QATAR UNIVERSITY

COLLEGE OF ENGINEERING

APPLYING VARIOUS MACHINE LEARNING METHODOLOGIES INTO THE FINANCIAL MARKET

BY

MAHMOUD G. MESLEH

A Thesis Submitted to

the College of Engineering

in Partial Fulfillment of the Requirements for the Degree of

Masters of Science in Electrical Engineering

January 2022

COMMITTEE PAGE

The members of the Committee approve the Thesis of
Mahmoud G. Mesleh defended on 28/11/2021.

Prof. Mustafa Serkan Kiranyaz
Thesis/Dissertation Supervisor

Dr. Muhammad Chowdhury
Committee Member

Approved:

Khalid Kamal Naji, Dean, College of Engineering

ABSTRACT

MESLEH, MAHMOUD, G, Masters: January 2022, Masters of Science in Electrical Engineering

Title: Applying Various Machine Learning Methodologies into the Financial Market

Supervisor of Thesis: Prof. Mustafa, Serkan, Kiranyaz.

The modernization of the financial market, with the introduction of the internet, made it easier for the average, everyday people, around the world to invest in the plentiful trading assets in the market. This created a revolution, propelling the foreign exchange market to be the most valuable and tradeable financial asset on the planet, with a daily turnover that surpasses $6 Trillion. As a result, predicting the future price can be very profitable, causing analysts and hedge funds to start a race toward searching for the best tools or algorithm that allows them to be ahead of the competition. With the introduction of faster and more powerful computers, the dream of automated, lightning-fast trading became a reality. Studies believe that more than 60% of the total traded volume in developed nations is performed by automated systems and algorithms.

This thesis will investigate the claims by different studies that machine learning algorithms can be used to accurately predict the future prices of the market. The thesis chose the EUR/USD exchange rate, to study, as it is the most volatile asset and it has the highest trading volume. Based on this, ten years of daily closing prices that included many trading assets, such as currencies, indices, and commodities were collected to study the effect of different trading assets on each other and understand the correlation effect.

The investigation starts with the use of linear regression techniques, including mean least-squares estimations and multiple linear regression, which failed to provide sustainable results or achieve an accuracy above 60%. In addition to that, a support vector machines model was built using a linear and a radial basis function kernel, where the linear kernel model recorded an accuracy of 60% when predicting the future price trend of the EUR/USD.

Finally, the thesis dives into the use of artificial neural networks, in the form of multi-layer perceptrons (MLPs), and long-short term memory (LSTMs). All forms of artificial neural networks have failed to predict the future price trend when using one day of previous closing prices as input. This has changed when an MLP regressor was trained to use the previous closing price of 30 days to predict one day into the future. This allowed the network to achieve accuracies that exceeded 80% when predicting the future price trend.

DEDICATION

Женщине, которая изменила мою жизнь.

Я люблю тебя. И посвящаю это тебе.

Моей Оле

ACKNOWLEDGMENTS

I would like to express my deepest gratitude to Prof. Serkan Kiranyaz, who is my supervisor for this thesis, for his immense support throughout this journey. His support was overwhelming, and his guidance was the key behind all of the progress made. I would like to extend my gratitude to my family and friends for their constant support.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1: INTRODUCTION

The financial market is a crucial pillar in today's economy, with the foreign exchange market (FOREX) having a daily trading volume of $6.6 Trillion. Investors in the financial market range from multi-billion dollar hedge funds, banks, and governments to everyday folks trying to invest some money into the stocks market, or simply exchanging currency at the airport while on vacation [1].

Many experts believe that 60-70% of the trading volume in the financial market of developed countries is automated algorithmic trading. Algorithmic trading was made easier thanks to the computing power and communication technology developed over time, in addition to having access to more data to train and evaluate trading models [2]. A list of commonly used tradeable assets can be found in Table 1.

Table 1. Different types of trading assets.

Asset	Definition	Examples
Equities/Stocks	Ownership of a corporation is offered for public trading	Apple (AAPL) General Motors (GM)
Commodities	Trading assets that include agricultural, precious metals, etc...	West Texas Oil (WTI) Gold Spots (XAU/USD)
Foreign Exchange (FOREX)	Trading the difference of a base currency against a basket of other currencies	Euro against US Dollar (EUR/USD) US Dollar against Japanese Yen (USD/JPY)

FOREX is the most commonly traded asset in the world, followed by stocks and equities. As a result of such a huge volume, the problem of financial market analysis becomes filtering all the noise, in addition to the stochastic nature of the market [3]. As a result, experts started coming up with ways to try to understand the market and predict future prices.

1.1 Financial Analysis

1.1.1 Fundamental Analysis

In fundamental analysis, predictions are based on the study of the supply and demand of the traded asset. In addition to that, the news, economical reports, interest rates, correlation with other assets, and many other fundamentals are considered before making a prediction. While compiling the dataset of the study, a correlation between multiple assets was noted, as can be seen as follows, showing the price of Platinum against the Russian Ruble vs. US Dollar (RUB/USD) exchange price.

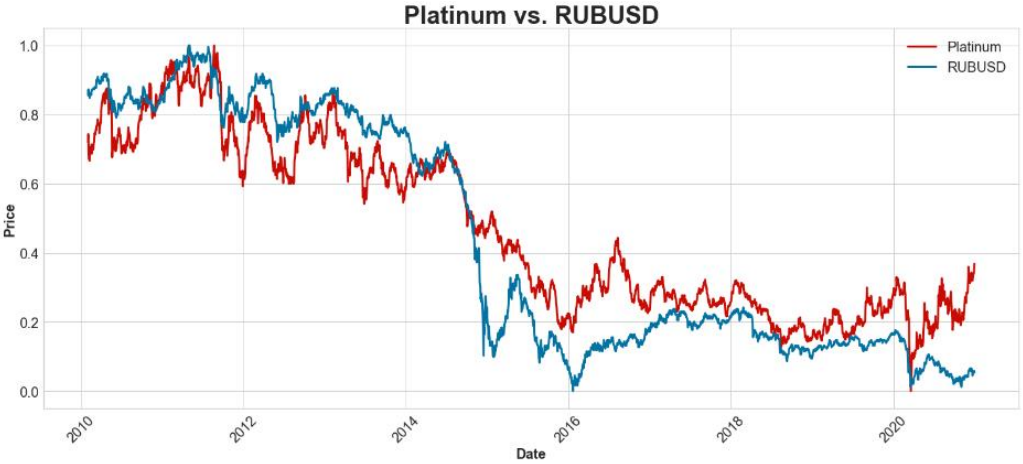


Figure 1. Price change of Platinum vs. RUB/USD.

After further investigation, it was made clear that Russia is the second-largest Platinum producer in the world [4], as a result, the Rubble exchange price greatly affects the price of Platinum. In this case, a Platinum investor would look closely into the Russian economy, and follow the economical reports that may affect the exchange rate of the Ruble.

Another great example of fundamental analysis can be seen in Figure 2. The United States Dollar Index, which measures the value of the United States Dollar against other widely used currencies, is plotted against the United States Conference

Board Consumer Confidence, which is a monthly economic report that measures the sentiments and confidence of consumers in the US market [5]. The proportional relationship between the two variables can be seen as follows.

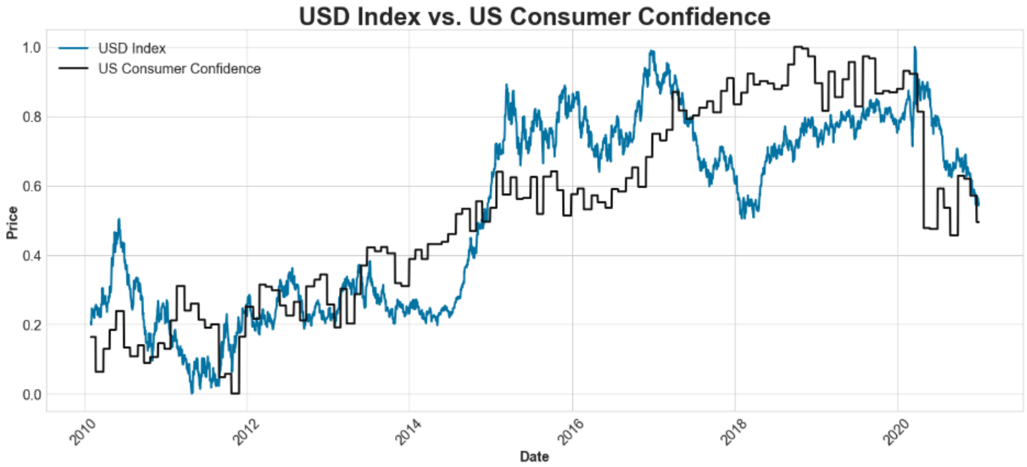


Figure 2. Changes in the USD Index vs. The United States Conference Board Consumer Confidence.

1.1.2 Technical Analysis

When using technical analysis, the analyst depends on indicators to predict the trend of the asset in the future [6]. Indicators are based on mathematical models, ranging from simple moving averages, weighted and exponential moving averages to more complex mathematical model-based indicators such as the Ichimoku Cloud or Fibonacci Retracements [7].

In a simple case study, the EUR/USD past data can be used to produce moving averages (MA) over the 50 days (fast) and 200 days (slow) periods. A very widely used trading strategy is simply going long when the slow MA crosses the fast (MA) upwards and shorting when it is the other way around. Figure 3 shows an example of such a trading strategy.

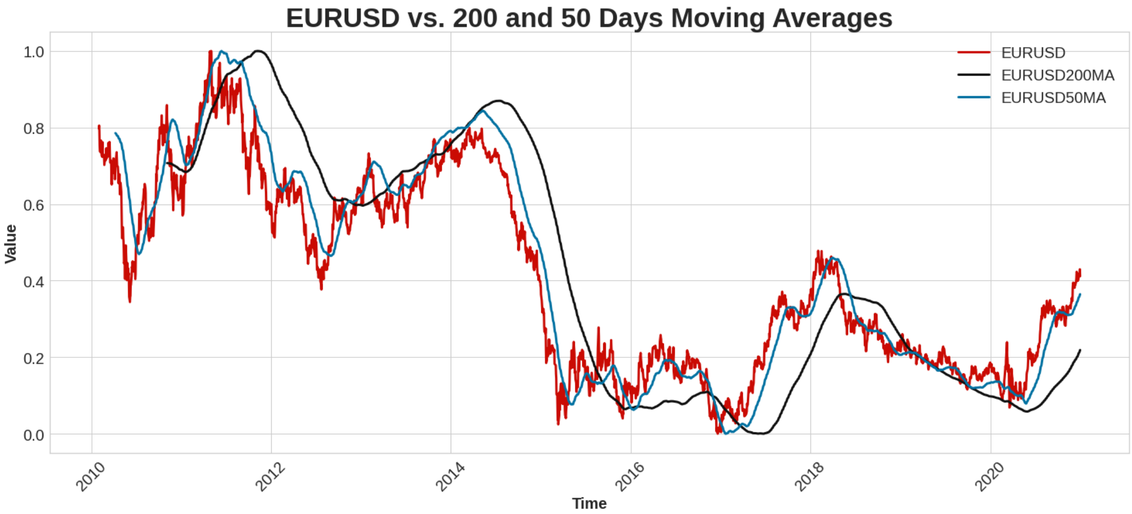


Figure 3. EUR/USD vs. 50 and 200 days moving averages.

More complex trading strategies can yield accurate results by using different indicators such as the Moving Average Convergence/Divergence (MACD) and the Relative Strength Index (RSI) [8].

1.2 Related Works

One of the earliest research into forecasting exchange rates is found to compare time series and structural models of FOREX based on their out-of-sample forecasting accuracy. The research has discovered that most linear models failed the random walk test, and were not effective in predicting future prices [9]. Random walk test is a random process, stochastic in nature, that in essence describes a path that comprises of random steps in a mathematical space, a model would take, based upon the inputs it receives. Since then, researchers have been trying to use linear and non-linear models to beat the naïve random walk theory, some have proven it possible while others refuted and proposed that it is not possible [1].

More complex linear and time-series models can be used to predict future prices, in addition to polynomial-based non-linear models, which have shown some success [10]. Autoregressive integrated moving average (ARIMA) models have also seen some success recently, with the help of excess data available to regress upon, but they are commonly used in less fluctuating trading assets, as a result, they tend to have a low return on investment percentage compared to other models.

With the advent of Artificial Neural Networks (ANNs), many studies have been conducted to forecast the future prices of stocks and other trading assets [11]. Most recent studies commonly use the Recurrent Neural Networks (RNNs) variant, and the long short-term memory (LSTM) model [12], they have been shown to achieve higher accuracies even in a low timeframe (hours/minutes) [2]. Nonetheless, a simple ANN model was proven effective in many short-term trading strategies, and in many countries, applied to the United Kingdom banking sector to predict share prices yielded results that are far better than traditional linear models [13]. Similar experiments were carried on using ANNs, on the US stocks market.

The US market is known to be volatile, and highly responsive to news, economical reports, and international events. This can prove to be a challenge for a model to be robust enough to predict future prices, but advanced models have proven to be able to overcome, in a way, the market noise [14]. The financial news and reports cause the market to move, as a result, analyzing the impact of news articles on the stocks market can provide an advantage toward predicting future price changes. Studies have found success in analyzing the sentiment of news articles; thus understanding their impact on the market ahead of sudden price changes [15].

The predictive powers of neural networks are not only limited to predicting the price of stocks. Some success was found with the prediction process of commodities [16], such as gold and oil [17]. The theory that artificial neural networks can detect the correlation between the different assets to predict future prices has seen some results to support such a claim, as with the help of tools, such as data mining, which is a technique used to find correlation and patterns in large sets of data can be effective to train a network to predict future prices [18].

Hybrid models that use different types of neural network architecture have seen some success in recent years [19]. Commonly used hybrid models contain an LSTM network to regress the data and produce an output using information produced from a convolutional neural network (CNN) that processes the news or social media for potential variables that might affect the price. However, this can still be achieved using support vector machines (SVM) models, or other simpler alternative models, to process news or social media posts for potential updates that might cause the price to change [20]. Recent studies have also seen success in using hybrid systems to predict highly volatile markets, like the cryptocurrency market. Bitcoin is considered one of the most volatile trading assets in recent times, with a daily price rate of change that can exceed 25%. However, a hybrid system of a linear regression model and a decision tree classifier [21] was able to predict prices with high accuracy [22].

A commonly used technique to boost the confidence of a system is to use model enablement. A set of different models/architectures will be constructed to predict prices, the system will choose the output, in this case, the market direction prediction, depending on the majority voting. Using such a technique has produced an accuracy upwards of 85% for commodities such as gold [23].

1.3 Motivation and Objectives

The main goal of this thesis is to investigate the possibility of building a model that predicts future prices. The focus will be aimed toward building a high-quality dataset that includes various trading assets and economical reports, having such a data set will help in understanding the correlation between the different assets and how the prices are impacted by economical reports. Toward the end of the study, multiple models will be created, both linear and non-linear, including different ANN models.

All the different models will be compared in terms of efficiency and accuracy to conclude the viability of using such models to predict future prices. The biggest obstacle in any financial forecasting research is the effect caused by economical reports on the prices, including the news, political and environmental events. This is seen in the unprecedented fluctuation in the global financial market as a result of the COVID-19 pandemic.

However, the study would use such unforeseen events to measure the robustness of the models, as the dataset used in this research spans over the past decade, including the year 2020. The research would establish a baseline by using basic linear models, these models would be compared to other more sophisticated non-linear models to verify the effectiveness and robustness of such models. The study will dive into the selection process behind choosing hyperparameters of the artificial neural networks, and the effects of such parameters on the overall performance of the model.

