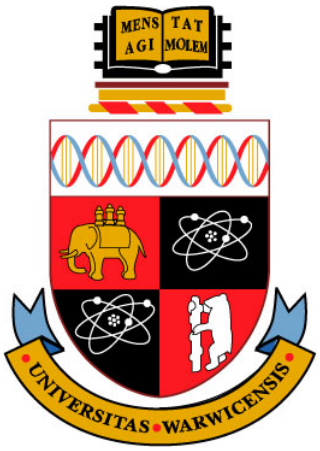


An exploration of forex trend prediction using machine learning techniques

CS310 Third Year Project

Hasanain Ghafoor

1820962

Supervisors: Nasir Rajpoot & Shan Raza

Department of Computer Science

University of Warwick

2021

Abstract

This project explores a range of approaches for Forex trend prediction in which bullish and bearish regions are forecasted in a binary classification problem. Both data preprocessing and Exploratory Data Analysis (EDA) steps are reviewed for financial time series and strategies within the current literature are criticized. Moreover, a novel approach to feature generation is proposed as well as an assessment of machine learning techniques for multivariate time series prediction. Furthermore, a range of classifiers are implemented and tested on historic foreign exchange data in which an evaluation of the models used, metrics employed and hyperparameters optimized is provided. A discussion on the limitations and areas of potential exploration within the research domain is also detailed.

Keywords: financial market prediction, forex, sentiment analysis, machine learning, data analysis, EDA, model optimization, ensemble models.

Acknowledgements

I would like to thank both of my supervisors Nasir Rajpoot and Shan Raza for providing me with consistent support despite the unpredictable conditions surrounding the project. Without their rapid feedback and challenging suggestions, both the breadth and depth of this work would not be as expansive as it is now. I would also like to thank my family and friends for supporting me throughout my studies. More specifically, I would like to thank my father for teaching me the importance of perseverance and for pushing me towards a career within computer science.

Contents

1 Introduction

The Forex market is the biggest and most liquid market in the world [1], this provides an incentive for the creation of algorithms to benefit from its volatility. With the ever-increasing use of machine learning within financial markets [2], the question stands whether prices and trends within the forex market can be predicted by machine learning models.

Within the forex market, currency pairs are traded with the hope of benefitting from the change in the exchange rate, relative to the position being held. Traders tend to make informed trades following substantial analysis. Traditionally, this comes in the form of technical, fundamental and sentiment analysis. Fundamental and sentiment analysis follow the nuances of the human psyche as well as the current news. Technical analysis, however, utilises statistical methods to produce indicators that help identify patterns within historic price action to predict future movements. The work of Oliveira et al [3] and Sespajayadi et al [4] motivate the use of such techniques for modelling financial markets and suggest areas for further exploration. Consequently, this project outlines a nuanced approach to all three forms of analysis for feature generation.

Moreover, this research comprehensively evaluates various approaches to forex trend prediction and ascertains the place of machine learning models within the domain. Here, a range of classical machine learning classifiers are employed and are fed engineered feature vectors to identify bullish and bearish regions within historical datasets. This follows from the disparity between results within the current research for the forex binary classification problem [5].

Furthermore, a holistic approach to research is taken in which areas of both classical data analytics and the supervised machine learning stack are explored. This includes the critical assessment of processes such as data labelling, transformation, feature selection and the evaluation metrics imposed on models.

2 Background

As this project combines both financial markets and computer science, niched areas from both domains are covered. This section serves to provide the pre-requisite knowledge to follow this work.

2.1 The Forex Market

The Forex market is a global marketplace in which currencies are exchanged in an over-the-counter fashion. With an average daily trading volume of over $5 trillion, the Forex market is easily the most liquid market in the world [1]. This provides an incentive for retail traders to buy and sell currencies in the hopes realizing profits.

Figure 1: Forex Prices Breakdown

Currencies are quoted on exchanges in pairs, such as GBP/USD or USD/EUR. This is because you are simultaneously buying one currency and selling the other during a transaction. Within a currency quote, as shown in Figure 1, there are two values, namely the base currency and the quote currency. The quoted price for a currency pair represents how much of the quote currency is needed to acquire one unit of the base currency.

