

For large value of C, the optimization will focus more on avoiding misclassification by choosing a smaller margin hyperplane which does a better work of getting all training data point classified correctly. Conversely, for a small value of C, the optimization will focus more on choosing a separating hyperplane which maximizes the margin even if that hyperplane misclassifies some point.

The modified mathematical formulation then becomes, for every point x_i we introduce a slack variable ξ, which measures the distance of x_i from the corresponding class margin when x_i is on the wrong side of the margin otherwise zero(0).

$$\begin{aligned} \underset{w,b}{min} \quad & \frac{1}{2}||w||^2 + C\sum_{i=1}^{n}\xi_i \\ s.t \quad & y_i(w^T x_i + b) \geq 1 - \xi_i \\ \xi_i \geq 0 \quad & for \quad any \quad i = 1, ..., n \end{aligned} \tag{3.12}$$

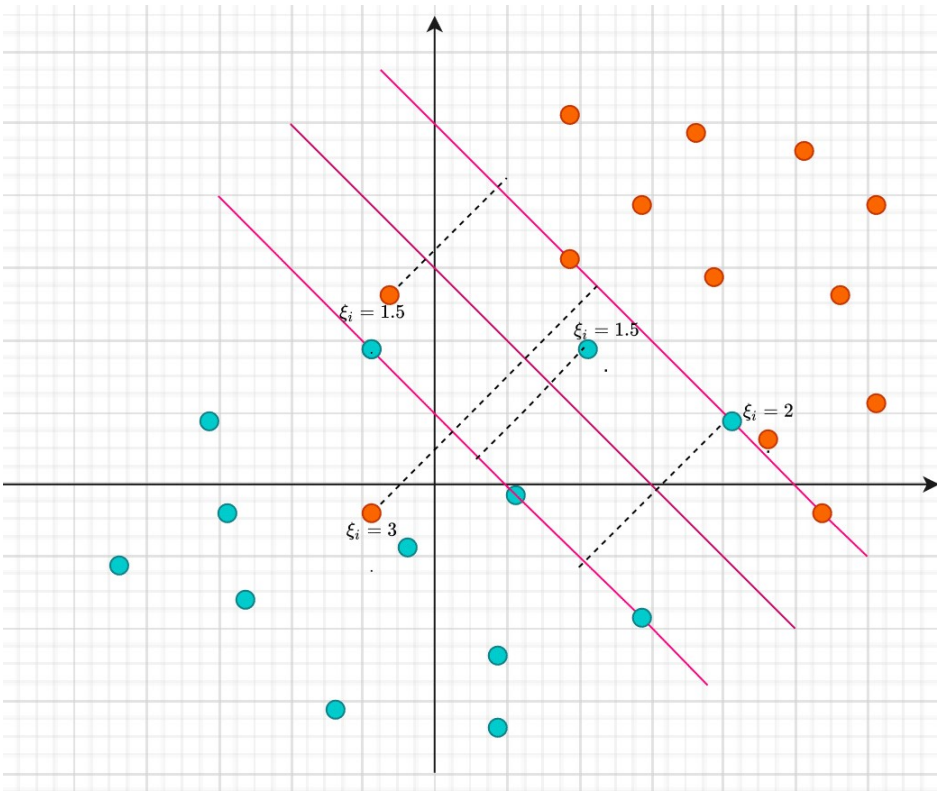

Figure 3.7: Soft Margin

Kernel Function

The key to support vector machine is the introduction of kernel function. When the data set is truly not separable, SVM is unable to find a linear hyperplane that can separate the input data into classes. However when the data set is mapped to a high dimensional space, the formation of new data sets is more easily separated, but the computational cost for this method is huge. The introduction of kernel function helps map data set to a high dimensional space without increasing the computational cost. A kernal function is defined as a function that corresponds to the dot product of two feature vectors in some expanded feature space.

$$K(x_i, x_j) = \phi(x_i) \cdot \phi(x_j) \tag{3.13}$$

The kernel functions used in this study are the followings.

$$\begin{aligned}
\text{Linear}: \; & k(x_i, x_j) = (x_i \cdot x_j) \\
\text{Polynomial}: \; & k(x_i, x_j) = (x_i \cdot x_j + 1)^d, \quad d = degree \\
\text{Polynomial Kernel}: \; & k(x_i, x_j) = exp\left(\frac{||x_i - x_j||}{2\gamma^2}\right), \quad \gamma = gamma
\end{aligned}$$

3.1.3 Support Vector Regression SVR

The principles of SVM for classification can easily be extended to that of SVM for regression. However, unlike the classification, the output is a real number i.e. $y \in \mathbb{R}$.

For most linear regression model, the main objective is to minimize the errors in order to obtain the most accurate model, and as such most of these regression models adopts different loss function to achieve this objective. Take simple linear regression for example, SLR uses the widely used technique called the method of least square to find its parameters that minimizes the sum of

squared errors. The objective function is

$$min \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \tag{3.14}$$

where y_i is the target and $\hat{y}_i$ is the predicted value.

Support Vector Regression(SVR) just like other linear regression model also adopt the principle of error minimization. However unlike simple linear regression, SVR gives us the flexibility to define how much error is acceptable in our model, and fit an appropriate line (or hyperplane) to fit the data. In support vector regression, Vapniks proposed the use of ε-insensitive loss function which symmetrically forms a flexible tube of minimal radius around the estimated function, such that the absolute values of the error, less than a certain degree is ignored. In this manner, points within the tubes are not penalized but those outside receives penalty (Khanna and Awad, 2015).

Now consider a simple case where f is a linear function, the basic problem is to find a function $f \in F$ that minimizes the risk function (Smola et al, 1996). The risk in question is the amount of prediction error that can be expected when using a model, and the risk function is the expected value of the loss function.

$$R[f] = \int \ell(y - f(x)) dP(x, y) \tag{3.15}$$

ℓ is the loss function. The risk function $R[f]$, cannot be directly evaluated because the distribution $P(x, y)$ is unknown. But instead we can compute an approximation called the empirical risk (Smola et al, 1996). The unknown coefficient w is determined by minimizing the sum of empirical risk and the complexity term $||w||$ (Müller et al, 1997).

$$
\begin{aligned}
&min \quad R_{emp}(f) + \frac{1}{2}||w||^2 \\
&\frac{1}{n}\sum_{i=1}^{n}(y_i - (w^T x_i + b)) + \frac{1}{2}||w||^2
\end{aligned} \tag{3.16}
$$

The empirical risk ($R_{emp}(f)$), is the average of the loss function of the training set. Adopting ϵ-insensitive loss function,

$$
\ell = \begin{cases} \infty & \text{if } |f(x_i) - y_i| \geq \epsilon \\ 0 & \text{otherwise} \end{cases} \tag{3.17}
$$

the $R_{emp}(f)$ can assumes values of 0 or ∞ in equation 3.17. However to get the flattest linear function, $R_{emp}(f)$ is set to zero(0). Hence in general, finding the **flattest** linear function corresponds to minimizing $\frac{1}{2}||w||^2$ (Smola et al, 1996).