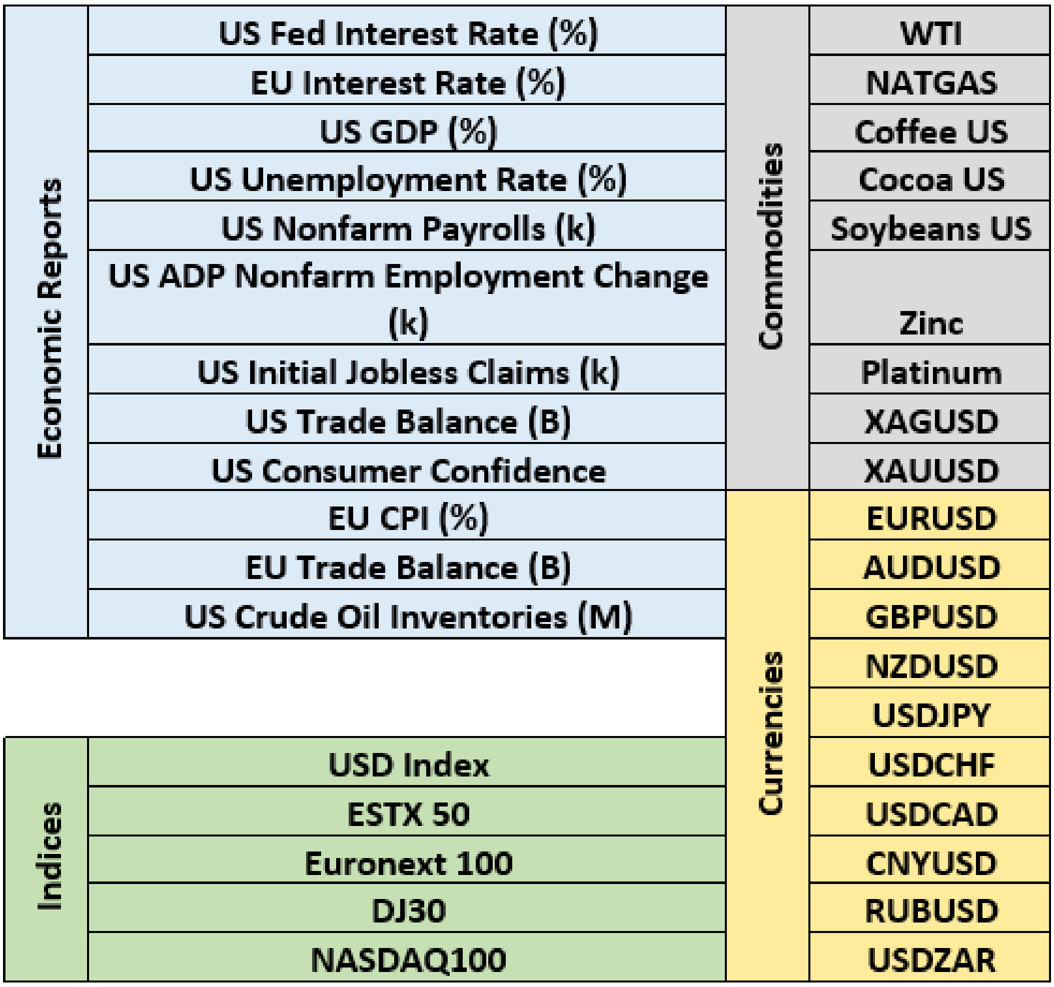
1.4 Thesis Outline

The rest of the thesis is organized as follows: Chapter 2 will discuss data exploration, in which the dataset that will be used throughout the study will be created, divided into categories, and preprocessed. In Chapter 3, a comprehensive literature review is conducted on the different concepts of models that will be created later in the research. Chapter 4 will show the performance of different machine learning models, and a comparison will be drawn between the different methodologies. Finally, Chapter 5 provides a conclusion that discusses the results obtained in prior chapters and provides some insight toward the future direction of this study.

CHAPTER 2: DATA EXPLORATION

2.1 Data Collection

Historical data that is plentiful and high in quality is crucial to understand the relationship between the different assets and build comprehensive and accurate models. As a result, multiple data sources and stock exchanges platform were used to build the dataset used in the research. The dataset used in this thesis starts from 1st of February 2010 up to 31st of December 2020, equating to 10 years' worth of daily closing price data. The dataset is broken into several categories, and an overview breakdown of the dataset can be seen in Figure 4.



Economic Reports	US Fed Interest Rate (%)	Commodities	WTI
	EU Interest Rate (%)		NATGAS
	US GDP (%)		Coffee US
	US Unemployment Rate (%)		Cocoa US
	US Nonfarm Payrolls (k)		Soybeans US
	US ADP Nonfarm Employment Change (k)		Zinc
	US Initial Jobless Claims (k)		Platinum
	US Trade Balance (B)		XAGUSD
	US Consumer Confidence		XAUUSD
	EU CPI (%)	Currencies	EURUSD
	EU Trade Balance (B)		AUDUSD
	US Crude Oil Inventories (M)		GBPUSD
			NZDUSD
			USDJPY
Indices	USD Index		USDCHF
	ESTX 50		USDCAD
	Euronext 100		CNYUSD
	DJ30		RUBUSD
	NASDAQ100		USDZAR

Figure 4. Break down of the study dataset.

2.2 Data Preprocessing

Using different trading assets to perform modeling and analysis imposes a huge problem when using raw data. Trading assets can vary in value, as a result, simple tasks such as visualizing the data becomes impossible to do. To solve this, linear data scaling is used in the form of data Normalization and Standardization.

2.2.1 Data Standardization

Standardization of the data is performed by applying Equation (1), where μ represents the mean, and σ represents the standard deviation of the variable X.

$$X_{\text{Standardized}} = \frac{X_{value} - \mu}{\sigma} \tag{1}$$

As a result, the standardized data will be centered around a mean value of zero. This allows the data to be easily displayed and plotted against other features that will have different numerical ranges.

2.2.2 Data Normalization

Normalized data using Equation (2) will be scaled to a range from 0 to 1 [24]. X_{max} and X_{min} represent the biggest and smallest values in the dataset X respectively.

$$X_{\text{Normalized}} = \frac{X_{value} - X_{min}}{X_{max} - X_{min}} \tag{2}$$

Normalization is commonly applied to data when the distribution is not a Gaussian distribution, this can be efficient when the algorithm does not assume the distribution of the dataset, such as in the case of artificial neural networks (ANNs).

2.2.3 Data Splitting and Evaluation Metrics

To unify the testing criteria for all the different methods used in the research, the data is split into a training set (80%) and a testing set (20%).

Coefficient of Determination (R^2): The results throughout this study will be measured using the R^2 score. The score is used to measure the amount of variance in the dependent variable that is produced as a prediction from a mathematical model using statistically independent variables. The R^2 test helps to show how much of the total variance of the test dataset is explained by the model. Equation (3) is used to produce the R^2 score of a model.

$$R^2 = 1 - \frac{\sum_i \left(y^i{}_{Actual} - y^i{}_{Model}\right)^2}{\sum_i \left(y^i{}_{Actual} - \bar{y}_{\,Actual}\right)^2} \tag{3}$$

where $y^i{}_{Actual}$ is the example from the dataset which will be used as the target value of the model, and $y^i{}_{Model}$ represent the corresponding output using the model, and $\bar{y}_{\,Actual}$ is the mean value of the target dataset. Normally, the value of R^2 ranges from 0 to 1, with a value of 1 indicating that changes in one variable can be perfectly explained by a discrepancy in a second variable.

Confusion Matrix (CM): used to describe the performance of a classification model [25]. The matrix is built as seen in Table 2.

Table 2. Confusion Matrix structure.

	Predicted False	**Predicted True**
Actual False	True Negative (TN)	False Positive (FP)
Actual True	False Negative (FN)	True Positive (TP)

True positives and negatives describe the case where the model predicted a true or false value correctly. Where false positives and negatives correspond to the case where the model incorrectly identifies a false value as true, or, a true value as false.

Accuracy: is used to produce an error rate percentage, indicating the rate at which the model can predict outputs correctly. It can be calculated as follows:

$$Accuracy\ (\%) = \frac{TP + TN}{TP + TN + FN + FP} * 100 \tag{4}$$

Precision: is used to measure the ratio of true positive outputs of the model's overall predictions. It can be calculated:

$$Precision = \frac{TP}{TP + FP} \tag{5}$$

Recall: is used to measure the ratio of true positives produced by the model, compared to the total number of positive class items in the dataset. It can be calculated as follows:

$$Recall = \frac{TP}{TP + FN} \tag{6}$$

F1-Score: is another measure of a classifier model's accuracy, it combines both precision and recall to evaluate the model performance. It can be calculated as:

$$F1 - Score = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{7}$$

Cross-Entropy (CE): is used as a loss function to evaluate a classification model. (N) is the total number of training examples.

$$CE = \frac{1}{N}\sum_{i}^{N} {y^i}_{Actual}\log\,({y^i}_{Model} + (1 - {y^i}_{Actual}))\log\,(1 - {y^i}_{Model}) \tag{8}$$

Mean Absolute Percentage Error (MAPE): is a statistical accuracy measurement for data forecasting. *MAPE* is calculated using Equation (9), where N is the size of the dataset.

$$MAPE = \frac{100}{N}\sum_{i}^{N}\frac{\left|{y^i}_{Actual} - {y^i}_{Model}\right|}{{y^i}_{Actual}} \tag{9}$$

Mean Root Square Error (RMSE): This is a commonly used error validation metric, and it is used to evaluate the performance of a regression model compared to the actual output [13]. Equation (10) is used to calculate the RMSE.

$$RMSE = \sqrt{\frac{\sum_{i}^{N}\left({y^i}_{Model} - {y^i}_{Actual}\right)^2}{N}} \tag{10}$$

Mean Absolute Error (MAE): is a measurement of error that is used to evaluate the performance of the model, it is calculated as follows:

$$MAE = \frac{1}{N}\sum_{i=1}^{N}\left|y^{i}{}_{Model} - y^{i}{}_{Actual}\right| \tag{11}$$

Mean Squared Error (MSE): is another measurement of error, it is calculated as follows:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}\left(y^{i}{}_{Model} - y^{i}{}_{Actual}\right)^{2} \tag{12}$$

2.3 Data Correlation

Correlation is a measurement of statistical association, it can help in identifying if a linear relationship exists between two variables. It is calculated as follows:

$$\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i \tag{13}$$

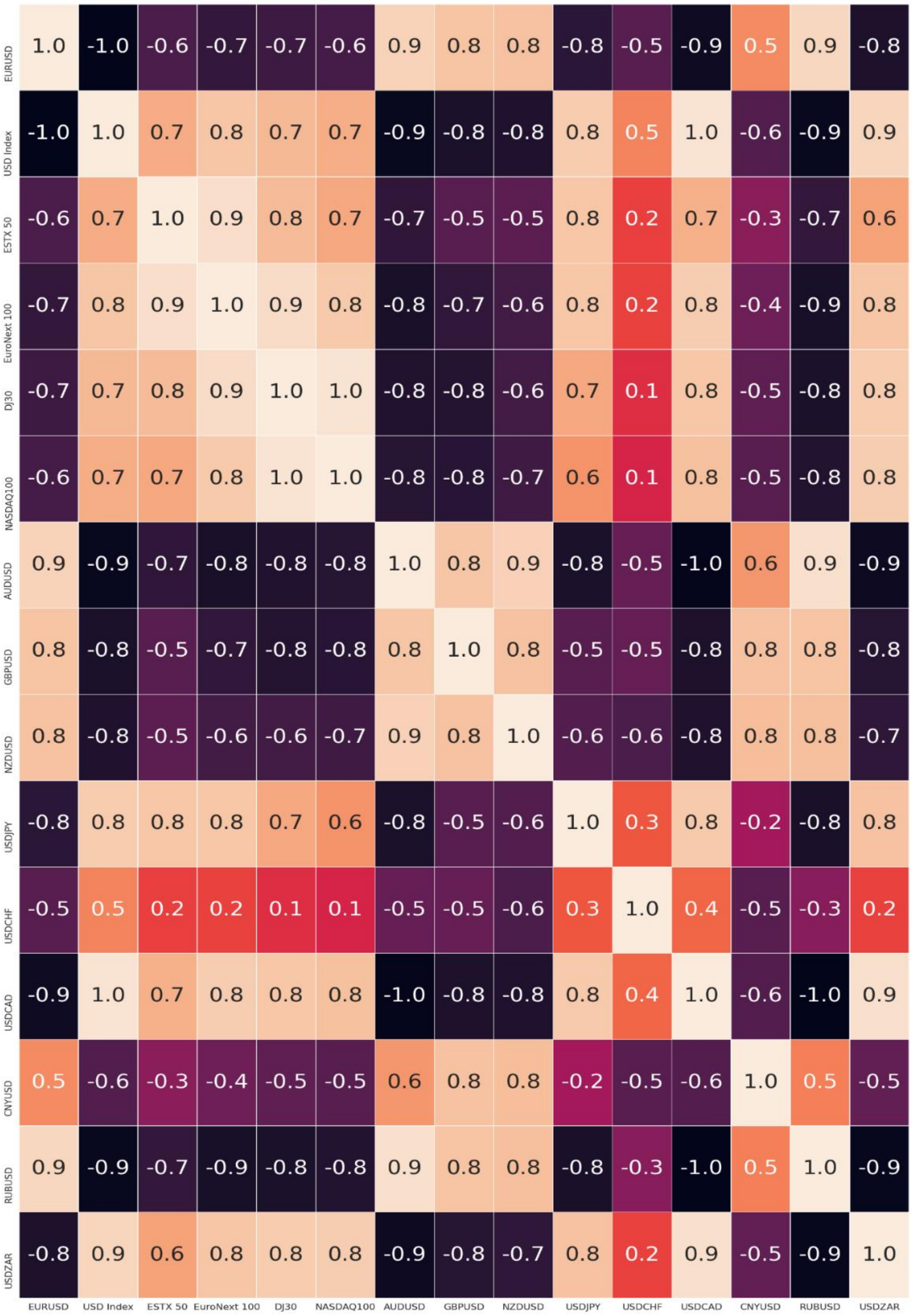
$$\bar{y} = \frac{1}{N}\sum_{i=1}^{N} y_i \tag{14}$$

$$s_x = \frac{1}{N}\sum_{i=1}^{N}(x - \bar{x})^2 \tag{15}$$

$$s_y = \frac{1}{N}\sum_{i=1}^{N}(y - \bar{y})^2 \tag{16}$$

$$C_{x,y} = \sum_{i=1}^{N}\frac{(x - \bar{x})^2(y - \bar{y})^2}{(s_x s_y)} \tag{17}$$

The correlation coefficient ($C_{x,y}$) requires calculating the mean $(\bar{x}, \bar{y})$ and the standard deviation (s_x, s_y) of both variables (x, y). This was applied to calculate the correlation across the variables in the dataset, as illustrated in Figure 5.



	EURUSD	USD Index	ESTX 50	EuroNext 100	DJ30	NASDAQ100	AUDUSD	GBPUSD	NZDUSD	USDJPY	USDCHF	USDCAD	CNYUSD	RUBUSD	USDZAR
EURUSD	1.0	-1.0	-0.6	-0.7	-0.7	-0.6	0.9	0.8	0.8	-0.8	-0.5	-0.9	0.5	0.9	-0.8
USD Index	-1.0	1.0	0.7	0.8	0.7	0.7	-0.9	-0.8	-0.8	0.8	0.5	1.0	-0.6	-0.9	0.9
ESTX 50	-0.6	0.7	1.0	0.9	0.8	0.7	-0.7	-0.5	-0.5	0.8	0.2	0.7	-0.3	-0.7	0.6
EuroNext 100	-0.7	0.8	0.9	1.0	0.9	0.8	-0.8	-0.7	-0.6	0.8	0.2	0.8	-0.4	-0.9	0.8
DJ30	-0.7	0.7	0.8	0.9	1.0	1.0	-0.8	-0.8	-0.6	0.7	0.1	0.8	-0.5	-0.8	0.8
NASDAQ100	-0.6	0.7	0.7	0.8	1.0	1.0	-0.8	-0.8	-0.7	0.6	0.1	0.8	-0.5	-0.8	0.8
AUDUSD	0.9	-0.9	-0.7	-0.8	-0.8	-0.8	1.0	0.8	0.9	-0.8	-0.5	-1.0	0.6	0.9	-0.9
GBPUSD	0.8	-0.8	-0.5	-0.7	-0.8	-0.8	0.8	1.0	0.8	-0.5	-0.5	-0.8	0.8	0.8	-0.8
NZDUSD	0.8	-0.8	-0.5	-0.6	-0.6	-0.7	0.9	0.8	1.0	-0.6	-0.6	-0.8	0.8	0.8	-0.7
USDJPY	-0.8	0.8	0.8	0.8	0.7	0.6	-0.8	-0.5	-0.6	1.0	0.3	0.8	-0.2	-0.8	0.8
USDCHF	-0.5	0.5	0.2	0.2	0.1	0.1	-0.5	-0.5	-0.6	0.3	1.0	0.4	-0.5	-0.3	0.2
USDCAD	-0.9	1.0	0.7	0.8	0.8	0.8	-1.0	-0.8	-0.8	0.8	0.4	1.0	-0.6	-1.0	0.9
CNYUSD	0.5	-0.6	-0.3	-0.4	-0.5	-0.5	0.6	0.8	0.8	-0.2	-0.5	-0.6	1.0	0.5	-0.5
RUBUSD	0.9	-0.9	-0.7	-0.9	-0.8	-0.8	0.9	0.8	0.8	-0.8	-0.3	-1.0	0.5	1.0	-0.9
USDZAR	-0.8	0.9	0.6	0.8	0.8	0.8	-0.9	-0.8	-0.7	0.8	0.2	0.9	-0.5	-0.9	1.0

Figure 5. Dataset correlation heatmap.

2.4 Data Visualization

To better understand the dataset, and the effect of the different features on a specific trading asset the following methodology has been followed. The initial step was to pick a trading asset from the dataset to predict future prices or trends. In this study, (EUR/USD) was chosen as it is the most volatile trading asset and it produces the highest trading volume.

The price of (EUR/USD) was converted into a binary output. This was done by comparing the closing price of the current day to the future closing price of the next day in the dataset, and converting the value into 1 if the price is going to increase for the next day, or 0 for the opposite. An example can be seen in Table 3.

Table 3. Price to binary conversion example.

EUR/USD	EUR/USD Next Day	EUR/USD Next Day Binary
1.3929	1.3963	1
1.3963	1.3899	0
1.3899	1.3736	0
1.3736	1.3664	0
1.3664	1.3653	0

Furthermore, the distribution of the binary conversion was displayed between price increase (value of 1) and price decrease (value of 0). The result can be seen in Figure 6, in which a count plot shows a near 50-50 even distribution.