Note that there are two quoted prices, the bid and the ask price. The bid price is the amount the broker is willing to buy a currency for whereas the ask price is how much they are selling for, the difference between the two is the spread. These terminologies are

commonplace within forex trading and are fundamental to understand how to execute trades and profit from the market.

2.1.1 Forex Data

Forex data typically comes in an (Open High Low Close) OHLC format in which columns of each observation correspond to Open, High, Low and Close prices. Intuitively, the Open and Close prices refer to the price when the market opens and closes. Meanwhile, the High and Low prices refer to the maximum and minimum price for a currency pair for a given time granularity. Moreover, Forex data forms a time series, as shown in Table 1, in which a temporal order must be retained.

Date	Open	High	Low	Close
28.02.2011	1.01867	1.02028	1.01257	1.01351
01.03.2011	1.0135	1.01823	1.00864	1.01708
02.03.2011	1.01721	1.01886	1.01267	1.01467
03.03.2011	1.01469	1.01511	1.00778	1.01419
04.03.2011	1.01419	1.01419	1.01419	1.01419
...	...	...	...	...

Table 1: Sample OHLC Forex Data

Figure 2 graphs the EUR/USD pair and demonstrates how Forex data is visualized for retail traders to interpret on a day-to-day basis. Figure 3 shows an alternative visualization known as a candlestick chart. Candlestick charts clearly display OHLC prices and provide a more transparent view of the Forex time series. This way, interesting analysis can be performed instantly on the data such as viewing the height of candlesticks as a volatility measure of buying/selling pressure.

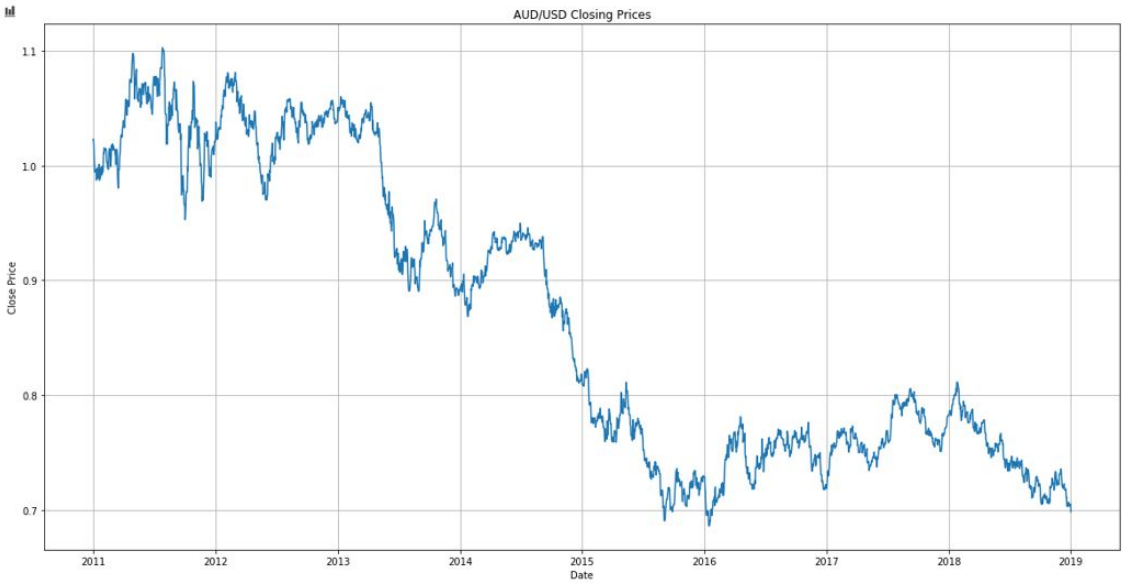


Figure 2: Line plot of AUD/USD Closing Prices

Figure 3: Sample Candlestick Chart

2.2 Technical Analysis

Within most Forex trading strategies, Technical Analysis is utilized to analyze historic price action and identify patterns or trends. This analysis often comes in the form of Technical Indicators that are holistically interpreted to make informed decisions [6]. The simplest example of a Technical Indicator is the moving average. This indicator applies a smoothing to the price series and can often remove unwanted noise within