Based on equation (3.17) and (3.16), and adopting a soft-margin approach similar to that employed in SVM classification, slack variable ξ, ξ^* is added to guard against outliers. These variable determine how many points can be tolerated outside the tube. The optimization problem below is obtained

$$
\begin{aligned}
\underset{w,b}{minimize} \quad & \frac{1}{2}||w||^2 + C\sum_{i=1}^{n}\xi_i + \xi_i^* \\
\text{subject to} \quad & y_i - (w^T x_i + b) \leq \varepsilon + \xi_i \quad i = 1, ..., n \\
& w^T x_i + b - y_i \leq \varepsilon + \xi_i^* \quad i = 1, ..., n \\
& \xi_i, \xi_i^* \geq 0 \quad i = 1, ..., n
\end{aligned} \tag{3.18}
$$

The solution to this constrained quadratic optimization problem can be solved by using the method of lagrange multiplier. The basic idea of the lagrange multiplier is to convert a constrained problem into a form such that, the derivative test of an unconstrained problem can be applied ("Lagrange multiplier", 2020).

This is formed by subtracting the product of the lagrange multiplier and the constraints from the objective function.

$$L = \frac{1}{2}||w||^2 + C\sum_{i=1}^{n}\xi_i + \xi_i^* - \sum_{i=1}^{n}\alpha_i(\varepsilon + \xi_i - y_i + w^T x_i + b) \\ - \sum_{i=1}^{n}\alpha_i^*(\varepsilon + \xi_i^* + y_i - w^T x_i - b) - \sum_{i=1}^{n}(\lambda_i\xi_i + \lambda_i^*\xi_i^*) \tag{3.19}$$

The minimum of equation 3.19 is found by solving for the gradient. Which implies taken the partial derivative with respect to the variables (w, b, ξ_i, ξ_i^*) and setting derivatives to zero. But then, the partial derivative of the lagrange multipliers have to satisfy the positive constraints i.e $\alpha_i, \alpha_i^*, \lambda_i, \lambda_i^* \geq 0$ (Khanna and Awad, 2015).

$$\partial_w L = w - \sum_{i=1}^{n}(\alpha_i - \alpha_i^*)x_i = 0 \tag{3.20}$$

$$\partial_b L = \sum_{i=1}^{n}(\alpha_i^* - \alpha_i) = 0 \tag{3.21}$$

$$\partial_{\xi_i^*} L = C - \alpha_i^* - \lambda_i^* = 0$$

$$\partial_{\xi_i} L = C - \alpha_i - \lambda_i = 0 \tag{3.22}$$

Substituting equation (3.20), (3.21) and (3.22) into (3.19) yields the dual optimization problem (Basak et al, 2007). This dual optimization problem is use to obtain the lagrange multipliers α_i and α_i^*.

$$\underset{\alpha,\alpha^*}{\text{maximize}} \quad \sum_{i=1}^{n}(\alpha_i - \alpha_i^*) - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)x_i^T x_j - \varepsilon\sum_{i=1}^{n}(\alpha_i + \alpha_i^*)$$

$$\text{subject to} \quad \sum_{i=1}^{n}(\alpha_i - \alpha_i^*) = 0 \quad \text{and} \quad \alpha_i, \alpha_i^* \in [0, C] \tag{3.23}$$

Equation (3.20) can be written as:

$$w = \sum_{i=1}^{n}(\alpha_i - \alpha_i^*)x_i \quad \text{thus} \quad f(x) = \sum_{i,j=1}^{n}(\alpha_i - \alpha_i^*)x_i^T x_j + b \tag{3.24}$$

Now if we define a kernel as $K(x_i, x_j) = x_i^T x_j$ we can rewrite equation $f(x)$ as

$$f(x) = \sum_{i,j=1}^{n}(\alpha_i - \alpha_i^*)K(x_i, x_j) + b \tag{3.25}$$

The term b is calculated from Karush-Kuhn-tucker(KKT) conditions which states that, the product of the lagrange multiplier or the dual variables and the constraints is equal to zero (Smola et al, 1996; Khanna and Awad, 2015).

$$\begin{aligned} \alpha_i(\varepsilon + \xi_i - y_i + w^T x_i + b) &= 0 \\ \alpha_i^*(\varepsilon + \xi_i^* - y_i - w^T x_i - b) &= 0 \end{aligned} \tag{3.26}$$

$$\begin{aligned} \lambda_i \xi_i &= 0 \\ \lambda_i^* \xi_i^* &= 0 \end{aligned} \tag{3.27}$$

Following the above equations, the term b is computed as

$$\mathrm{b} = \begin{cases} y_i - w^T x_i - \varepsilon & \text{for } \alpha_i \in [0, C] \\ y_i - w^T x_i + \varepsilon & \text{for } \alpha_i^* \in [0, C] \end{cases} \tag{3.28}$$

3.2 Data

The data for this research contain daily exchange rates of the Ghana cedis to the US dollar from January 2014 to April 2020. The data was obtained from *https://www.m.investing.com/currencies*.

3.3 Data Preprocessing

Data Pre-processing is one of the most important step to consider when using data for training any machine learning algorithm. Here we preprocessed the data using NumPy and Pandas libraries. Different functions on Pandas Dataframe was called on the data to check and remove missing values. The data was then differenced as it violated the ADF test for stationarity. The differencing technique transforms the original data(x_t) into a new series y_t, where

$$y_t = x_t - x_{t-1} \qquad \text{for} \quad t = 2, 3, ... \tag{3.29}$$

Literature reveals that, moving averages and past values can be considered as inputs to support vector regression in the process of predicting exchange rates (kamruzzaman et al, n.d; Nanayakkara et al, 2014). Hence to suit the standard linear and nonlinear machine learning algorithms on the problem, we re-framed the differenced data to supervised learning using moving average and past lag.