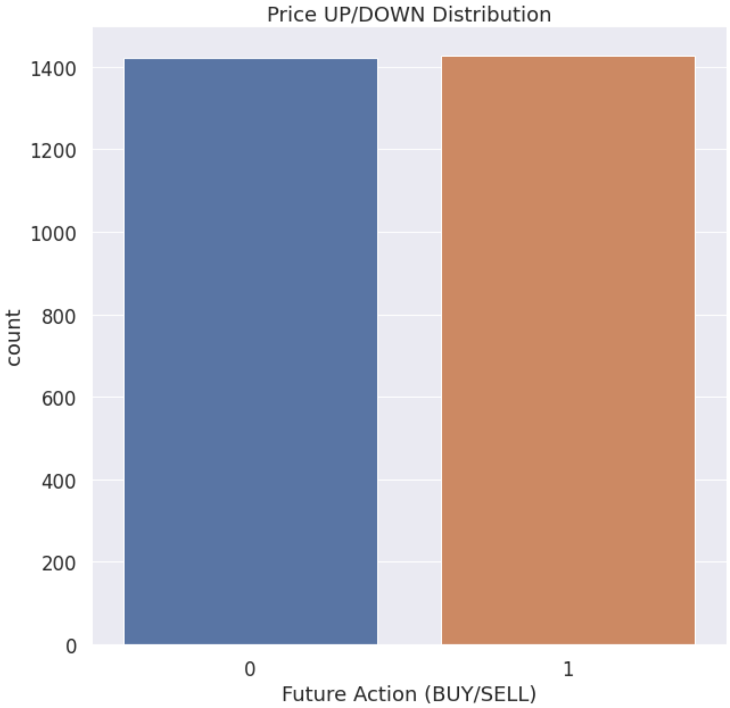


Figure 6. EUR/USD distribution between buy/sell.

The correlation between instruments can be visually inspected as shown in Figure 7. In this example, the EUR/USD is plotted against the trading assets and financial reports that produce the highest correlation value. These assets will be used as inputs to the different models that will be developed in this study.

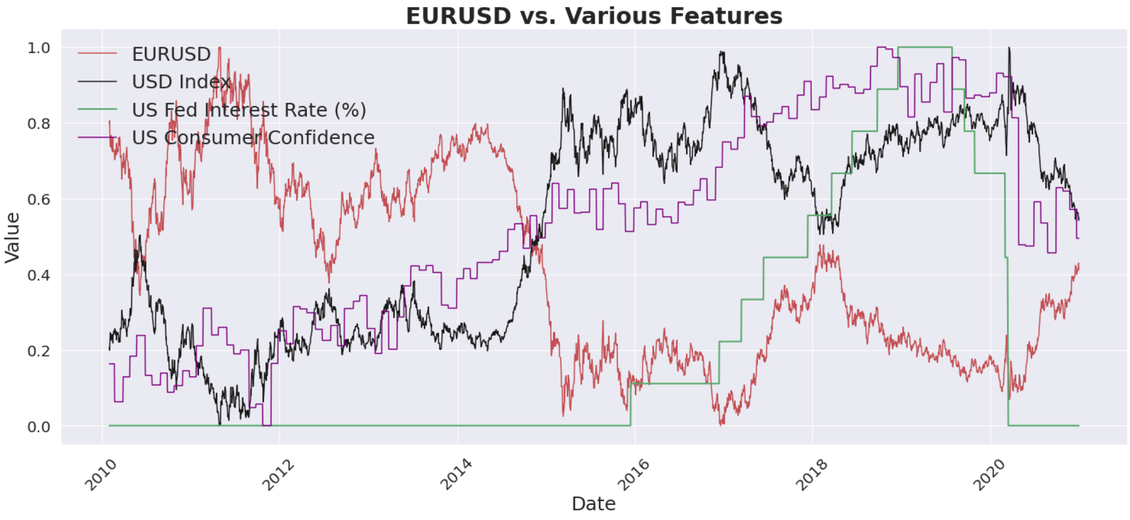


Figure 7. The EUR/USD plotted against various assets/financial reports.

CHAPTER 3: METHODOLOGIES

3.1 Regression Analysis

Regression analysis is used to describe the relationship between a set of variables in the form of a mathematical model. Regression analysis is used in numerous applications and it is considered the most commonly used statistical technique [26].

In a regression problem, the main goal is to use the available data to produce a model that minimizes the error of the model's predicted output with the truth output. Figure 8 illustrates the basic approach for building a regression model.

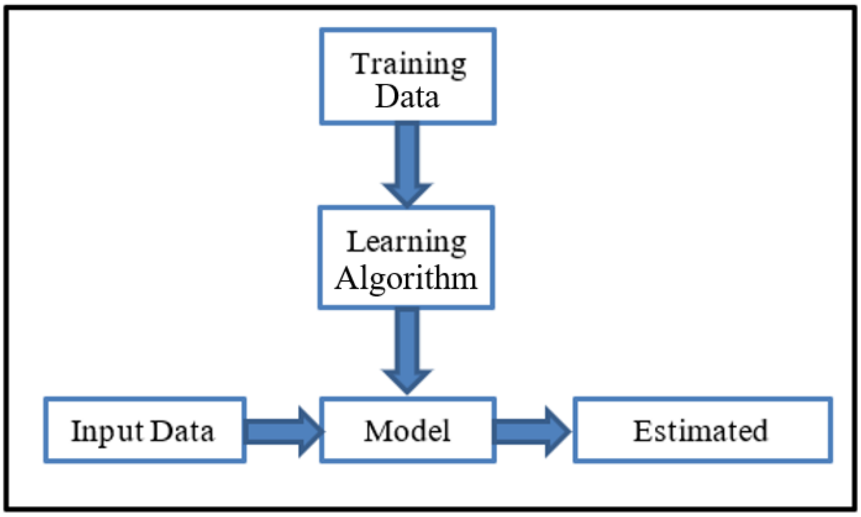


Figure 8. The process behind building a model.

The regression model is built using the information learned from the input variables, known as features, to product predictions for the output variable, known as the target.

3.1.1 Linear Regression

The linear regression model is formulated as in Equation (18), where θ_0 represents the bias (or intercept term) in the model, and θ_1 is the weight. This model is also referred to as the simple linear regression model [27].

$$y(x_1) = \theta_0 + \theta_1 x_1 \tag{18}$$

where adding more variables, as terms of (x) increase the complexity, in which the model will be regressed on a higher dimension. This model is called Multiple Linear Regression [28]. An example of such a model with two input variables can be seen in Equation (19).

$$y(x_1, x_2) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 \tag{19}$$

The θ_n term determines the weight of the term x_n toward the output of the model. Figure 9 illustrates an example of applying a linear model to fit a dataset.

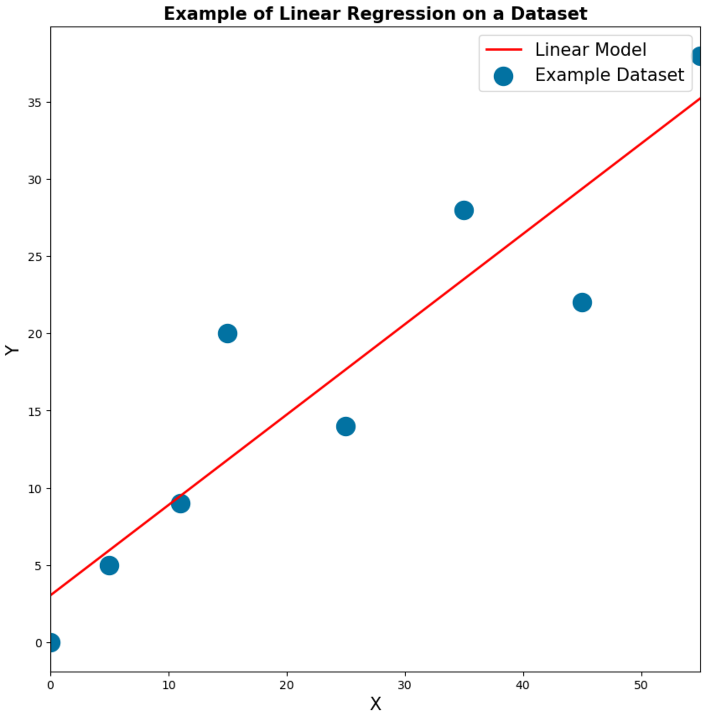


Figure 9. Example of a linear regression model.

3.1.2 Least Mean Squares Estimation

To estimate the unknown parameters seen in Equation (18), an approach known as the least mean squares (LMS) is used. This is done by defining a cost function $J(\theta)$, which is constructed in Equation (20). The cost function is used to measure the closeness of the predicted value of the model, y_{model}, compared to the actual output in the training data set, y_{actual}. The main goal is to minimize the error between the model and the actual training data. This is done by minimizing the value of cost function $J(\theta)$, where N is the number of training data.

$$J(\theta) = \frac{1}{2N}\sum_{i=1}^{N}(y_{model}(x^{(i)}) - y_{actual}^{(i)})^2 \tag{20}$$

3.1.3 Gradient Descent

Gradient descent is a powerful and commonly used optimization algorithm. It is used to minimize the cost function iteratively. This is performed by taking the derivative of the cost function $J(\theta)$ and minimizing the error of the partial derivative term of the cost function $\frac{\partial}{\partial\theta_j}J(\theta)$ [27]. This is expressed in Equations (21) and (22). The term $y_{model}\left(x^{(i)}\right)$ represents the model output using input from the training dataset.

$$j = 0, \frac{\partial}{\partial\theta_0}J(\theta_0, \theta_1) = \frac{1}{N}\sum_{i=1}^{N}\left(y_{model}\left(x^{(i)}\right) - y_{actual}^{(i)}\right) \tag{21}$$

$$j = 1, \frac{\partial}{\partial\theta_1}J(\theta_0, \theta_1) = \frac{1}{N}\sum_{i=1}^{N}\left(y_{model}\left(x^{(i)}\right) - y_{actual}^{(i)}\right)\left(x^{(i)}\right) \tag{22}$$

where $x^{(i)}$ and $y_{actual}^{(i)}$ are the input and target output from the training dataset, respectively. Gradient descent uses a learning rate parameter α. This parameter dictates the sensitivity of the algorithm, as it affects the magnitude of the steps that the GD applies each iteration. The process is performed over many iterations for θ_0, θ_1 as shown in Equations (23) and (24) until both converge. Convergence occurs when the gradient calculated in (21) and (22) vanishes.

$$\theta_0 = \theta_0 - \alpha\left[\frac{1}{m}\sum_{i=1}^{N}\left(y_{model}\left(x^{(i)}\right) - y_{actual}^{(i)}\right)\right] \tag{23}$$

$$\theta_1 = \theta_1 - \alpha\left[\frac{1}{m}\sum_{i=1}^{N}\left(y_{model}\left(x^{(i)}\right) - y_{actual}^{(i)}\right)\left(x^{(i)}\right)\right] \tag{24}$$

A flow chart, simplifying the process of gradient descent is illustrated in Figure 10.

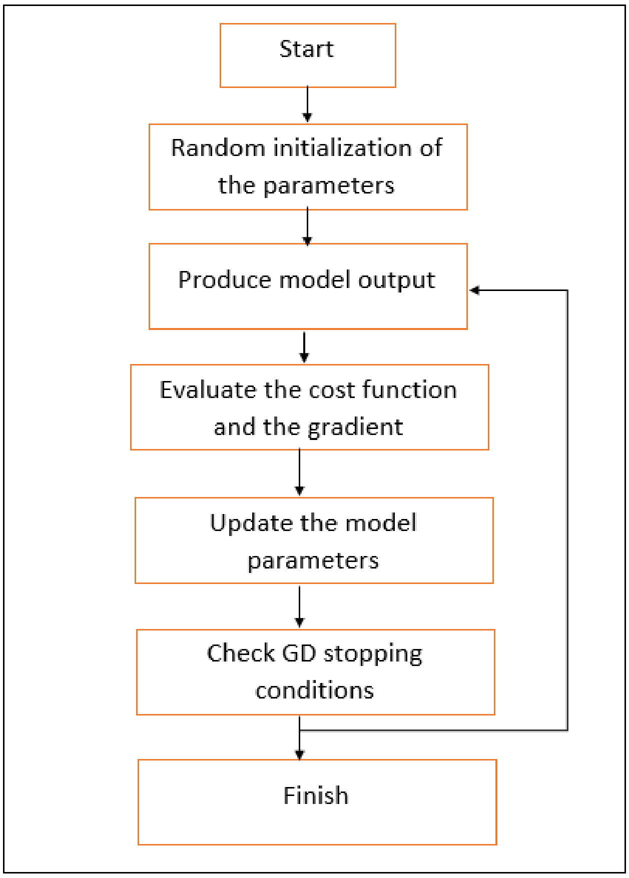

Start
Random initialization of the parameters
Produce model output
Evaluate the cost function and the gradient
Update the model parameters
Check GD stopping conditions
Finish

Figure 10. A flowchart showing the process of applying gradient descent.

3.2 Logistic Regression

Logistic regression is used to solve classification problems. In the most basic case, a dataset would have two potential outputs, for example, a true or a false output, this results in a classification problem with two classes. The model is used to predict such output based on a probabilistic function. A commonly used function is the sigmoid function, in which the output ranges between [0,1], expressed in Equation (25).

$$y = \frac{1}{1 + e^{-x}} \tag{25}$$

The Sigmoid function in a range of $[-10,10]$ is plotted in Figure 11.

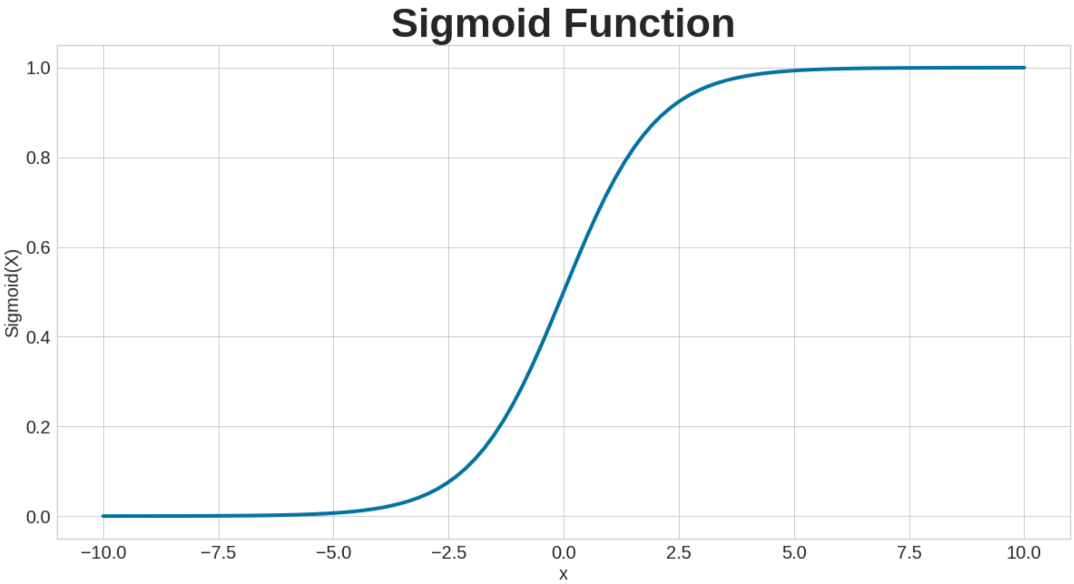


Figure 11. Sigmoid function output.

The output of the sigmoid function is a probabilistic prediction, a threshold can be used to classify the output based on the given data [25]. The model $h_\theta(x)$ uses the sigmoid function to produce a probabilistic output value bounded between $0 \leq h_\theta(x) \leq 1$, where $h_\theta(x)$ is modeled as follows:

$$h_\theta(x) = g(\boldsymbol{\theta}^{\boldsymbol{T}}\boldsymbol{x}) = \frac{1}{1 + e^{-\boldsymbol{\theta}^{\boldsymbol{T}}\boldsymbol{x}}} \tag{26}$$

The dot product of the model's parameter vector $\boldsymbol{\theta}$ and the input vector $\boldsymbol{x}$ is evaluated by the sigmoid function. The probability that the output is equal to 1, given $\boldsymbol{x}$, and parametrized by $\boldsymbol{\theta}$ is produced as follows:

$$P(y = 1|x; \theta) = h_\theta(x) \tag{27}$$

Alternatively, the probability of producing an output of zero, classified as false is as follows.

$$P(y=0|x;\theta)=1-P(y=1|x;\theta) \tag{28}$$

The cost function $J(\theta)$ is expressed in Equation (29).

$$J(\theta)=\frac{1}{N}\sum_{i}^{N} cost(h_\theta(x^i), y_{actual}^{(i)}) \tag{29}$$

where the term $y_{actual}^{(i)}$ corresponds to the (ith) target output in the training dataset [29]. The *cost* conditions are defined as follows.

$$cost\left(h_\theta(x), y_{actual}^{(i)}\right)=\begin{cases} -\log\left(h_\theta(x)\right), if\ y=1 \\ -\log\left(1-h_\theta(x)\right), if\ y=0 \end{cases} \tag{30}$$

Substituting the terms into Equation (29) produces the following cost function, which can be used to apply gradient descent optimization to minimize the error of the cost function.

$$J(\theta)=\frac{1}{N}\left[\sum_{i}^{N} y_{actual}^{(i)} \log\left(h_\theta\left(x^i\right)\right)+\left(1-y_{actual}^{(i)}\right)\log\ \left(1-h_\theta\left(x^i\right)\right)\ \right] \tag{31}$$

Taking the partial derivative of the cost function in Equation (31) is shown as follows:

$$\frac{\partial}{\partial\theta_j}J\left(\theta_j\right)=\frac{1}{N}\sum_{i=1}^{N}\left(y_{model}\left(x^{(i)}\right)-y_{actual}^{(i)}\right), for\ j=0 \tag{32}$$

$$\frac{\partial}{\partial\theta_j}J\left(\theta_j\right)=\frac{1}{N}\sum_{i=1}^{N}\left(y_{model}\left(x^{(i)}\right)-y_{actual}^{(i)}\right)\left(x^{(i)}\right), for\ j=1\ to\ (n+1) \tag{33}$$

where the term *n* is the number of input variables (features), and *j* indexes the parameter θ. The parameters can be updated using gradient descent as shown previously in Equations (23) and (24) until convergence.

3.3 Support Vector Regression (SVR)

Similar to the least mean square regression, where a model is fitted to produce an output with a minimized error. SVR uses a more complex algorithm that takes this a step further [30]. The example in Figure 12 shows data points that represent the predicted values of a model, y_{model}. The blue line is the target data, y_{actual}. The two dashed lines represent the boundaries of the SVR model, drawn a distance of ε away from the target dataset.

The parameter ε should be set and can be later optimized to produce more accurate results from the SVR model. The error calculations of the SVR consider exclusively the data points outside the boundary lines, as the area inside the boundaries is considered as a tolerance margin [31].

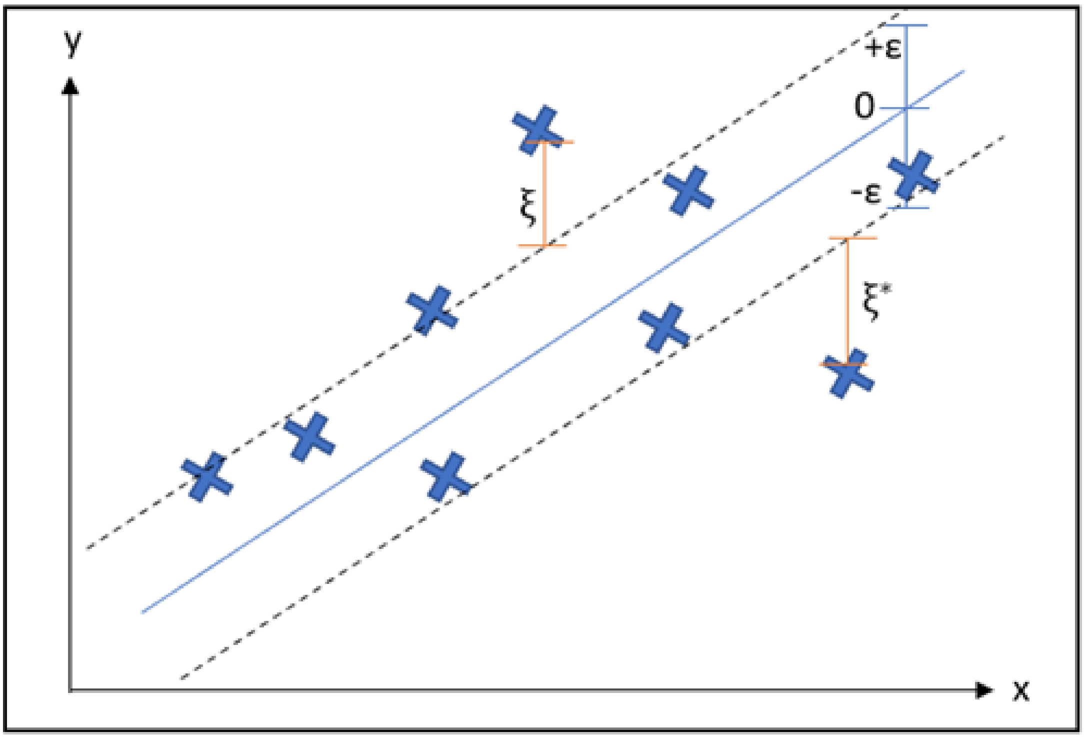


Figure 12. An example of SVR applied on a dataset.

The terms ξ and ξ* represent the distance between the boundary lines and the predicted values that fall outside the tolerance margin. The SVR model and boundaries can be represented as follows:

$$y_{model} = f(x) = \langle \boldsymbol{\omega}, \boldsymbol{x} \rangle + b = \sum_{i=1}^{N} \boldsymbol{\omega}_i \boldsymbol{x}_i + b \tag{34}$$

$$y_{upper\ boundary} = \langle \boldsymbol{\omega}, \boldsymbol{x} \rangle + b\ + \varepsilon \tag{35}$$

$$y_{lower\ boundary} = \langle \boldsymbol{\omega}, \boldsymbol{x} \rangle + b - \varepsilon \tag{36}$$

where N is the number of data samples, $\boldsymbol{x}_i$ is the ith training example, $\boldsymbol{\omega}$ is the learned weight vector, and the term b represents the bias. To train an SVR model, the following has to be solved.

$$\begin{gathered} Minimize\ \frac{1}{2} \|\omega\|^2 + C \sum_{i=1}^{N} \xi_i + \xi_i^* \\ Subject\ to \begin{cases} y_{actual}^{(i)} - \langle \boldsymbol{\omega}, \boldsymbol{x}_i \rangle - b \leq \varepsilon + \xi_i \\ < \boldsymbol{\omega}, \boldsymbol{x}_i > +b - y_{actual}^{(i)} \leq \varepsilon + \xi_i^* \end{cases} , C > 0 \end{gathered} \tag{37}$$

The parameter C is used as a constraint to control the penalty imposed on the prediction outputs that fall outside the boundaries, this can help in preventing the model from over-fitting.

3.4 Artificial Neural Networks

The three main components that make up the human nervous system are receptors, effectors, and the central nervous system. Stimuli affecting the human body are translated by receptors in the form of electrical impulses. These signals are transmitted by the brain into the central nervous system to produce a physical response. The building block of a neural system is a neuron.

The structure of a biological neuron is illustrated in Figure 13. It consists of a cell body that receives the incoming signal from other neurons through dendrites. The output signal of a neuron is transmitted through a nerve fiber, known as an axon. The axon branches into other interconnected axons that lead to other neurons in the network [32].

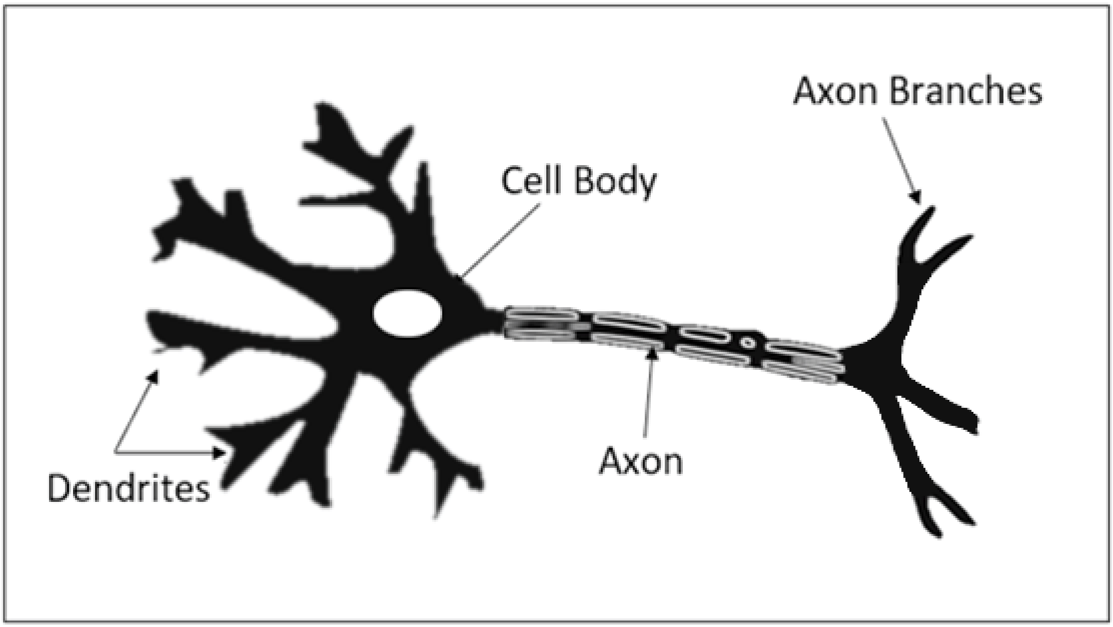


Figure 13. The structure of a biological neuron.

The cell body of a neuron (Soma) is in charge of processing the incoming input signals, and the decision of what neuron should send an output signal. Neurons typically receive multiple input signals from their dendritic trees. The case of whether a neuron sends an output signal depends on the weighted sum of all the received input signals. This process is decided in the soma, and the output signal of the neuron is transmitted through the axon to other neurons in the network.

The artificial neuron is known as a perceptron. It is the basic unit in an artificial neural network (ANN), which is made up of multiple perceptrons. The inner workings of an artificial neuron imitate that of a biological neuron. The perceptron takes the weighted input sum and produces an output, that is based on a mathematical activation function.

ANNs not only share the name with biological neuron networks, but they also share a similar structure and mechanism. ANNs were created as learning algorithms that mimic how the mammal neural system learns and analyzes. ANNs have been around since the 1940s but they went through phases of interest that diminished after some time [33].

Using ANNs was inconvenient and inefficient in the past as a result of low computational power and lack of quality data, but with the boom in technological advancement witnessed every year, and the use of the Internet to store and transmit data, in addition to newer more complex ANNs architecture being discovered made neural networks a crucial part in today's development projects [1].

Applications of ANNs are used by everyone daily, such as a music application learning its user's taste to suggest similar picks, spam e-mail detection, and classification, or facial recognition technology currently used in phones. The structure of an artificial neuron, which is known as a perceptron is illustrated in Figure 14.

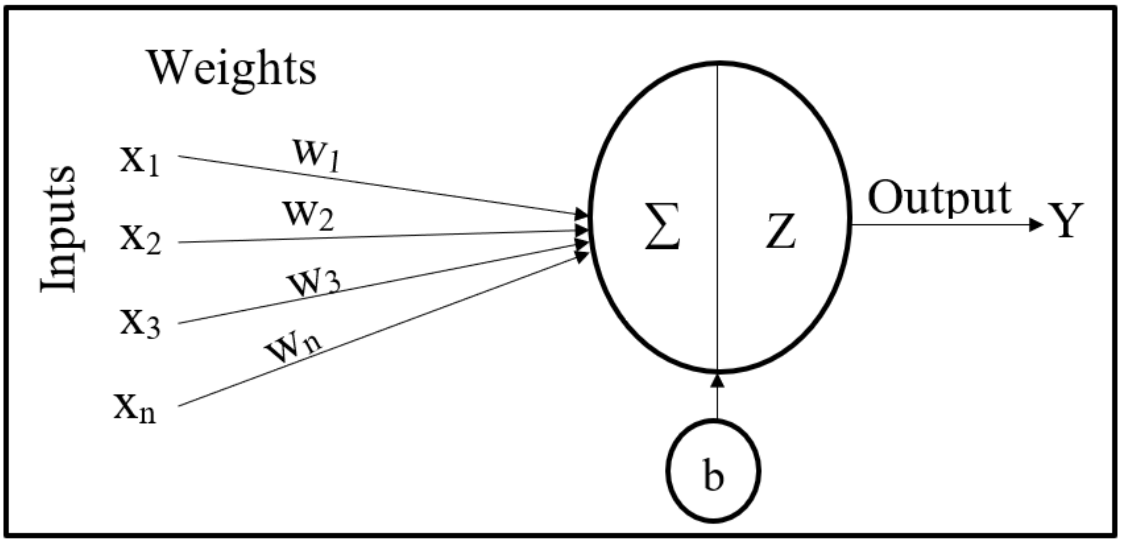


Figure 14. The basic structure of a Perceptron.

In the perceptron, each input is multiplied by a weight value, then a summation with a bias value is performed. The result is then passed into an activation function, Z, which outputs the value Y, as the output of the neuron [34]. The output of an artificial neuron is expressed in Equation (38).

$$Y = Z(\sum_{i=1}^{N} w_i x_i + b) \tag{38}$$

where the input X, and weights W, are in the following form, and n is the number of input features.

$$\boldsymbol{X} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix}, \boldsymbol{W} = [w_1 \quad w_2 \quad w_3 \quad \dots \quad w_n] \tag{39}$$

A few unique characteristics of ANNs include:

1- ANNs can learn complex tasks.

2- ANNs ease the process of constructing a model with a large number of inputs/outputs.

3- The cooperative behavior of the artificial neurons in the network defines its computational power, and no individual artificial neuron carries precise information.

3.4.1 Multi-Layer Perceptrons (MLP)

MLPs consist of feed-forward and fully connected layers made up of perceptrons, resulting in a more complex network. An example of the structure of an MLP can be seen in Figure 15.

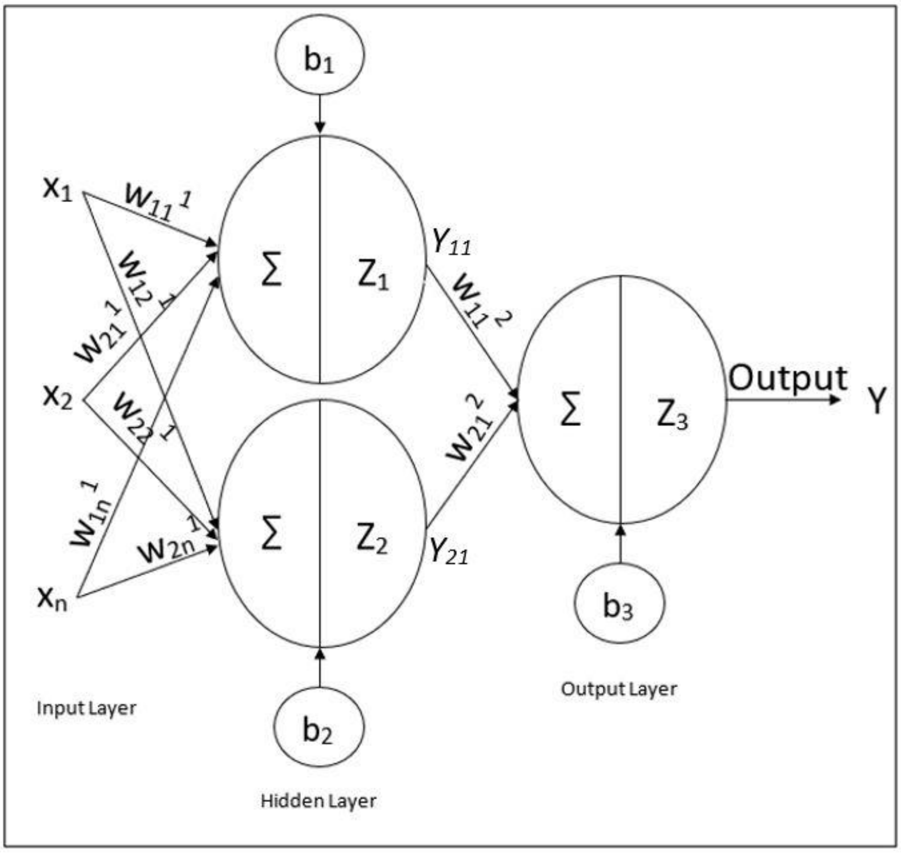


Figure 15. The basic structure of a multi-layer perceptron (MLP).

Similar to the ANN structure shown earlier in Equation (38), the output of the MLP in Figure 15 is found to be the following:

$$Y = Z\left[\left[Z\left(Y_{11}{W_{11}}^{2}\right) + Z\left(Y_{21}{W_{21}}^{2}\right)\right] + b_3\right] \qquad (40)$$

For the network to learn, the first step starts with initializing the network parameters randomly. The next step is forward propagation (FP), which involves passing the training dataset items from the input throughout the output. The loss (error) between the actual and target (truth) output is computed.