Moving average indicators such as MA5 (moving average on 5 days), MA10, MA20, MA60, MA120 and current data X_t were used to build the SVR model. Exchange rate for the period $t + 1$ is predicted. With respect to the moving average, exponential moving average(EMA) is preferred over simple moving average(SMA) as EMA ia said to give more weight to the latest data than SMA.

$$EMA_t = EMA_{t-1} * \left(1 - \frac{2}{k}\right) + \left(x_t * \frac{2}{k}\right) \tag{3.30}$$

The available data was divided into training and testing set. In general 1641 daily exchange rate was considered of which the first 1312 was used as training set and the remaining 329 as testing data to evaluate the model.

3.4 Parameter Tuning

As stated in section 1.4 the objective, other aim of this study is to investigate the variation of performance with respect to regularization parameter C and epsilon(ε). In this experiment, C was varied from a small value (0.01) to a large value (10^2) and ε was also set to $0.01, 0.1, 100$. The degree of the polynonial and rbf gamma were arbitrarily chosen to be 1.

3.5 Evaluation Metrics

The performance of the method is evaluated using Root Mean Square Error (RMSE), Mean Absolute Error (MAE), Mean Squared Error(MSE) and the R-squared (R^2).

$$
\begin{aligned}
MAE &= \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \\
MSE &= \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \\
RMSE &= \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \\
R^2 &= 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y_i})^2} \\
\text{where} \quad \bar{y_i} &= \frac{1}{n}\sum_{i=1}^{n}(y_i)
\end{aligned}
\tag{3.31}
$$

3.6 Tool

This section of the methodology gives a brief description of the tool used and some libraries imported which aided in the implemention of SVR algorithm. Python is used as the basic programming language for developing the model.

3.6.1 Python

Python has emerged over the last couple decades as a first-class tool for scientific computing tasks, including the analysis and visualization of large datasets. The handy bunch of libraries has also made python popular among developers, in the area of artificial intelligence and machine learning.

3.6.2 Numpy

Numpy is an array-processing package. It provide a high-performance, multidimensional array object, and tool for working with these arrays. It is the fundamental package for scientific computing with python.

3.6.3 Pandas

Pandas is a python software library for data manipulation and analysis. In particular, it offers data structures, and operations for manipulating numerical tables and time series.

3.6.4 Matplotlib

Matplotlib is a comprehensive library for creating an interactive visualizations in python. It is built on Numpy arrays and designed to work with the broder Scipy stack.

3.6.5 Seaborn

Seaborn is a python visualization library built on top of matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.

3.6.6 Jupyter Notebook

The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative test. It uses includes data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning and many more.

Chapter 4

RESULTS

In this section we present the result and the performance of the three kernel functions as well as the parameter C and epsilon .

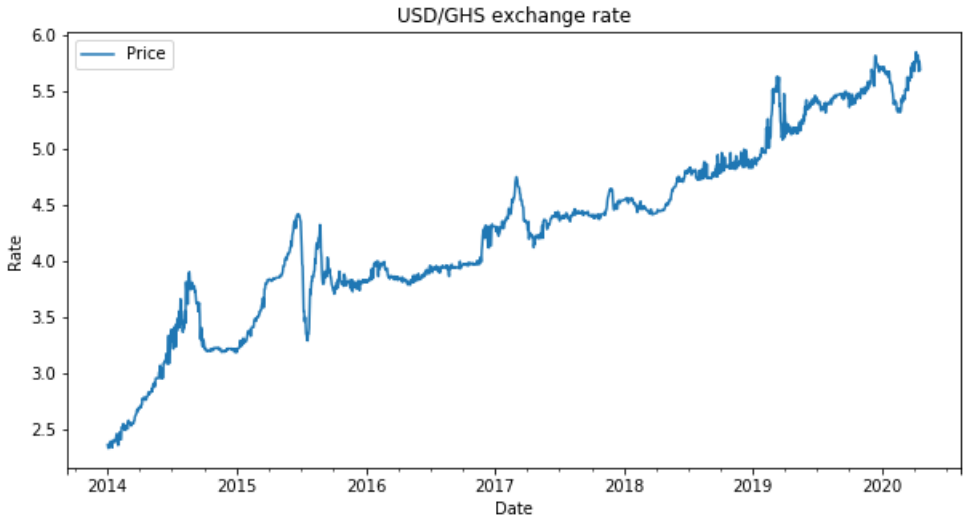

Figure 4.1: Time series Plot of the data

Figure 4.1 is the graph showing the plot of the data. The daily exchange rate comprises of 1643 observation with the minimum and maximum rates being 2.335 and 5.855 respectively. The data has a mean of 4.267410 and a standard deviation value of 0.803632. The time series data shows traces of an increasing trend. Based on the ADF test(p-value = 0.52), the time series plot shows non stationary variation and hence does require the application of differencing to achieve stationarity. The difference plot is seen in figure 4.2

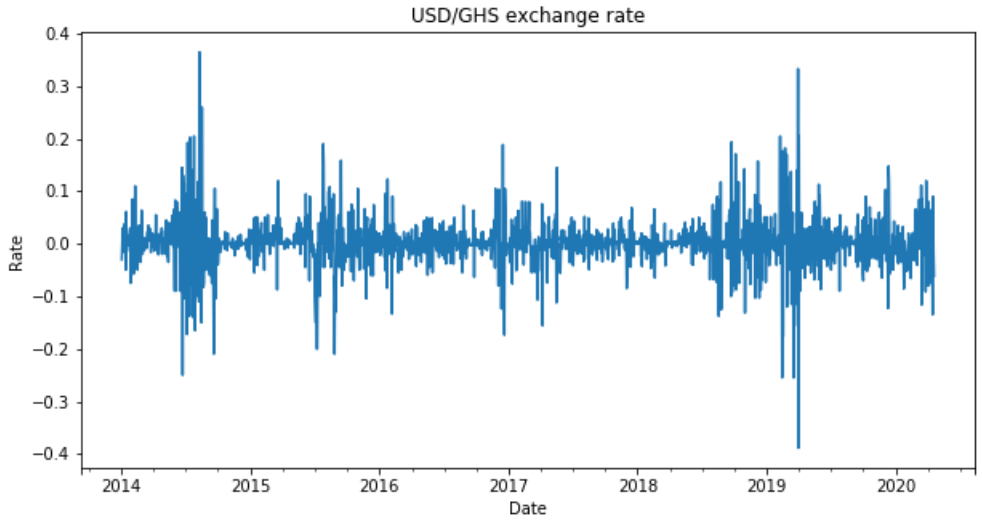

Figure 4.2: Difference Plot of the time series data

4.1 Analysis of Kernel functions

Table 4.1 and Figure 4.3 shows the result of the three kernel function. Table 4.1 shows that, radial basis kernel yields the best result wheres the polynomial kernel produce the least result among the three kernel. The MSE of the linear, polynomial and radial basics function are 0.00014, 0.000182, and 0.000134 respectively. This small values of the metrics indicates how close the predicted values are to the actual values. The maximum metrics difference of the radial basis kernel and the linear kernel is about 0.0006% indicating how closed the performance of the linear kernel was to that of the radial basis kernel. The prediction accuracy of the linear and radial basis kernel were 96.1% and 96.4% respectively. The polynomial kernel followed with an accuracy of 95.1%.