Next, the error is back-propagated from the output layer through the entire network, and back to the first hidden layer of the MLP. During the process of backpropagation (BP), the sensitivities of the different weights and biases on the network are iteratively optimized. The combined process of forward-propagation, backpropagation, and updating the network parameters is iterated until convergence, i.e., the sensitivities start to vanish in each layer of the network. The forward propagation (FP) from layer $l-1$ to l can be expressed as follows:

$$y_k^l = Z\left(x_k^l\right) = x_k^l = \theta_k^l + \sum_{i=1}^{N} w_{ik}^{l-1} y_i^{l-1} \tag{41}$$

where y_k^l is the output, x_k^l is the input of the *k*th neuron in layer l, of the *k*th neuron in layer l, w_{ik}^{l-1} is the weight between the *i*th neuron in the previous layer to the *k*th layer in the current layer and Z is the activation function of the *k*th neuron, respectively. One of the commonly used loss functions is MSE which is expressed in Equation (42), where y_i^L is the actual output and Y_i is the target output from the training set.

$$J = \frac{1}{2}\sum_{i=1}^{N}(y_i^L - Y_i) \tag{42}$$

The weight and bias sensitivities are computed as follows:

$$\frac{\partial J}{\partial w_{ik}^{l-1}} = \frac{\partial J}{\partial x_k^l}\frac{\partial x_k^l}{\partial w_{ik}^{l-1}} = \frac{\partial J}{\partial x_k^l} y_i^{l-1} \tag{43}$$

$$\frac{\partial J}{\partial \theta_k^l} = \frac{\partial J}{\partial x_k^l}\frac{\partial x_k^l}{\partial \theta_k^l} = \frac{\partial J}{\partial x_k^l} \tag{44}$$

A dependency is revealed between the sensitivities and the partial derivative of the loss with respect to the input of the neuron, x_k^l. This derivative, $\frac{\partial J}{\partial x_k^l}$ is known as delta error of the neuron ($\delta_k^l = \frac{\partial J}{\partial x_k^l}$). At the output layer, it can be directly be computed as follows:

$$\delta_k^l = \frac{\partial J}{\partial x_k^l} = \frac{\partial J}{\partial y_k^l} Z'\left(x_k^l\right) = \left(y_k^l - Y_k\right) \cdot Z'\left(x_k^l\right) \tag{45}$$

Using the chain rule of partial derivatives, the backpropagation of the delta errors can be expressed as follows:

$$\delta_k^l = Z'\left(x_k^l\right) \sum_{i=1}^{N_{l+1}} \delta_i^{l+1} w_{ki}^l \tag{46}$$

When the delta errors are back-propagated and parameter sensitivities are computed for all neurons in the network, they can be cumulated for a certain number of items in the train set. This is called "mini-batch" and its size can vary between 1 to the number of items in the train set. If a mini-batch size is 1, then it is called "online" learning and if it is equivalent to the size of the train set, it is called "batch" learning. There are several optimization methods such as Stochastic Gradient Descent (SGD) [34], SGD with momentum, AdaDelta [35], and Adam [36]. The most basic optimizer is SGD which can be expressed as follows:

$$w_{ik}^l = w_{ik}^l - \alpha \frac{\partial J}{\partial w_{ik}^{l-1}} \tag{47}$$

$$\theta_k^l = \theta_k^l - \alpha \frac{\partial J}{\partial \theta_k^l} \tag{48}$$

This process will continue until the parameters converge to an optimal value that minimizes the output error of the model, compared to the target output of the training dataset. The backpropagation algorithm to train the MLP network can be expressed in the following steps:

1. Building the network by selecting the number of hidden layers and neurons in each layer.
2. Randomly initializing the network parameters (weights and biases).
3. Using the testing dataset to apply forward propagation through the network.
4. Computing the error between the network output and the targets from the testing dataset.
5. Computing the delta errors of the output layer.
6. Computing the delta errors starting from last the hidden layer until the input

layer.

7. Computing the parameter sensitivities (derivatives).
8. Applying an optimization method to update the weights and biases accordingly.
9. Repetition of steps 3 through 8 for each example in the training dataset.

3.4.2 Recurrent Neural Networks (RNNs)

Feedforward networks, for example, MLPs, produce outputs that are assumed to be independent of the inputs of the network. This can be inefficient when the input data is made up of sequences, for example, words or time-series-based data such as the stock market prices [37].

This is where RNNs have an advantage because of having a feedback mechanism, known as the hidden state. A general structure of the RNN structure folded and unfolded can be seen in Figure 16.

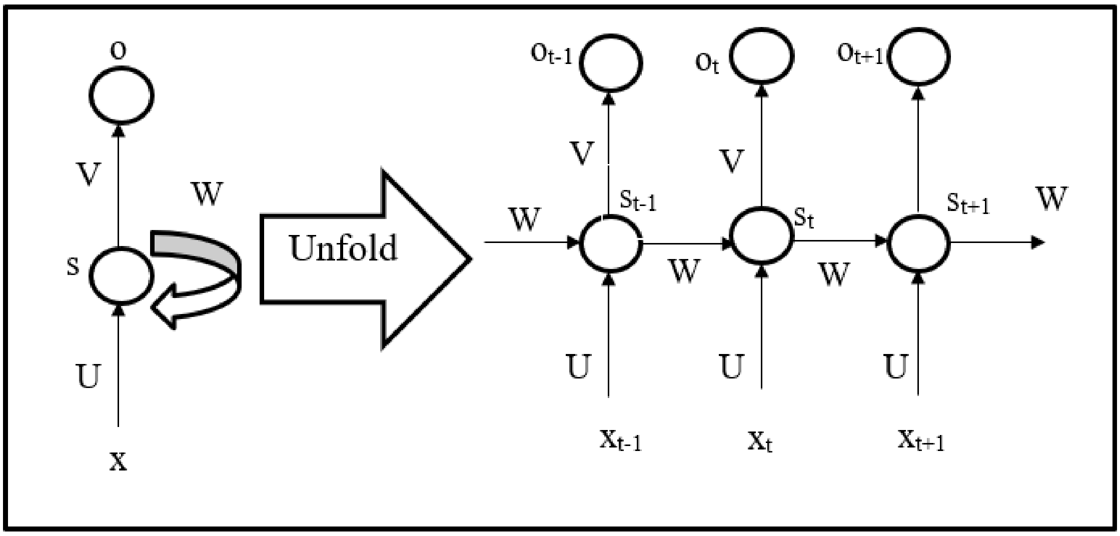


Figure 16. Unfolding the recurrent neural network (RNN).

The hidden state (s_t) is known as the memory of the network that depends on the previous layer's hidden state, in addition to the input of the current layer, resulting in Equation (49).

$$s_t = Z(Ux_t + Ws_{t-1}) \tag{49}$$

The vanishing gradient problem appears when the backpropagation algorithm propagates back through the RNN, passing through all the neurons in the network to update the weights. The cost function computed at deeper layers in the network will be used to update the weights of the shallower layers in the network. The problem is encountered when the gradient that has been calculated in deeper networks is multiplied back through earlier weights in the network, causing the gradient to slowly dimmish throughout the network layers.

The factor at which the gradient diminishes through an RNN is known as W_{rec}. It causes two potential problems. When the value of W_{rec} is small, the RNN suffers from the vanishing gradient problem. Contrary, when the value is large, the network experiences the exploding gradient problem [33]. This problem may cause the RNN to consume a long time to converge while training or fail to converge in some cases.

3.4.3 Long-Short Term Memory (LSTM)

As a variant of RNNs, LSTM has the advantage of being able to forget irrelevant information, this helps the network overcome the vanishing gradient problem. The LSTM structure can be seen in Figure 17.

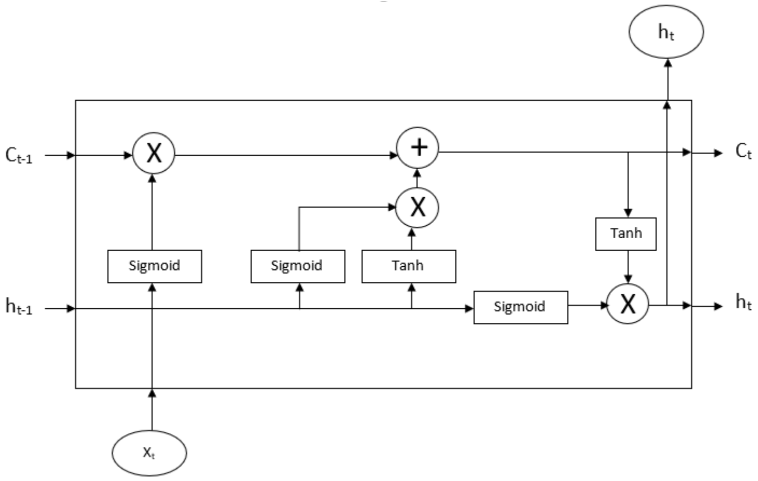


Figure 17. Basic long-short term memory (LSTM) module.

Where C$_t$ is known as the memory cell, and h$_t$ is the output of the LSTM module. The input X$_t$ is manipulated through the module's different states. The multiplication and addition operations can be used as valves to decide how much of the input information the memory can remember or forget. LSTM module contains three different gates, the input, forget and output gates, simplified in Figure 18.

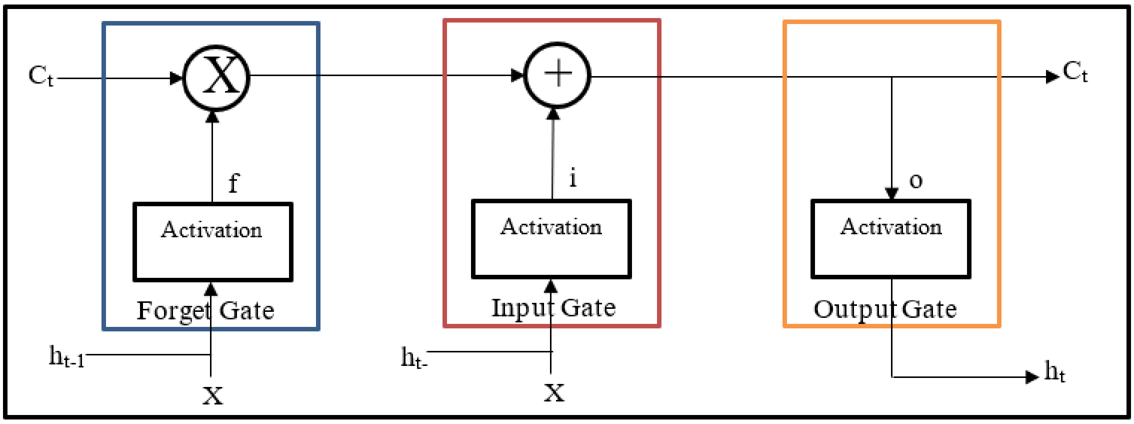


Figure 18. The different gates inside an LSTM module.

As shown in Figure 18, the multiplication gate in the LSTM module allows the layer to accumulate and access information over a longer period, which overcomes the problem of vanishing gradient that may cause the network to forget the initial input. Information stored in the cell will be overwritten by new information coming from the network only if the gate activation changes to one [33], causing the gate to open. The equations corresponding to each gate are as follows:

$$i_t = Z(w_i[h_{t-1}, x_t] + b_i), input\ gate\ output \tag{50}$$

$$f_t = Z(w_f[h_{t-1}, x_t] + b_f), forget\ gate\ output \tag{51}$$

$$o_t = Z(w_o[h_{t-1}, x_t] + b_o), output\ gate\ output \tag{52}$$

where the value of the cell state (C$_t$) is found using Equations (53)(54), the term $\widetilde{C}_t$ is the initial cell state.

$$\widetilde{C}_t = Z(w_c[h_{t-1}, x_t] + b_c) \tag{53}$$

$$C_t = f_t * C_{t-1} + i_t * \widetilde{C}_t \tag{54}$$

Finally, the output of the LSTM module is found using equation (55).

$$h_t = o_t * Z(C_t) \tag{55}$$

3.4.4 Hyperparameters Tuning

The main problem with ANNs is the fact that they require many hyperparameters to be tuned for the desired learning performance. This would be a challenge as applying the trial-and-error methodology can be time-consuming.

In this study, a grid search system was created with a pre-defined list of recommended hyperparameters, based on previous studies. The system shuffles and tests all the possible combinations of hyperparameters against their output accuracy and R^2 score. These are the parameters used in tuning the neural networks throughout this research, divided into classification and regression problems.

3.4.4.1 Optimizers

Choosing the right optimizer can be important for faster convergence in BP and a better generalization. The results of this research are divided into classification and regression tasks. The evaluated optimizers are presented in Table 4.

Table 4. Different types of optimizers are used for evaluation.

Classification	**Regression**
Adam [36]	Adam [36]
AdaDelta [35]	SGD [34]

3.4.4.2 Loss Functions

Loss functions are used by the optimization algorithm to repeatedly estimate the error of the model. The optimizer will update the parameters of the network accordingly, to produce a lower error in the next iteration.

Table 5. The different types of loss functions will be used to evaluate the regression and classification problems.

Classification	**Regression**
Binary Crossentropy (8)	Mean Absolute Error (11)
Mean Squared Error (12)	Mean Squared Error (12)

3.4.4.3 Performance Metrics

Performance metrics are used to test the model performance. They are applied to the model after the training process, using the testing dataset as input for the model, to measure the performance on unseen data.

Table 6. The different types of performance metric functions will be used to evaluate the regression and classification problems.

Classification	**Regression**
Binary Crossentropy (8)	Mean Absolute Percentage Error (9)
Accuracy (4)	Mean Squared Error (12)

3.4.4.4 Activation Functions

In an ANN, each node contains an activation function that alters the input to produce an output. The output can be linear or non-linear depending on the type of activation function. The grid search system was used to evaluate the activation functions presented in
Table 7.

- Rectified Linear Unit (ReLU): ReLU is a commonly used piecewise linear function as expressed in Eq. (56). It produces a non-zero output only if the input is positive; otherwise, the function will produce zero as the output. The function is commonly used due to its computational efficiency.

$$Z(x) = \begin{Bmatrix} x & x > 0 \\ 0 & x \leq 0 \end{Bmatrix} \tag{56}$$

- Exponential Linear Unit (ELU): Unlike other activation functions, the ELU activation function contains a positive factor within it, allowing it to converge faster and produce more accurate results. As in (57), it can produce negative and positive outputs, making it possible to use in the input layer.

$$Z(x) = \begin{Bmatrix} x & x > 0 \\ \alpha(e^z - 1) & x \leq 0 \end{Bmatrix} \tag{57}$$

- Hyperbolic Tangent (Tanh): As expressed in (58), Tanh shares its non-linearity with the sigmoid function, but it has the advantage of being zero-centered. Recently it became a very popular choice for an output layer activation function as it produces a stronger gradient. The main problem with such a function is the problem of vanishing gradient, which can cause the model to fail from converging.

$$Z(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{58}$$

- Scaled Exponential Linear Unit (SELU): Similar to ELU, the main difference is that the result is multiplied by a scaling factor that is used to help in improving the training time and allows the model to converge faster.

$$Z(x) = \begin{Bmatrix} \lambda x & x > 0 \\ \lambda\alpha(e^z - 1) & x \leq 0 \end{Bmatrix} \tag{59}$$

- Swish: Similar to the sigmoid function, the main difference is the addition of a scaling factor that helps the model converge faster.

$$Z(x) = \frac{x}{1 + e^{-\beta x}} \tag{60}$$

Table 7. The different types of activation functions will be used to evaluate the regression and classification problems.

Classification	**Regression**
Relu	Elu
Tanh	Relu
Sigmoid	Selu
Swish	Tanh

3.4.5 Short-Term Financial Data Prediction by Long-Term Regression

To produce the learning and prediction data inputs for the models, a moving window has been created. The learning and prediction window's size and the number of segmentation from the original dataset are the hyperparameters to be set to some practical values in advance. An illustration of this is in Figure 19.

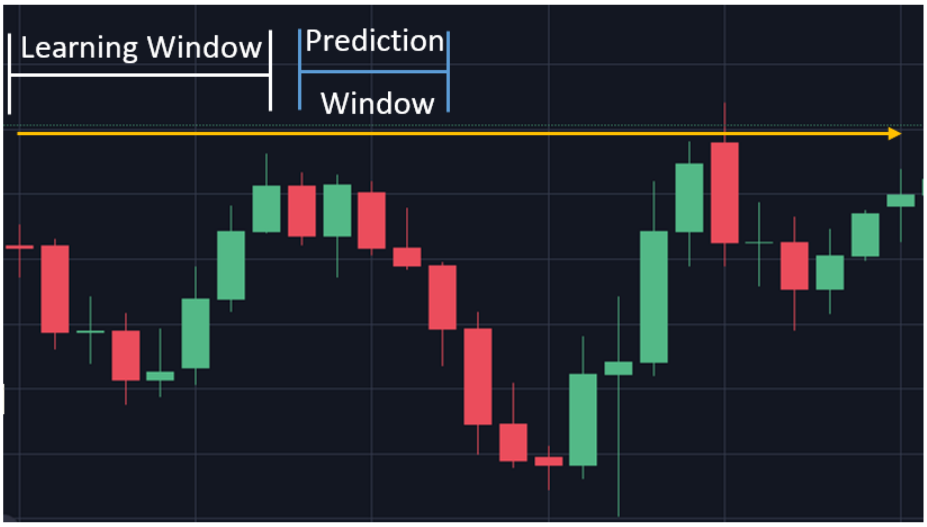


Figure 19. The sliding window is used to construct the training and testing datasets.

Each candle in the exemplary figure displays the price of one day, the learning and prediction windows are set to be 8 and 4 days, respectively, representing one segment. The moving window will slide through the specified number of segmentation to create a learning set of 8 days of the previous price, which is used by the model to produce a prediction 4 days into the future.

CHAPTER 4: RESULTS AND DISCUSSION

4.1 Experimental Setup

4.1.1 Hardware and Software Specifications

The study was conducted on Python (v3.7) using Keras (v2.4.3) and TensorFlow (v2.0) APIs to construct, train and evaluate all the models produced in this thesis. The hardware configuration of the machine is as follows:

Table 8. Hardware specification of the system used throughout the thesis

CPU	Intel Core i7-6700k @ 4.00-GHz
GPU	Nvidia GTX 1080 Ti with 10-GB
RAM	32-GB

4.1.2 Accuracy Calculations

The most crucial element when forecasting a financial asset is correctly predicting the future price trend. Predicting the future direction of the price, going up or down, is quantified in the form of accuracy in this study.

This is calculated by producing a prediction from the model, this prediction is compared against the target future price. If the future price is above the current day price, meaning the price should increase, then the current day price is compared again with the model predicted price. If the model future prediction shows an increase in the price, then this is stored, as a correct prediction.

In the case where the model is incorrect, this is also stored, but as a false prediction. The same logic is applied, in reverse, for the downward trend case. This process is computed for all the examples in the dataset, and the accuracy is calculated by applying Equation (4).