Table 4.1: SVR Error and Confidence

SVR Kernel	R^2	MAE	MSE	RMSE
Linear	0.962740	0.007876	0.000140	0.011833
Poly	0.951658	0.009004	0.000182	0.013478
RBF	0.964385	0.007670	0.000134	0.011569

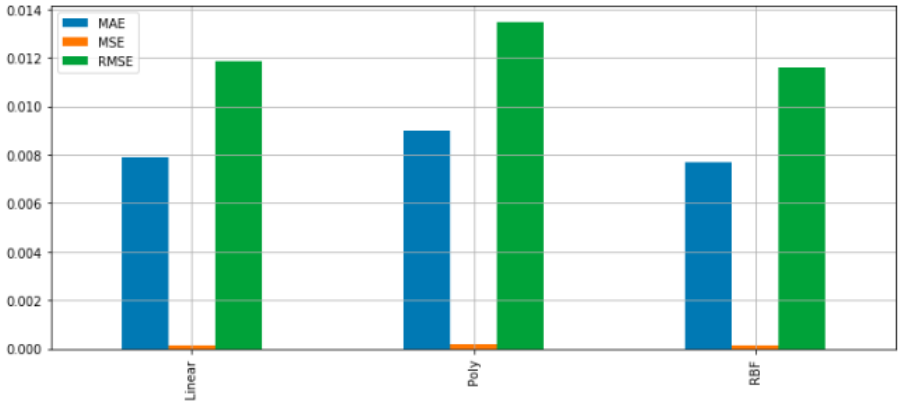

Figure 4.3: Metrics performance

Figure 4.4 to 4.7 shows the plots of predicted data against actual data

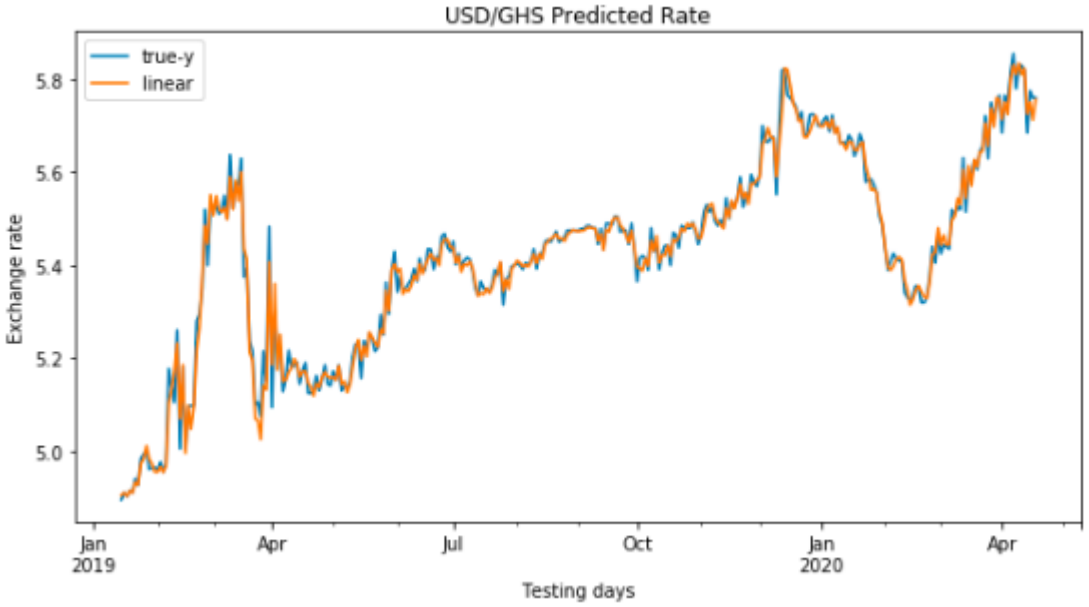

Figure 4.4: Linear Kernel

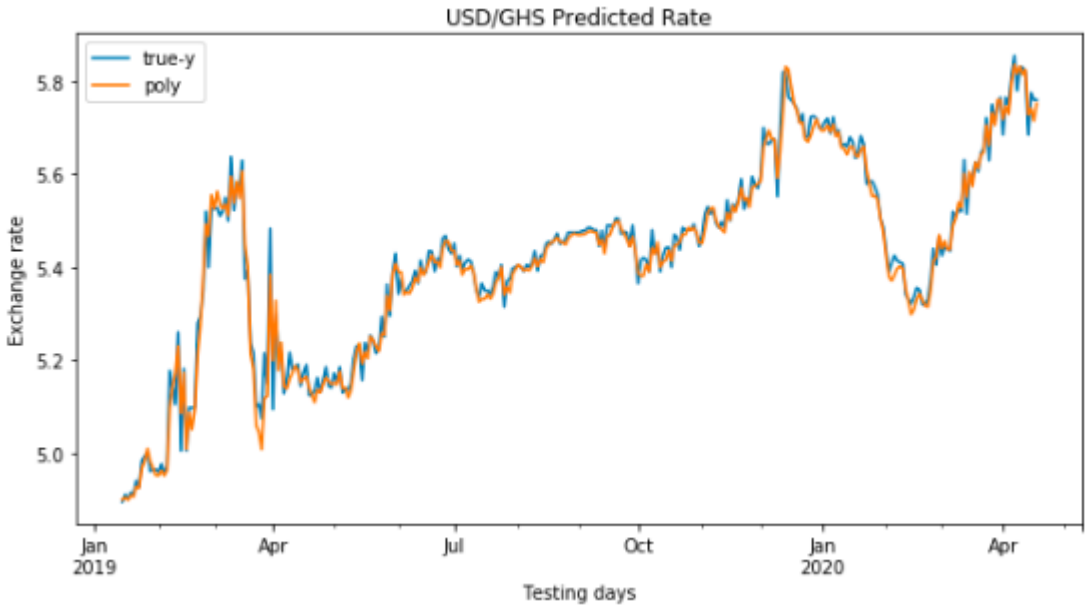

Figure 4.5: Polynomial Kernel

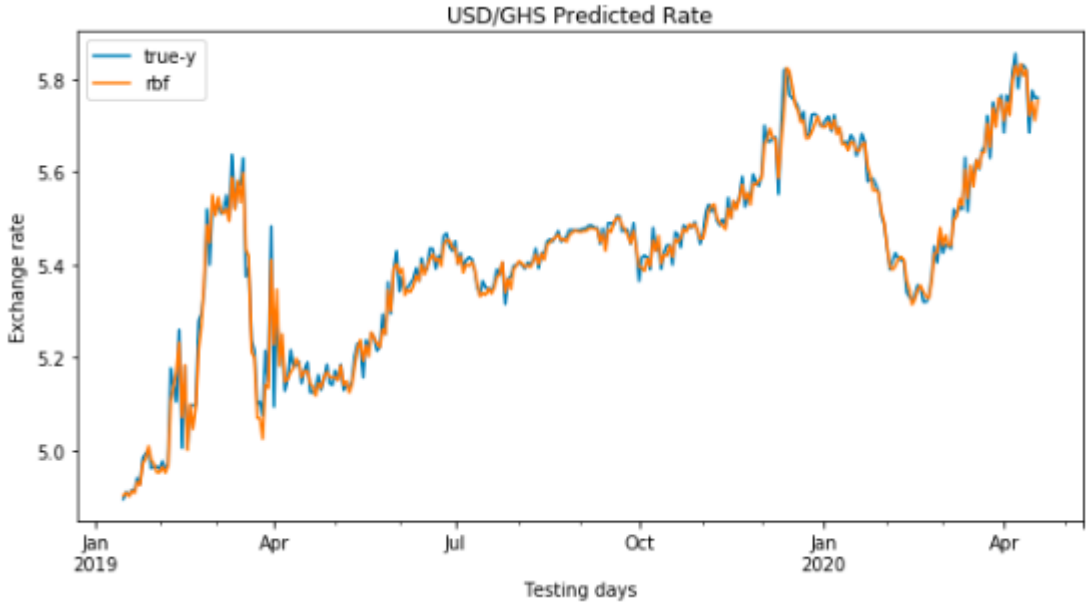

Figure 4.6: Radial Basis Kernel

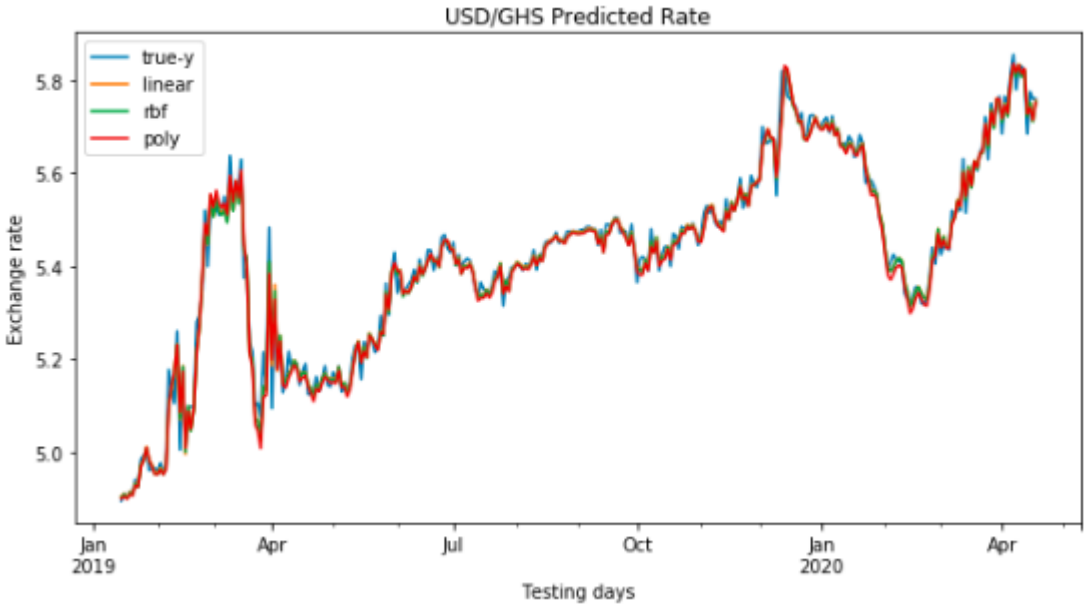

Figure 4.7: All Kernel Function

Table 4.2 to 4.4 shows the variation performance of the parameter C, and ε. When ε is held constant and C increases from 0.01 to 100, the accuracy of the model also increases. Opposite behaviour is observed when C is held constant. The model performance decreases as ε increases. thus the best result is obtained when $\varepsilon = 0.01$ for all kernels. This indicates that, the choice of C and ε can reduce or improve the performance of a model. The best performance for all kernel function on the data is achieved, when C= 100 and $\varepsilon = 0.01$. This represent the highlighted row.

Table 4.2: Linear Kernel

C	epsilon	test-R^2	train-R^2	MAE	MSE	RMSE
	0.001	0.115467	0.101797	0.034903	0.003324	0.057653
0.01	0.010	0.122591	0.101386	0.034809	0.003297	0.057421
	0.100	0.036851	0.029148	0.036414	0.003619	0.060161
	0.001	0.582248	0.521323	0.024392	0.001570	0.039621
0.1	0.010	0.582168	0.516142	0.024417	0.001570	0.039625
	0.100	0.234506	0.205869	0.032500	0.002877	0.053634
	0.001	0.897516	0.838906	0.013178	0.000385	0.019624
1	0.010	0.886908	0.824358	0.013837	0.000425	0.020615
	0.100	0.638986	0.547899	0.023806	0.001357	0.036832
	0.001	0.946389	0.914633	0.009542	0.000201	0.014194
10	0.010	0.942204	0.906533	0.009985	0.000217	0.014737
	0.100	0.841115	0.744938	0.016967	0.000597	0.024435
	0.001	0.962740	0.944173	0.007876	0.000140	0.011833
100	0.010	0.961721	0.941520	0.007992	0.000144	0.011994
	0.100	0.884590	0.816422	0.014372	0.000434	0.020825