4.2 Linear Regression Results

After compiling and experimenting on the dataset, interesting relationships have been discovered between the different trading assets. Examples of such relationships are illustrated in Figure 20, where the joint plot of EUR/USD is constructed against the three assets that produce the highest correlation coefficient, as deduced from Figure 5. Such relationships will be explored by building simple and multiple linear regression models to investigate the effectiveness of using highly correlated features to predict the future price of EUR/USD.

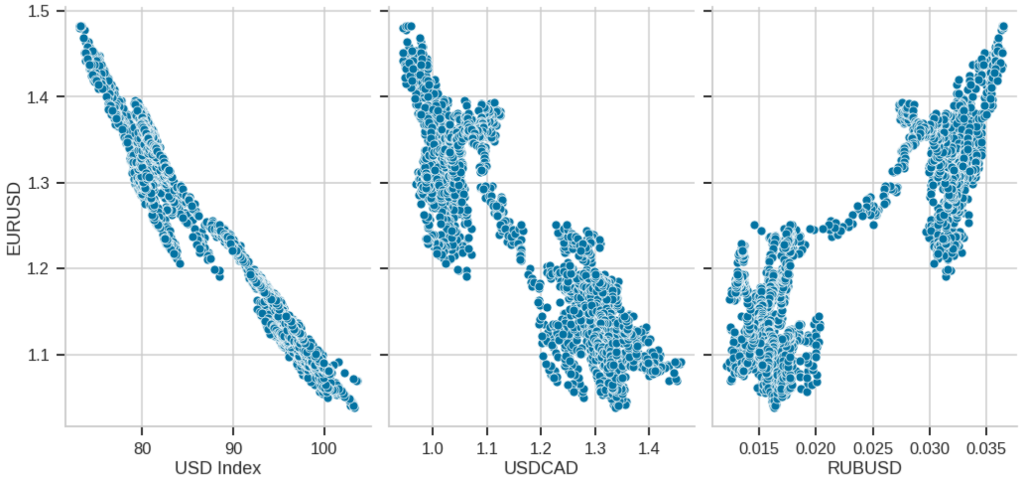


Figure 20. EUR/USD joint plot against USD Index, USD/CAD, and RUB/USD.

4.2.1 Simple Linear Regression Results

The simple linear regression model was constructed using USD Index as the input. The input and output were created using the moving window to train and test the model. A shift of one day was implemented, as a result, the model is constructed to use the closing price of each day as input, to predict the closing price of the next day. The data are then split into 80% and 20% for training and testing respectively. The linear regression model is expressed as follows:

$$EUR/USD = 2.418 - 0.0134 * USD\ Index \tag{61}$$

The model is applied to the testing and training set to produce the RMSE, MAE, $R^{2,}$ and accuracy. In addition to the 10-fold cross-validation results produced using the model to evaluate the R^2 score, shown in Table 9.

Table 9. Simple linear regression model RMSE, MAE, R^2, and accuracy results, including 10-fold CV R^2 scores.

	Test set	**Train set**
RMSE	0.015	0.025
MAE	0.012	0.020
R^2	0.812	0.951
Accuracy (%)	50.053	

10-fold CV R^2 scores										
1	2	3	4	5	6	7	8	9	10	**Average**
0.824	0.746	-1.383	-0.363	0.962	0.245	0.948	0.595	0.443	0.896	0.391

Analyzing the results found in Table 9, the simple linear regression model shows an average R^2 score of (0.391) caused by the inconsistent performance of the model on the different segments of the CV set. The model produced an accuracy of (50.053%), meaning that the model fails to predict future trends of EUR/USD price for the next day. The results of the simple linear regression model will be used as a baseline to compare the performance of other models. The model prediction is plotted against the actual value of EUR/USD in Figure 21.

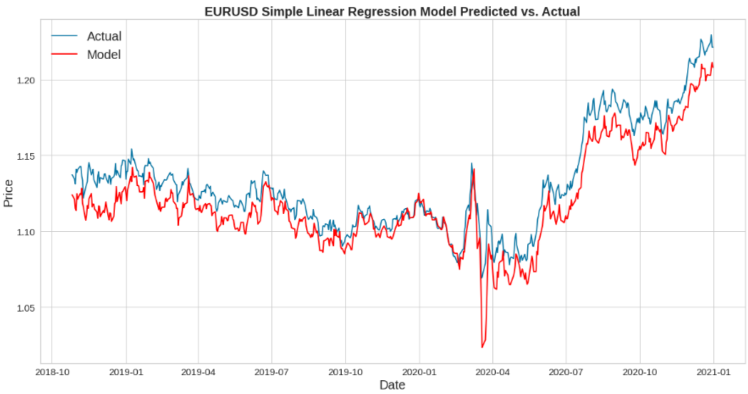


Figure 21. Simple linear regression model of EUR/USD showing predicted vs. actual price.

4.2.3 Multiple Linear Regression Results

The model is constructed using the three input features seen in Figure 20. Using USD Index, USD/CAD, and RUB/USD the model is found in Equation (62) and the results are extrapolated in Table 10.

$$EUR/USD = 2.379 - 0.019 * USD\ Index + 0.401 * USD/CAD + 1.477 * RUB/USD \tag{62}$$

Table 10. Multiple linear regression model RMSE, MAE, R^2, and accuracy results, including 10-fold CV R^2 scores.

	Test set	**Train set**
RMSE	0.009	0.020
MAE	0.007	0.016
R^2	0.932	0.968
Accuracy (%)	49.982	

10-fold CV R^2 scores

1	2	3	4	5	6	7	8	9	10	**Average**
0.883	0.816	-0.414	0.336	0.950	-0.356	0.836	0.809	0.609	0.957	0.542

Similar to the results obtained using the simple linear regression model in Table 9, the multiple linear regression results in Table 10 shows inconsistency in the 10-fold CV R^2 scores, averaging to a value of (0.542). This is an improvement of +28% over the simple linear regression average R^2 score. The results, however, show the model's inability to predict the future direction of the EUR/USD, with an accuracy of (49.982%). The model output is plotted against the actual EUR/USD prices in Figure 22.

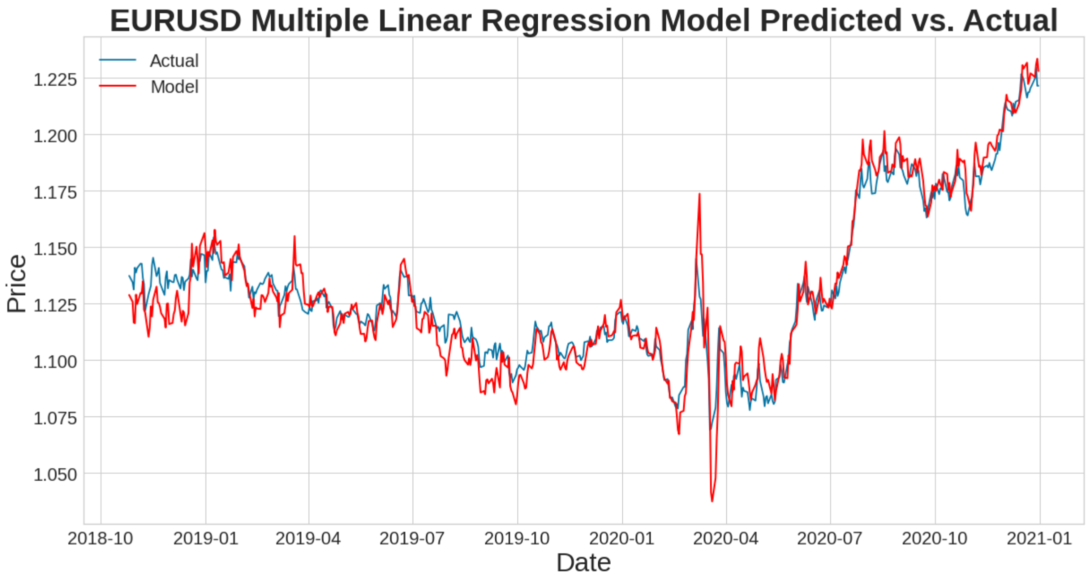


Figure 22. Multiple linear regression model output against the actual value of EUR/USD.

4.3 Support Vector Regression (SVR) Results

The SVR model was built using two kernels, linear and radial basis function (RBF). Linear SVR kernel is commonly used when the data can be linearly separated, it can also be efficient when a large number of features is used to create a model. Another commonly used kernel is the radial basis function (RBF), where the function computes the inner product of two projected vectors, using a radial-based transformation [34].

The implementation of the model required applying Equation (1) on the training and testing datasets, to standardize the data before fitting the model. The USD Index is used as the model input. The output was prepared using the moving window, the closing price of the current day is used to predict the price of EUR/USD for the next day. Table 11 shows the results obtained from both linear and RBF SVR kernels.

Table 11. SVR (Linear and RBF) models RMSE, MAE, R^2, and accuracy results, including 10-fold CV R^2 scores.

	Linear SVR Kernel		**RBF SVR Kernel**	
	Test set	**Train set**	**Test set**	**Train set**
RMSE	0.008	0.007	0.016	0.007
MAE	0.006	0.006	0.008	0.005
R^2	0.942	0.950	0.879	0.957
Accuracy (%)	61.425		56.406	

10-fold CV R^2 scores										
Linear SVR Kernel										
1	2	3	4	5	6	7	8	9	10	**Average**
0.815	0.766	-0.115	0.015	0.971	0.646	0.966	0.657	0.552	0.918	0.619
RBF SVR Kernel										
1	2	3	4	5	6	7	8	9	10	**Average**
0.822	0.297	0.851	-1.685	0.96	0.376	0.961	0.696	0.892	0.939	0.511

The SVR results seen in Table 11 shows an improvement of the average 10-fold CV R^2 score by 37% for the linear kernel, and by 23% with the RBF kernel, over the simple linear regression model, seen in Table 9.

The most significant change, compared to the previous model's performances, was in terms of future trend prediction accuracy. The SVR linear kernel model scored 61% accuracy, and 56% for the RBF kernel model. The two SVR model predictions are plotted against the actual EUR/USD testing set, illustrated in Figure 23. The increase in performance of the linear kernel over RBF can be explained by its ability to perform better in linear separation.

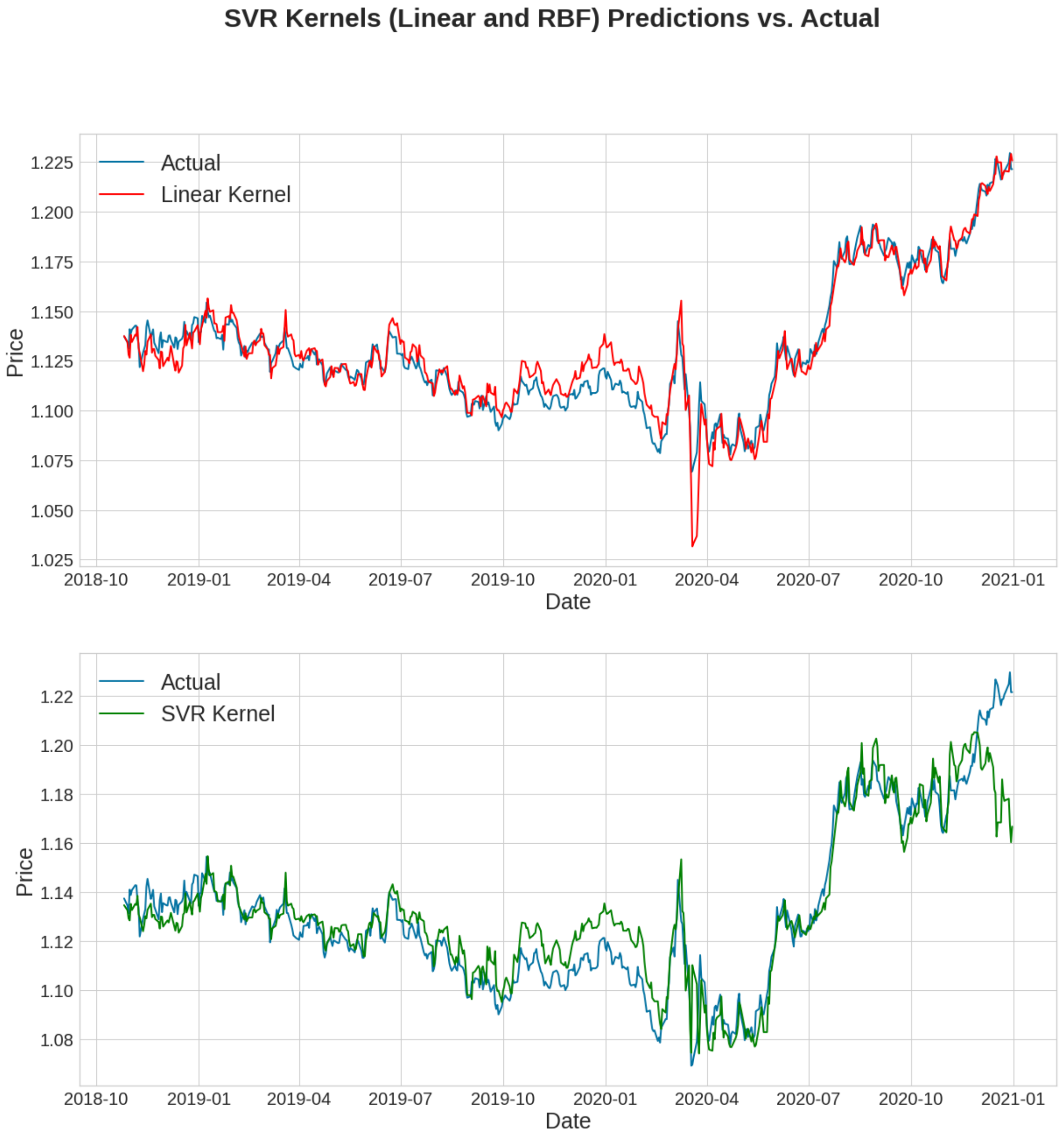


Figure 23. SVR kernels outputs vs. actual EUR/USD prices.

4.5 Results by Using ANNs

The construction of ANNs requires the selection of many hyperparameters. This process is made easier using the grid search system to find the parameters that produce the best results.

4.5.1 MLP Regression Results

To build the MLP model, the input was set to be the current day closing price of EUR/USD, USD Index, and USD/CAD. The target output has been constructed using the moving window, training the network to predict one day into the future. The grid search system was implemented to find the hyperparameters.

The best model is found to have the parameters shown in Table 12. Training the MLP network with the selected hyperparameters took 172 seconds to complete. The training and validation losses are plotted over the number of iterations, displayed in Figure 24.

Table 12. MLP regression problem model, grid search selected parameters.

Hyperparameter	Selected parameters for the regression MLP model
Optimizer	Adam
Loss function	MeanAbsoluteError
Metric function	MeanSquaredError
Batch size	1
Number of iterations	100 epochs
Input layer	Input size = 3, number of neurons = 3, activation function = Selu
Hidden layer	Number of neurons = 50, activation function = Selu
Output layer	Output size = 1, number of neurons = 1, activation function = elu
Number of trainable parameters	263

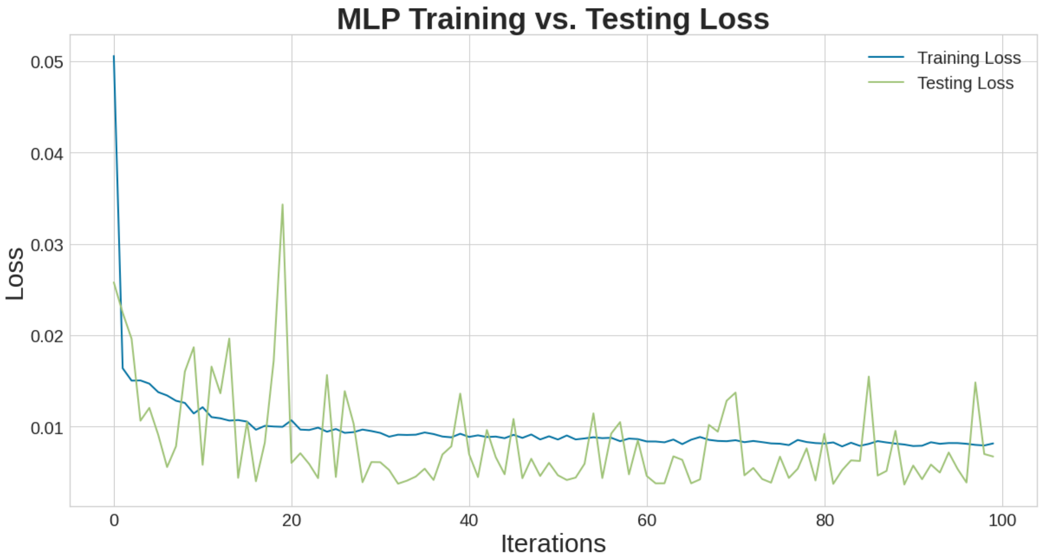


Figure 24. The MLP regression model training and testing loss plot, over the number of iterations.

The output results in Table 13 show an extraordinary improvement of the 10-fold CV average R^2 score by 60% over the simple linear regression model. The MLP scored a consistent R^2 score above 0.9 across all CV sets, which all previous models failed to achieve. However, the MLP model scored a future trend prediction accuracy of (52%), which translates to the fact that the model fails to identify the next-day trend direction accurately. The predicted output of the model, based on the testing set is plotted in Figure 25.

Table 13. MLP regression model RMSE, MAE, R^2, and accuracy results, including 10-fold CV R^2 scores.