Table 4.3: Plynomial Kernel

C	epsilon	test-R^2	train-R^2	MAE	MSE	RMSE
	0.001	0.021346	0.018210	0.036707	0.003678	0.060643
0.01	0.010	0.021848	0.017966	0.036729	0.003676	0.060628
	0.100	0.004930	0.003446	0.036997	0.003739	0.061150
	0.001	0.179437	0.160140	0.033642	0.003084	0.055529
0.1	0.010	0.186015	0.159228	0.033542	0.003059	0.055306
	0.100	0.055890	0.045632	0.036047	0.003548	0.059563
	0.001	0.705382	0.635341	0.020847	0.001107	0.033273
1	0.010	0.694952	0.623532	0.021187	0.001146	0.033857
	0.100	0.305595	0.283436	0.030992	0.002609	0.051083
	0.001	0.915554	0.863621	0.012005	0.000317	0.017814
10	0.010	0.908114	0.853113	0.012578	0.000345	0.018582
	0.100	0.723755	0.646652	0.020357	0.001038	0.032219
	0.001	0.951658	0.924342	0.009004	0.000182	0.013478
100	0.010	0.947827	0.916523	0.009439	0.000196	0.014002
	0.100	0.853365	0.768511	0.016448	0.000551	0.023474

Table 4.4: Radial Basis Kernel

C	epsilon	test-R^2	train-R^2	MAE	MSE	RMSE
	0.001	0.201433	0.183319	0.033126	0.003001	0.054780
0.01	0.010	0.206711	0.181211	0.033051	0.002981	0.054599
	0.100	0.064123	0.053826	0.035879	0.003517	0.059303
	0.001	0.726944	0.664761	0.019956	0.001026	0.032033
0.1	0.010	0.713270	0.651345	0.020393	0.001077	0.032825
	0.100	0.332924	0.313602	0.030287	0.002507	0.050067
	0.001	0.915954	0.872977	0.011625	0.000316	0.017772
1	0.010	0.905066	0.861811	0.012327	0.000357	0.018888
	0.100	0.733017	0.661501	0.019925	0.001003	0.031674
	0.001	0.948308	0.928279	0.008958	0.000194	0.013937
10	0.010	0.945963	0.921009	0.009269	0.000203	0.014250
	0.100	0.850161	0.782843	0.015978	0.000563	0.023729
	0.001	0.964385	0.948258	0.007670	0.000134	0.011569
100	0.010	0.958856	0.946615	0.007929	0.000155	0.012434
	0.100	0.879205	0.816507	0.014648	0.000454	0.021305

Chapter 5

CONCLUSION AND RECOMMENDATION

5.1 Conclusion

In this project we investigated the effect of kernel functions, the reguralization parameter, and epsilon on the accuracy of a linear-based model which is the support vector regression. The results show a close error(MAE, MSE, RMSE) of the linear, polynomial and radial basis function kernel. In all, the radial basis function appeared to be the kernel with the least error. We conclude that although kernel functions have effect on the accuracy of the model, kernel function should be chosen based on the behaviour of the dataset. The performance of the model is also affected by the choice of regularization parameter C and ε. The model produce high accuracy when C and ε are choosen as 100 and 0.01 respectively.

5.2 Recommendation

In future we recommended that, some other kernel functions and selection of feature indicators of financial time series data be evaluated on accuracy of the same data.

REFERENCES

Müller, A.C., & Guido, S (2016). *Introduction to machine Learning with python*(1st ed.). O'relly Media,Inc.

Kowalczyk, A.(2017). *Support Vector Mechine Succinctly.* Morrisville, NC: Syncfusion,Inc.

Khanna, R., & Awad, M. (2015). *Efficient learing machines: Theories, concepts and applications for engineers and system designers.* Apress. httpp://www/doi.org/10.1007/978-1-4302-5990-9.

Smola, A., Burges, Chris., Drucker. H., Golowich, S., Hemmen, L.V., Muller, R.K., Schölkopf, B., & Vapnik, V.,(1996). Regression estimation with support learning machines, version 1.01. http://www.first.gmd.de/smola.

Smola, A.J., & Schölkopf, B.(2003, September 30) A tutorial on support vector regression. *statistics and Computing*, 14(3),199-222. doi: 10.1023/b:stco.0000035301.49549.88

Adetunji, A., Akinwale, P., Taofiki, A., & Bidemi, A.A. (2011) Artificial Neural Network Model for Forecasting Foreign Exchange Rate *World of Computer Science and Information Technology Journal (WCSIT)*, 1(3), 110-118.

Goyal, S., Sowrirajan, H., & Veeramacheneni, T.(2019) *Machine learning for statistical arbitrage: Using news media to predict currency exchange rates.* CS 229 projects, Spring 2019 edition. http://cs229.standford.edu/proj2019spr/

Yekkehkhany. B., Safari1. A., Homayouni. S., & Hasanlou. M.(2014, November 15-17) A comparison study of different kernel functions for SVM-based classification of multi-temporal polarimetry SAR Data. *The 1st ISPRS International Conference on Geospatial Information Research.*

Chiroma. H., Abdulkareem. S., Abubakar. A.I., & Herawan. T.(2014) Kernel functions for the support vector machine: Comparing performances on crude oil price data .

Shrawan. K.T., & Shubhamoy. D.,(2013, March) Effect of various kernels and feature selection methods on SVM performance for detecting email spams. *International Journal of Computer Applications*, 66(21).

Appiah, S.T., & Adetunde, I.A.(2011, August 15). Forecasting exchange rate between the Ghana cedi and the US dollar using time series analysis. *Current Research Journal of Economic Theory*, 3(2), 76-83., 0975-8887.

Chandar, S.K., Sumathi, M., & Sivanandam, S.N. (2015, Feb-Mar). Forecasting of foreign currency exchange rate using neural network *International Journal of Engineering and Technology (IJET)*, 7(1).

Basak, D., Pal, S., & Patranabis, D.C.(2007, October). Support vector regession. *Neural Information Processing-Letters and Reviews*, 11(10).

Bofa, A., Avuglah, R.K., & Harris, E. (2019). Artificial neural networks forecast of the exchange rate: Study of the Ghanaian cedi to the American dollar. *International Journel of Economics & Business*, 3(2), 299-305.

Richard, M.,(2007). Forecasting Volatility Project report 2007, Submitted to Department of Mathematics, Uppsals University, Country, pp: 71.