	Test set	**Train set**
RMSE	0.008	0.011
MAE	0.007	0.009
R^2	0.945	0.990
Accuracy (%)	52.404	

10-fold CV R^2 scores										
1	2	3	4	5	6	7	8	9	10	**Average R^2**
0.941	0.937	0.950	0.946	0.978	0.988	0.990	0.990	0.991	0.992	0.970

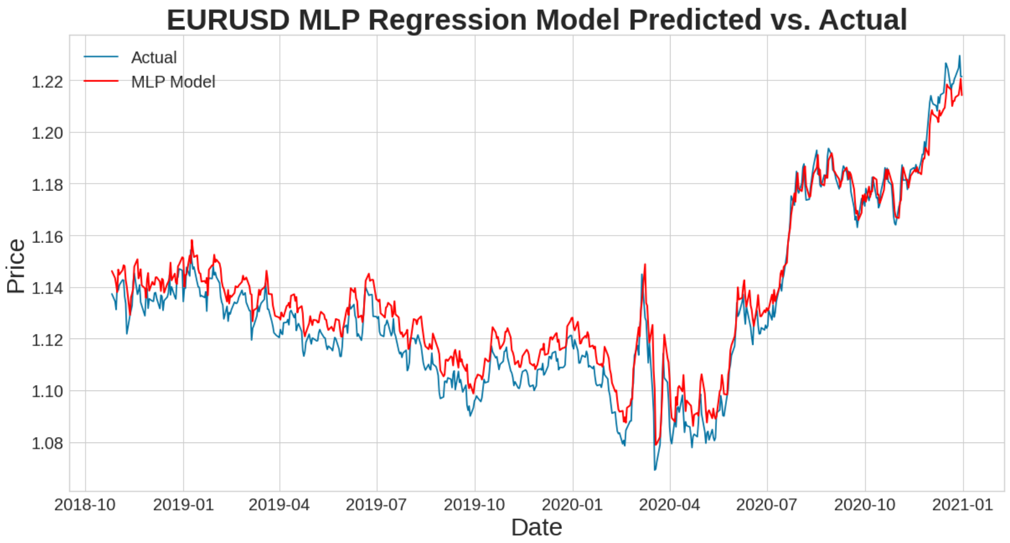


Figure 25. MLP regression model prediction plotted against the test set of EUR/USD and both training and testing set.

4.5.2 MLP Classification Results

The MLP classification model is built to predict the future direction of the EUR/USD. This is done by constructing a learning dataset that uses the current day closing price of the EUR/USD, USD Index, and USD/CAD as inputs, and interpolating an output that predicts the next day closing price direction of the EUR/USD. The model will have two classification categories, the closing price will be higher than the current day closing price, or lower.

The input and output datasets are then split into training and testing, with a size of 80% and 20% respectively. The input data are then standardized using Equation (1) before passing it through the MLP classification model for training. The model hyperparameters are obtained by selecting the grid search system's best result, seen in Table 14. Training the model consumed an estimated time of 141 seconds. The training and testing accuracies are plotted over the total number of iterations in Figure 26.

Table 14. MLP classification problem model, grid search selected parameters.

Hyperparameter	Selected parameters for the classification MLP model
Optimizer	AdaDelta
Loss function	binary_crossentropy
Metric function	accuracy
Batch size	1
Number of iterations	300 epochs
Input layer	Input size = 3, number of neurons = 3, activation function = tanh
Hidden layer	Number of neurons = 20, activation function = elu
Output layer	Output size = 1, number of neurons = 1, activation function = sigmoid
Number of trainable parameters	113

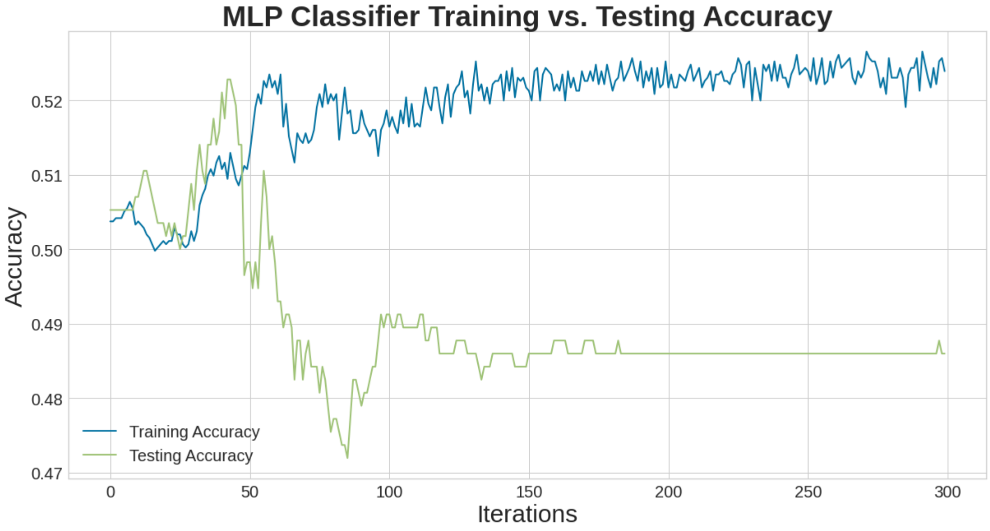


Figure 26. The MLP classification model training and testing accuracy plot, over the number of iterations.

The testing dataset is applied to the model to test the performance. Table 15 displays the accuracy, F1-score, in addition to the 10-fold CV accuracy across both datasets. The model results show the inability of the model to produce accurate results, as the future predictions appear to be random when the input is one day of closing prices. The confusion matrix in Table 16 supports this conclusion.

Table 15. MLP classification model accuracy, F1-score, and 10-fold CV accuracy.

	Test set	**Train set**
Accuracy (%)	48.596	52.435
F1 score	0.383	0.522

10-fold CV Accuracy (%)										
1	2	3	4	5	6	7	8	9	10	**Average Accuracy (%)**
47.183	47.535	48.474	48.679	50.211	50.234	50.150	50	50.273	49.964	49.271

Table 16. MLP classification model confusion matrix.

	Test set		**Train set**	
	Predicted False	Predicted True	Predicted False	Predicted True
Actual False	186	96	603	537
Actual True	197	91	547	592

4.5.3 LSTM Regression Results

To build the LSTM model, hyperparameters were selected based on the best results obtained through the grid search system. LSTM model training proved to be the most time-consuming, out of all the previously used models. Training the model with the hyperparameters shown in Table 17 consumed an estimated time of 1266 seconds. An identical input structure to the MLP model was implemented.

Using the current day closing price of EUR/USD, USD Index, and USD/CAD to predict the next day's EUR/USD closing price. Input data were split into 80% and 20% training and testing sets, respectively. The data were standardized before being passed to the model, using Equation (1). The model training and testing losses throughout the entirety of the iterations are plotted in Figure 27.

Table 17. LSTM regression model hyperparameters were obtained using the grid search system.

Hyperparameter	**Selected parameters for the regression LSTM model**
Optimizer	SGD
Loss function	MeanSquaredError
Metric function	MeanAbsolutePercentageError
Batch size	1
Number of iterations	100 epochs
Input layer	Input size = (3,3), number of neurons = 3, activation function = linear
1st hidden layer	Number of neurons = 30, activation function = tanh
2nd hidden layer	Number of neurons = 15, activation function = tanh
Output layer	Output size = 1, number of neurons = 1, activation function = relu
Number of trainable parameters	6,916

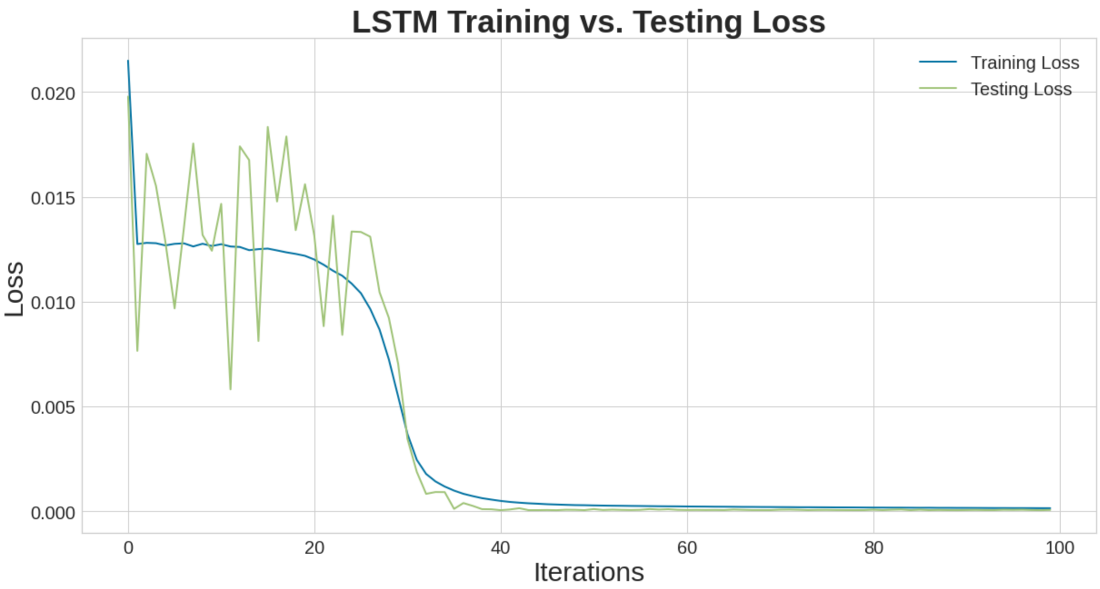


Figure 27. LSTM regression model training and testing loss plot, over the number of iterations.

Similar to the MLP model results, the LSTM model achieved an improvement of 59% over the average R^2 score, when compared to the simple linear regression. The model had a constant R^2 score higher than 0.9 across all the 10-fold CV sets. However, the model still fails to accurately predict the next-day trend, achieving a very poor accuracy of 49%. The results are shown in Table 18. The LSTM regression model testing set output is plotted against the EUR/USD in Figure 28.

Table 18. LSTM regression model RMSE, MAE, R^2, and accuracy results, including 10-fold CV R^2 scores.

	Test set	**Train set**
RMSE	0.008	0.012
MAE	0.006	0.009
R^2	0.947	0.988
Accuracy (%)	49.649	

10-fold CV R^2 scores

1	2	3	4	5	6	7	8	9	10	**Average R^2**
0.951	0.932	0.941	0.941	0.976	0.986	0.988	0.989	0.989	0.989	0.968

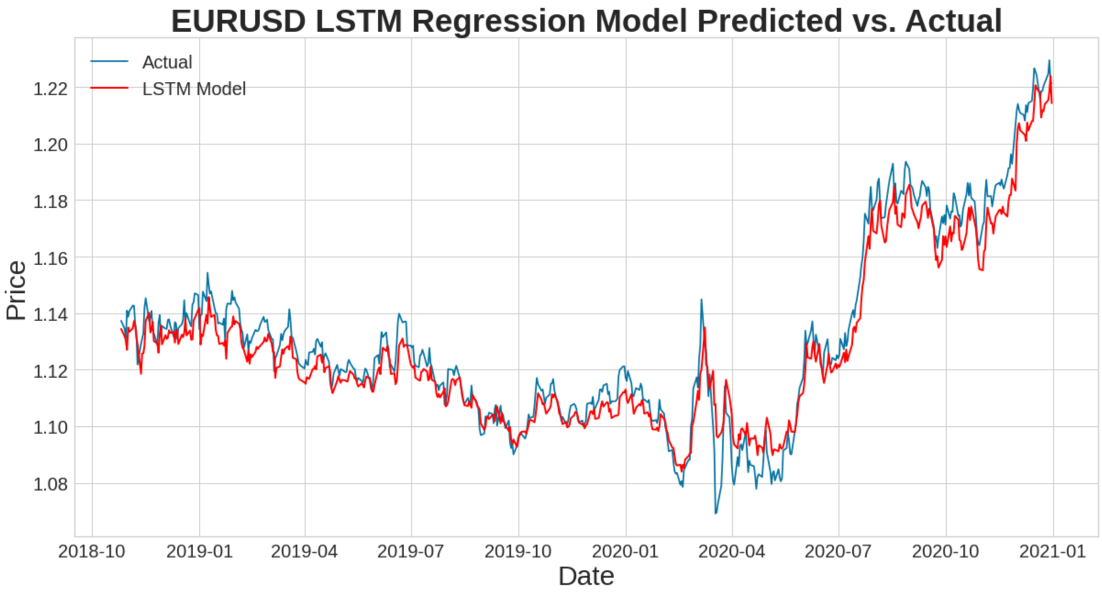


Figure 28. LSTM regression model prediction plotted against the test set of EUR/USD and both training and testing set.

4.5.4 Expanded Results for MLP Regression

The results displayed in Table 13 for the MLP regression model, Table 15 for the MLP classification model, and Table 18 for the LSTM regression model, indicate that the neural network fails to learn the price direction pattern using only one of the previous prices as input. As a result, the moving window was expanded to enclose 30 days for learning and predict one day into the future. This arrangement produces 2819 segmentation on the data set.

Data will be divided into 80% and 20% training and testing sets respectively. The grid search system was used on the MLP regression model, as it is 86% faster to train than LSTM. Time consumption is crucial because the system was constructed to have 3 hidden layers with a wider range of hyperparameters, resulting in 5700 different possible combinations for the MLP model. The selected hyperparameters by the grid search system are shown in Table 19. The model training and testing losses throughout the entirety of the iterations are plotted in Figure 29.

Table 19. Improved MLP model hyperparameters.

Hyperparameter	**Selected parameters for the regression MLP model**
Optimizer	SGD
Loss function	MeanSquaredError
Metric function	MeanAbsolutePercentageError
Batch size	1
Number of iterations	150 epochs
Input layer	Input size = 90 number of neurons = 90, activation function = tanh
1st hidden layer	Number of neurons = 100, activation function = tanh
2nd hidden layer	Number of neurons = 50, activation function = tanh
3rd hidden layer	Number of neurons = 100, activation function = tanh
Output layer	Output size = 1, number of neurons = 1, activation function = elu
Number of trainable parameters	27,541

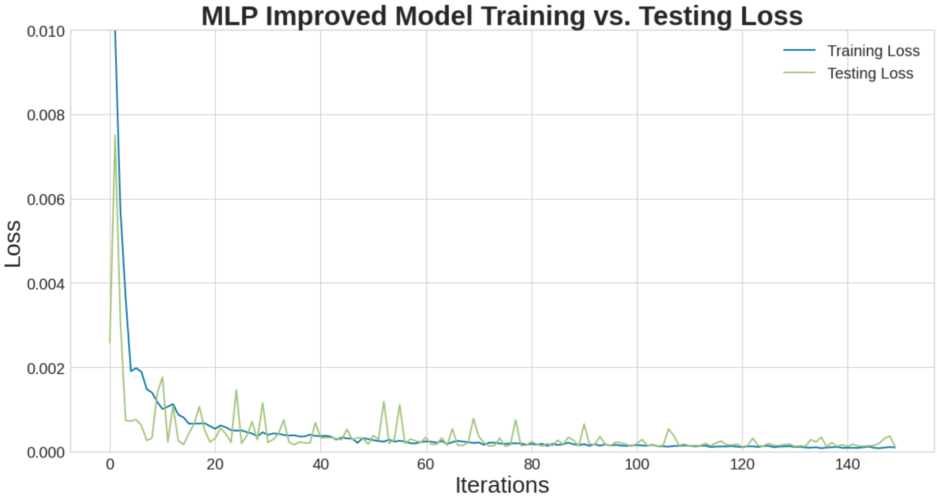


Figure 29. The improved MLP regression model training and testing loss plot, over the number of iterations.

The results of the improved model are shown in Table 20. The model produced an average R^2 score improvement of 57% over the simple linear regression model 10-fold CV sets, and a decrease of 7% when compared to the MLP regressor in Table 13. However, the improved model was 83% successful in identifying and predicting the next day's closing price direction. This is an improvement of 40% over most of the previous model's prediction accuracies, which, other than the SVR models, were shown to be random. The predicted output of the model is plotted against the actual value of the EUR/USD in Figure 30, showing the effectiveness of the model to accurately predict the future direction of the EUR/USD closing price.

Table 20. The improved MLP regression model RMSE, MAE, R^2, and accuracy results, including 10-fold CV R^2 scores.

	Test set	**Train set**
RMSE	0.011	0.008
MAE	0.007	0.005
R^2	0.867	0.995
Accuracy (%)	83.500	

10-fold CV R^2 scores

1	2	3	4	5	6	7	8	9	10	**Average R^2**
0.991	0.948	0.899	0.575	0.978	0.989	0.738	0.984	0.377	0.891	0.901

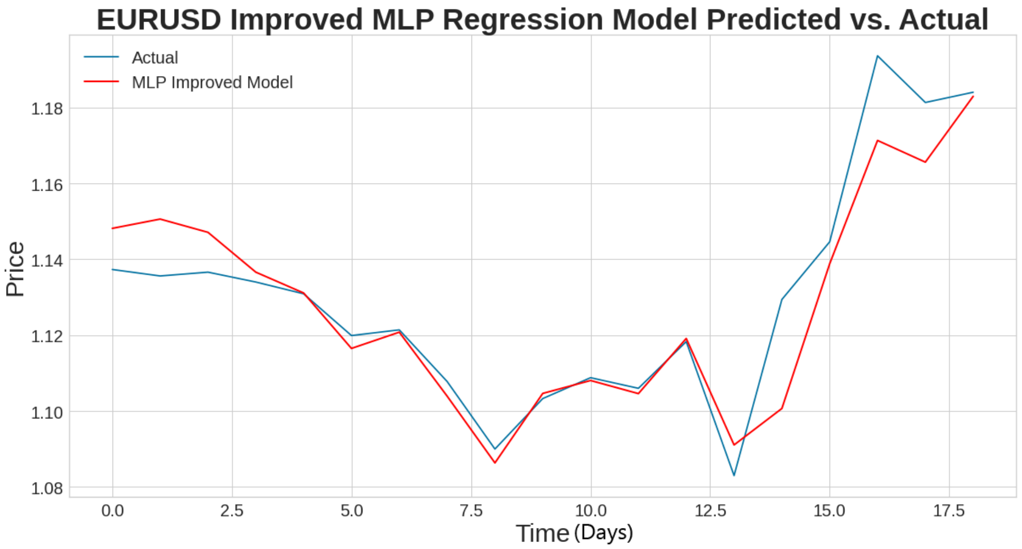


Figure 30. The improved MLP regressor predicted output plotted against the EUR/USD testing set.