Nanayakkara, K.A.D.S.A., Chandrasekara, N.V., & Jayasundara, D.D.M. (2014, March). Forecasting exchange rates using time series and neural network approaches. *European International Journal of Science and Technology*,3(2).

Chandrasekara, N.V. & Tilakaratne, C.D., (2009). Forecasting Exchange rates using Artificial Neural Networks. Sri Lankan Journal of Applied Statistics, 10, 187-201.

Nagpure, A.R.,(2019, April). Prediction of multi-currency exchange rates using deep learning. *International Journal of Innovative Technology and Exploring Engineering (IJITEE)*, 8(6).

Muhammad Yasir, M., Durrani, M.Y., Afzal, S., Maqsood, M., Aadil, F., Mehmood, I., & Rho, S., (2019). An intelligent event-sentiment-based daily foreign exchange rate forecasting system.

He, H., & Shen, X. (2007, August) Bootstrap methods for foreign currency exchange rates prediction. *Proceedings of 2007 IEEE INNS Joint Conference on Neural Network*. Orlando, FL, USA.

Nasution, B.P., & Agah, A. (2000) Currency exchange rate forecasting with neural networks. 10(3).

Rojas, C.G., & Herman, M.(n.d.). Foreign exchange forecasting via machine learning.

Abreu, G., Neves, R., & Horta, N.(2018, May 29). Currency exchange prediction using machine learning, genetic algorithms and technical analysis.

Lin, C., S., Khan, H.A., Wang, Y.C., & Chang, R.Y.(2006, April). A new approach to modeling early warning systems for currency crises : Can a machine-learning fuzzy expert system predict the currency crises effectively? *JE Discussion Papers*. http://www.e.u-tokyo.ac.jp/cirje/research/03research02dp.html

Kamruzzaman, J., Sarker, R.A., & Ahmad, I.(n.d). SVM based models for predicting foreign currency exchange rates.

Ahmad, A.R., Khalid, M. & Yusof, R.(n.d). Machine learning using support vector mechine.

Kamruzzaman, J., & Sarker, R.A (2004, January) Forecasting of currency exchange rates using ANN: A case study. DOI: 10.1109.

Sokhanvar, A.,(2013, July) *Application of time series models in forecasting exchange rate*[Unpublished master dissertation]. Eastern Mediterranean University

Alamili, M. (2011, January) *Exchange rate prediction using support vector machines: A comparison with artificial neural networks.*

Sekar, V.B., Yoloye, M., Vasudevan, N., & Dommeti, S.M. (2017) *predicting currency exchange rate*

Muller, R.K., Smola, A.J., Rátsch, G., Schölkopf, B., Kohlmorgen, J., & Vapnik, V.,(1997). Predicting times series with support vector machines(p.999-1004).

Machine Learning: What it is and why it matters. (n.d.). Retrieved from https://www.sas.com/en-us/insights/analytics/machine-learning.html.

Browlee, J. (2019, August 12).Supervised and Unsupervised Machine Learning algorithm. Retrieved from https://machinelearningmastery.com/supervised-and-unsupervised-machine-learning-algorithms/

Lagrange multiplier.(2020, March 30). Retrieved April 14, 2020 from http://en.m.wikipedia.org/wiki/Lagrange-multiplier.

Lauer, F. (2014).An interactive journey into Machine Learning: The optimal separating hyperplane and the margin. Retrieved from http://mlweb.loria.fr/book/en/optimalhyperplane.html

APPENDIX

Data Exploration

```
In [191]: # import packages

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

%matplotlib inline

import warnings

warnings.filterwarnings('ignore')
```

```
In [192]: # load file

USD_whole = pd.read_csv('Data/USD_GHS.csv',parse_dates=['Date'],index_col='Date
```

```
In [193]: # Data information

USD_whole.info()
```

```
<class 'pandas.core.frame.DataFrame'>

DatetimeIndex: 1643 entries, 2020-04-17 to 2014-01-01

Data columns (total 5 columns):

Price       1643 non-null float64

Open        1643 non-null float64

High        1643 non-null float64

Low         1643 non-null float64

Change %    1643 non-null float64

dtypes: float64(5)

memory usage: 77.0 KB
```

```
In [194]: USD_whole.shape
```

```
Out[194]: (1643, 5)

In [195]: USD_whole.head()

Out[195]:              Price   Open    High    Low  Change %
Date
2020-04-17  5.6988  5.760  5.8350  5.650     -1.06
2020-04-16  5.7600  5.840  5.9150  5.715     -0.26
2020-04-15  5.7750  5.915  5.9200  5.650      1.58
2020-04-14  5.6850  5.740  5.8675  5.650     -2.32
2020-04-13  5.8200  5.915  5.9150  5.820     -0.17

In [196]: # sorting values by date in ascending order
USD_whole = USD_whole.sort_values('Date')

In [197]: ## Since we are dealing with univariate data we select only the pri
USD_GHS = USD_whole[['Price']]
USD_GHS.head()

Out[197]:              Price
Date
2014-01-01  2.3650
2014-01-02  2.3350
2014-01-03  2.3350
2014-01-06  2.3650
2014-01-07  2.3515

In [198]: USD_GHS.shape

Out[198]: (1643, 1)

In [199]: # Descriptive Statistics
USD_GHS.describe()
```

```
Out[199]:              Price
count  1643.000000
mean      4.267410
std       0.803632
min       2.335000
25%       3.820000
50%       4.320000
75%       4.811250
max       5.855000
```

```
In [200]: # print out year with the minimum price
USD_GHS[(USD_GHS['Price']==min(USD_GHS['Price']))]
```

```
Out[200]:             Price
Date
2014-01-02  2.335
2014-01-03  2.335
```

```
In [201]: # print out year with the maximum price
USD_GHS[(USD_GHS['Price']==max(USD_GHS['Price']))]
```

```
Out[201]:             Price
Date
2020-04-07  5.855
```

```
In [202]: # ceck for missing values
USD_GHS.isnull().sum()
```

```
Out[202]: Price    0
dtype: int64
```

Data Visualization

```
In [203]: # print the graph of the exchange rate
import matplotlib.pyplot as plt
```

```
import seaborn as sns
USD_GHS.plot(figsize=(12,5))
plt.title("USD/GHS exchange rate")
plt.ylabel('Rate')
plt.show()

In [204]: USD_GHS.plot(kind='kde')

Out[204]: <matplotlib.axes._subplots.AxesSubplot at 0x293f003a588>
```