4.6 Applying MLP to Crude Oil (WTI) and Natural Gas (NATGAS)

Similar to the previously constructed MLP classifier, the hyperparameters were selected using the grid search algorithm. The best result for crude oil (WTI) was seen using 60 days moving window in addition to using the RUB/USD as an input feature to the neural network. Natural gas (NATGAS) produced the highest accuracy using 30 days moving window. The hyperparameters of each model are seen in the following table. This resulted in the figures below showing the model training and testing losses throughout the entirety of the iterations for both trading assets.

Table 21. Hyperparameters of the MLP classifiers for NATGAS and WTI.

Hyperparameter	**Selected parameters for NATGAS MLP classifier model**	**Selected parameters for WTI MLP classifier model**
Optimizer	Adam	Adam
Loss function	SquaredHinge	SquaredHinge
Metric function	Accuracy	BinaryAccuracy
Batch size	32	32
Number of iterations	500 epochs	500 epochs
Input layer	Input size = 30 number of neurons = 30, activation function = linear	Input size = 120, number of neurons = 120, activation function = relu
1st hidden layer	Number of neurons = 37, activation function = tanh	Number of neurons = 150, activation function = linear
2nd hidden layer	Number of neurons = 22, activation function = tanh	Number of neurons = 90, activation function = linear
3rd hidden layer	Number of neurons = 15, activation function = tanh	Number of neurons = 60, activation function = linear
4th hidden layer	Number of neurons = 7, activation function = tanh	Number of neurons = 30, activation function = linear
Output layer	Output size = 1, number of neurons = 1, activation function = relu	Output size = 1, number of neurons = 1, activation function = sigmoid
Number of trainable parameters	3,378	53,581

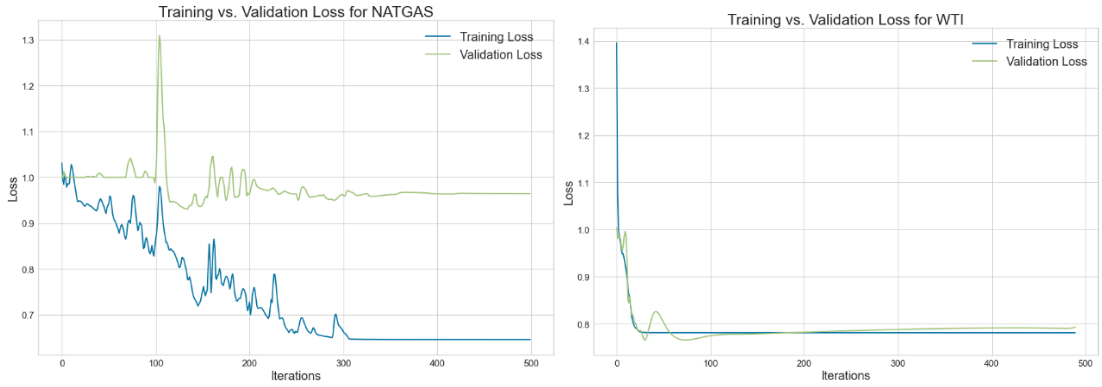


Figure 31. MLP classification model training and testing loss plot, over the number of iterations for NATGAS and WTI.

Based on the newly constructed models, the models produced results that exceeded 70% accuracy on both the training and testing sets. Such results can be seen in the following table and figure.

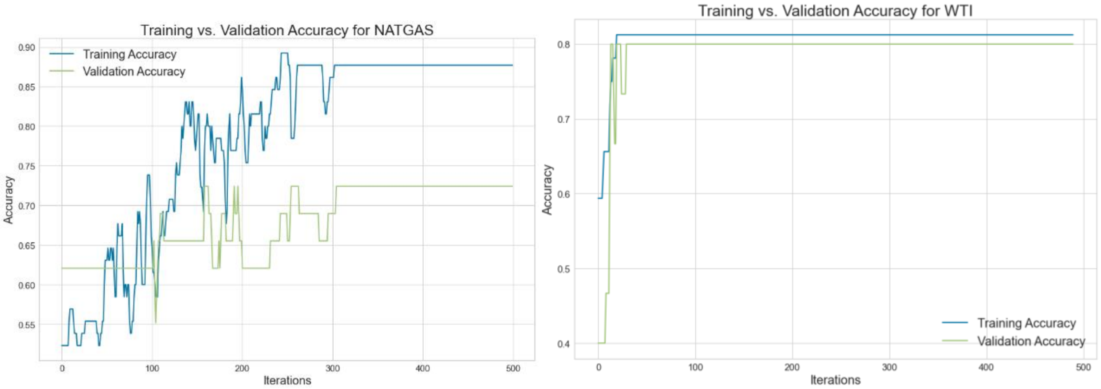


Figure 32. NATGAS and WTI MLP classifiers training vs. validation accuracies.

Table 22. NATGAS and WTI MLP classifiers confusion matrices.

NATGAS	**Test set**		**Train set**	
	Predicted False	Predicted True	Predicted False	Predicted True
Actual False	17	1	34	0
Actual True	7	4	8	32
WTI	**Test set**		**Train set**	
	Predicted False	Predicted True	Predicted False	Predicted True
Actual False	5	1	19	0
Actual True	2	7	6	7

Table 23. NATGAS and WTI MLP classifier training and testing set accuracies.

	NATGAS	**WTI**
Testset Accuracy (%)	72.41	80.00
Trainset Accuracy (%)	87.69	81.25

4.7 Results Comparison

Using the data collected during the study, a comparison has been drawn to exhibit the performance differences between the models. Table 24 shows the scoring of each model based on the testing set of the EUR/USD.

Table 24. Summary of the models used in the study. Showing the RMSE, MAE, R^2, Average CV $R^{2,}$ and accuracy.

Model	**RMSE**	**MAE**	**R^2**	**Average CV R^2**	**Accuracy (%)**
Simple Linear Regression	0.015	0.012	0.812	0.391	50.053
Multiple Linear Regression	0.009	0.007	0.932	0.542	49.982
SVR Linear Kernel	0.008	0.006	0.942	0.619	61.425
SVR RBF Kernel	0.016	0.008	0.879	0.511	56.406
MLP Regressor	0.008	0.007	0.945	0.970	52.404
LSTM Regressor	0.008	0.006	0.947	0.968	49.649
Improved MLP Regressor	**0.011**	**0.007**	**0.867**	**0.901**	**83.500**

The results show that using one day of previous closing prices, regardless of the number of features, leads to random future direction predictions. SVR models displayed potential, as both scored higher than any model, except the improved MLP. The significant performance of the improved MLP shows that the network failed to learn when it was given only one day, to interpret patterns. But when given 30 days of previous prices, the network was able to learn and produce considerably accurate future predictions for the price direction.

4.7.1 Time Complexity Analysis

Throughout this thesis, the one-day time frame was used to train, test, and predict the closing price/direction of an instrument for the next day. Time complexity is used to quantify the efficiency of an algorithm in terms of the time it takes for such an algorithm to produce an output. The following Table 25 was constructed based on the models used in the thesis, where it displays the time it takes each model to output a prediction.

Table 25. The time it took to produce an output for each model.

Model	Elapsed Time (Seconds)
Simple Linear Regression	0.0035
Multiple Linear Regression	0.0050
SVR Linear Kernel	0.0052
SVR RBF Kernel	0.0187
MLP Regressor	0.2062
Improved MLP Regressor	0.5074
LSTM Regressor	1.2691

As expected, the elapsed time increase as the model used to produce predictions becomes more complex. However, because the timeframe is selected to be relatively high (1-day), the elapsed time is rendered negligible as the prediction is made for the next 24-hours closing price. In the case of using lower timeframes (5-min, 1-min, 1-sec, etc...) the time complexity becomes crucial as such delays can render the model obsolete if it is unable to produce predictions promptly.

CHAPTER 5: CONCLUSION AND FUTURE WORK

The research aim of this thesis was to dive into the different methods of machine learning to achieve accurate future predictions of financial assets. The majority of the time was spent on collecting, exploring, and building a proper dataset, that included a variety of trading assets and economical reports. During the data exploration stage, it was found that most trading assets show correlation with each other, which was shown to be true when different assets were plotted against highly correlated counterparts, and shown to move accordingly and share price changes.

This study has started with establishing a baseline, which was done by modeling the EUR/USD price using various regression models. Starting with linear models that showed very low R^2 scores and failed to achieve accuracy levels higher than 50%. Furthermore, the research explored support vector regression models (SVR) with two kernels. Both linear and RBF kernels have proven to be candidates for further study, as the linear kernel scored an accuracy above 60%.

As the final stage of this study, the focus is particularly drawn on using ANNs. With both MLP and LSTM models being created, trained, and optimized. Choosing and refining the network's hyperparameters have proven to be a challenge, as small changes in some parameters can greatly affect the results of the model. The main problem faced during the study was the exceptionally long time needed to train LSTM networks, which resulted in worse models than MLPs, and this can be due to the limited parameters tested and explored. The use of 30 days of previous closing prices to predict one day into the future has proven successful. The MLP trained network has produced a future price trend prediction accuracy that exceeded 80%, which is a rate that was not seen before, in any of the models.

The study has shown that there is a possibility for ANNs and non-linear models to be used to predict financial asset prices. Future work will explore the outcome of using different assets, including more features and expanding the combination of hyperparameters. Lower prediction window time-frames are an important aspect to explore in the future, as the possibility of predicting the prices of the next 1 hour, 30 minutes, or 5 minutes might also be a viable option.

REFERENCES

[1] L. Yu, S. Wang and K. K. Lai, Foreign-exchange-rate forecasting with artificial neural networks, New York: Springer, 2011.

[2] S. Zhelev and D. R. Avresky, "Using LSTM neural network for time Series predictions in financial markets," *2019 IEEE 18th International Symposium on Network Computing and Applications (NCA),* 2019.

[3] A. Tsantekidis, N. Passalis, A. Tefas, J. Kanniainen, M. Gabbouj and A. Iosifidis, "Using deep learning to detect price change indications in financial markets," *2017 25th European Signal Processing Conference (EUSIPCO),* 2017.

[4] U. G. S. (USGS), "Mineral Commodity Summaries 2020," U.S. Geological Survey, Washington, 2020.

[5] M. El Alaoui, E. Bouri and N. Azoury, "The Determinants of the U.S. Consumer Sentiment: Linear and Nonlinear Models," *International Journal of Financial Studies*, 2020.

[6] D. Van den Poel, C. Chesterman, M. Koppen and M. Ballings, "Equity price Direction prediction for day trading: Ensemble classification using technical analysis indicators with interaction effects," *2016 IEEE Congress on Evolutionary Computation (CEC),* 2016.

[7] J. J. Murphy, Technical analysis of the financial markets: a comprehensive guide to trading methods and applications, Paramus, NJ: New York Institute of Finance, 1999.

[8] W. Lertyingyod and N. Benjamas, "Stock price trend prediction using artificial neural Network techniques: Case Study: THAILAND stock exchange," *2016 International Computer Science and Engineering Conference (ICSEC),* 2016.

[9] R. Meese and K. S. Rogoff, "Empirical exchange Rate models of the SEVENTIES : Are any fit to survive?," *International Finance Discussion Paper*, vol. 1981, no. 184, pp. 1-51, 1981.

[10] S. Tiwari, A. Bharadwaj and S. Gupta, "Stock price prediction using data analytics," *2017 International Conference on Advances in Computing, Communication and Control (ICAC3)*, 2017.

[11] R. Y. Nivetha and C. Dhaya, "Developing a prediction model for stock analysis," *2017 International Conference on Technical Advancements in Computers and Communications (ICTACC)*, 2017.

[12] Z. Yeze and W. Yiying, "Stock price prediction based on information entropy and artificial neural network," *2019 5th International Conference on Information Management (ICIM)*, 2019.

[13] E. Turkedjiev, M. Angelova and K. Busawon, "Validation of artificial neural network model for share price uk banking sector short-term trading," *2013 UKSim 15th International Conference on Computer Modelling and Simulation*, 2013.

[14] Z. Zhang, Y. Shen, G. Zhang, Y. Song and Y. Zhu, "Short-term prediction for opening price of stock market based On self-adapting Variant PSO-Elman neural network," *2017 8th IEEE International Conference on Software Engineering and Service Science (ICSESS)*, 2017.

[15] I. M. Kaya and M. E. Karsligil, "Stock price prediction using financial news articles," *2010 2nd IEEE International Conference on Information and Financial Engineering*, 2010.

[16] C. Wang and Q. Gao, "High and low Prices prediction of SOYBEAN futures with LSTM neural network," *2018 IEEE 9th International Conference on Software*

Engineering and Service Science (ICSESS), 2018.

[17] K. A. Manjula and P. Karthikeyan, "Gold price prediction using ensemble based machine learning techniques," *2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI),* 2019.

[18] A. Vohra, N. Pandey and S. Khatri, "Decision making support system for prediction of prices in agricultural commodity," *2019 Amity International Conference on Artificial Intelligence (AICAI),* 2019.

[19] M. R. Vargas, C. E. dos Anjos, L. G. Bichara and G. A. Evsukoff, "Deep Leaming for stock market prediction using technical indicators and financial news articles," *2018 International Joint Conference on Neural Networks (IJCNN),* 2018.

[20] Y. Wang and Y. Wang, "Using social media mining technology to assist in price prediction of stock market," *2016 IEEE International Conference on Big Data Analysis (ICBDA),* 2016.

[21] N. I. Nwulu, "A decision trees approach to oil price prediction," *2017 International Artificial Intelligence and Data Processing Symposium (IDAP),* 2017.

[22] K. Rathan, S. V. Sai and T. S. Manikanta, "Crypto-Currency price prediction using decision tree and regression techniques," *2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI),* 2019.

[23] P. K. Mahato and V. Attar, "Prediction of gold and silver stock price using ensemble models," *2014 International Conference on Advances in Engineering & Technology Research (ICAETR - 2014),* 2014.

[24] A. Samarawickrama and T. Fernando, "A recurrent neural network approach in predicting daily stock prices an application to the Sri LANKAN stock market,"

2017 IEEE International Conference on Industrial and Information Systems (ICIIS), 2017.

[25] S. Ravikumar and P. Saraf, "Prediction of stock prices using machine learning (regression, classification) algorithms," *2020 International Conference for Emerging Technology (INCET)*, 2020.

[26] M. A. Golberg and H. A. Cho, Introduction to regression analysis, Southampton: WIT Press, 2005.

[27] S. Chatterjee and J. S. Simonoff, Handbook of regression analysis, Hoboken, NJ: John Wiley & Sons, 2013.

[28] D. C. Montgomery, E. A. Peck and G. G. Vining, Introduction to linear regression analysis, Oxford: Wiley-Blackwell, 2013.

[29] J. Gong and S. Sun, "A new approach of stock price prediction based on logistic regression model," *2009 International Conference on New Trends in Information and Service Science,* 2009.

[30] D. Basak, S. Pal and D. Patranabis, "Support Vector Regression," *Neural Information Processing – Letters and Reviews,* vol. 11, 2007.

[31] T. Kleynhans, M. Montanaro, A. Gerace and C. Kanan, "Predicting Top-of-Atmosphere THERMAL RADIANCE USING Merra-2 atmospheric data with deep learning," *Remote Sensing,* vol. 9, no. 11, p. 1133, 2017.

[32] A. I. Galuškin, Neural networks theory, Berlin: Springer, 2010.

[33] A. Graves, Supervised sequence labelling with recurrent neural networks, Springer, 2014.

[34] M. Awad and R. Khanna, Efficient Learning Machines Theories, Concepts, and Applications for Engineers and System Designers, Berkeley, CA: Apress, 2015.

[35] J. Duchi, E. Hazan and Y. Singer, "Adaptive Subgradient Methods for Online Learning and Stochastic Optimization," *Journal of Machine Learning Research,* no. 12, pp. 2121-2159, 2011.

[36] D. . P. Kingma and B. . L. Jimmy, "ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION," *ICLR 2015,* 2015.

[37] B. Jeevan, E. Naresh, P. B. kumar and P. Kambli, "Share price prediction using machine learning technique," *2018 3rd International Conference on Circuits, Control, Communication and Computing (I4C),* 2018.

[38] A. K. Sirohi, P. K. Mahato and A. Vahida, "Multiple kernel learning for stock price direction prediction," *2014 International Conference on Advances in Engineering & Technology Research (ICAETR - 2014),* 2014.