Test for Stationarity

```
In [207]: # test for stationarity
from statsmodels.tsa.stattools import adfuller
result = adfuller(USD_GHS['Price'])
print('ADF Statistic: %f' % result[0])
print('p-value: %f' % result[1])
print('Critical Values:')
for key, value in result[4].items():
print('\t%s: %.3f' % (key, value)) # data not stationary

ADF Statistic: -1.522769
p-value: 0.522219
Critical Values:
1%: -3.434
5%: -2.863
10%: -2.568
```

Data is not stationary since ADF value is greater than all critical values again p-value is greater than any of the alpha value hence we differences

```
In [208]: # differencing
USD_GHS['diff_1'] = USD_GHS.diff()
USD_GHS.head()

Out[208]:              Price  diff_1
Date
2014-01-01  2.3650     NaN
2014-01-02  2.3350 -0.0300
2014-01-03  2.3350  0.0000
2014-01-06  2.3650  0.0300
2014-01-07  2.3515 -0.0135

In [209]: # test again for stationarity on the diff_1 column
result = adfuller(USD_GHS['diff_1'][1:])
print('ADF Statistic: %f' % result[0])
print('p-value: %f' % result[1])
print('Critical Values:')
for key, value in result[4].items():
print('\t%s: %.3f' % (key, value))

ADF Statistic: -9.530064
p-value: 0.000000
Critical Values:
1%: -3.434
5%: -2.863
10%: -2.568
```

Data is stationary since ADF value is lesser than all values of the critical again p-value is lesser than any of the alpha value

```
In [212]: # we save the data USD.csv to the file
USD_GHS.to_csv('USD.csv', sep=',')
```

SVR Model

```
In [153]: # import packages
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')

In [154]: USD = pd.read_csv('USD.csv')

In [155]: USD.head()

Out[155]:          Date   Price  diff_1
0  2014-01-01  2.3650     NaN
1  2014-01-02  2.3350 -0.0300
2  2014-01-03  2.3350  0.0000
3  2014-01-06  2.3650  0.0300
4  2014-01-07  2.3515 -0.0135

In [156]: # remove null values
USD.dropna(inplace=True)
USD.head()
```

Data Preprocessing

```
In [90]: # Changing data to supervised learning
# adding moving average indicators of 5,10,20,60, 120 weeks and lag_1 as indi
# setting inputs values
USD_supervised = pd.DataFrame()
```

```
USD_supervised['MA5'] = USD['diff_1'].ewm(span=5).mean()
USD_supervised['MA10']= USD['diff_1'].ewm(span=10).mean()
USD_supervised['MA20']=USD['diff_1'].ewm(span=20).mean()
USD_supervised['MA60']=USD['diff_1'].ewm(span=60).mean()
USD_supervised['MA120']=USD['diff_1'].ewm(span=120).mean()
USD_supervised['t'] = USD['diff_1'].shift(1)
USD_supervised['t+1'] = USD['diff_1']
```

```
In [91]: USD_supervised.head()
```

```
Out[91]:          MA5       MA10       MA20       MA60      MA120        t      t+1
1 -0.030000 -0.030000 -0.030000 -0.030000 -0.030000      NaN -0.0300
2 -0.012000 -0.013500 -0.014250 -0.014750 -0.014875 -0.0300  0.0000
3  0.007895  0.003987  0.001998  0.000667  0.000333  0.0000  0.0300
4 -0.000992 -0.001774 -0.002476 -0.003054 -0.003212  0.0300 -0.0135
5  0.001692  0.000457 -0.000426 -0.001121 -0.001308 -0.0135  0.0060
```

```
In [94]: USD_supervised.shape
```

```
Out[94]: (1641, 7)
```

```
In [95]: #defining input(x) and output(Y) values
# isolating X and Y columns
X = USD_supervised.iloc[:,:-1]
y = USD_supervised.iloc[:,-1]

# calculate train data size
train_size = int(len(USD_supervised)*0.8)

# split
X_train, y_train = X[0:train_size], y[0:train_size]
X_test, y_test = X[train_size:len(X)], y[train_size:len(y)]
```

```
#X_train,y_train,X_test,y_test
```

```
In [96]: print('Length of train data :',train_size)
print('**************')
print('Length of test data :',len(USD_supervised)-train_size)
```

```
Length of train data : 1312
**************
Length of test data : 329
```

SVR

```
In [74]: # import packages
from sklearn.svm import SVR
from sklearn import metrics
from sklearn.metrics import accuracy_score
```

Metrics evaluation

```
In [75]: def mean_absolute(actural_y,pred_y):
return(metrics.mean_absolute_error(actural_y, pred_y))
```

```
In [76]: def mean_squared(actural_y,pred_y):
return(metrics.mean_squared_error(actural_y,pred_y))
```

```
In [77]: def root_mean(actural_y,pred_y):
return(np.sqrt(metrics.mean_squared_error(actural_y,pred_y)))
```

Invert difference

```
In [78]: def invert_difference(orign_data, diff_data, interval):
return[diff_data[i-interval]+orign_data[i-interval] for i in range(interval,len
```

true y values

```
In [98]: true_y = USD.iloc[train_size:,1].values
date = USD.iloc[train_size:,0].values
```

Linear Kernel

```
In [114]: svr_linear = pd.DataFrame({'C': [], 'epsilon': [], 'test-R2': [], 'tr
for C in [0.01,0.1,1,10,100]:
for epsilon in [0.001,0.01,0.1]:
linear_model = SVR(C=C,epsilon=epsilon,kernel='linear')

#train
linear_model.fit(X_train,y_train)

#predict
linear_pred = linear_model.predict(X_test)

# Evaluate
R2_score = linear_model.score(X_test,y_test)
train_score = linear_model.score(X_train,y_train)
MAE = mean_absolute(y_test,linear_pred)
MSE = mean_squared(y_test,linear_pred)
RMSE = root_mean(y_test,linear_pred)

# form dataframe
svr_linear = svr_linear.append({'C': C, 'epsilon': epsilon, 'test-R2': R2_score

# best predicted y values
if R2_score == max(svr_linear['test-R2']):
linear_pred = linear_pred
```

```
#Invert difference of best predicted y values
linear_pred_y = invert_difference(true_y, linear_pred, 1)

In [174]: print(svr_linear.head(8))
print('\n'+'***************'+'\n')
print(svr_linear.tail(7))
print('\n'+'***************'+'\n')
print('Best linear score: \n')
best_linear_model = svr_linear[(svr_linear['test-R2']==max(svr_linear['test-R2
print(best_linear_model)
```