the data, thus illustrating underlying trends. Figure 4 demonstrates 15-day and 30-day Simple Moving Averages (SMA) for sample OHLC data. The calculation for SMA is as follows:

$$SMA = \frac{P_{i-1} + P_{i-2} + \ldots + P_{i-n}}{n} \tag{2.1}$$

where, P_i is the price of the current observation and n is the period size.

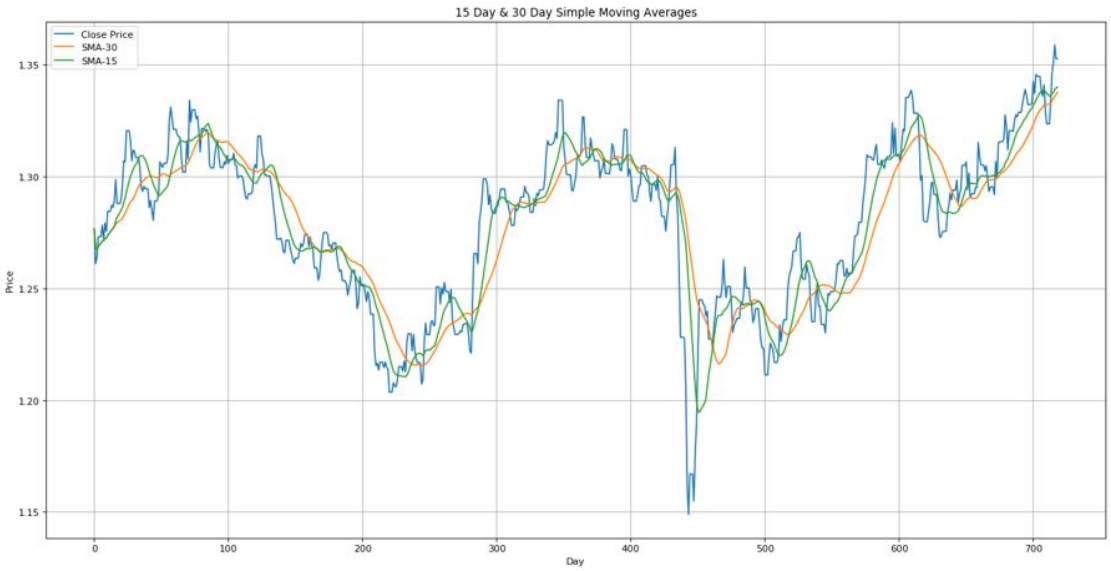


Figure 4: Simple Moving Averages

Technical Analysis spans much further than moving averages, each indicator performing bespoke calculations and returning various types. Nevertheless, a common parameter amongst most technical indicators is the "period". The "period" defines the number of observations used to generate a single value for an indicator, synonymous to a window size. Such parameters can be tuned to attain different technical indicator outputs and optimize the generated features. Thus, it is important to consider the parameters used when performing Technical Analysis.

Although technical indicators provide valuable insights into the trends within historic price action, they each have their own limitations. A limitation of the SMA indicator is its lagging nature. This is because calculations are based on n-periods of previous data and so trends are often realized late. Figure 4 illustrates this as the 15-day and

30-day SMA's converge slightly after the dramatic movement of price. Moreover, it is important to understand the limitations of each technical indicator to be able to correctly apply them and prevent naïve analysis.

2.3 Sentiment Analysis

Social media has rapidly moved to the forefront of society as over 3.6 billion people worldwide consumed its content in 2020 [7]. This has opened the potential for data mining and analytics to better understand the behavior of the population. Sentiment Analysis capitalizes on the growth of social media and data analytics by focusing on the behaviors of retail traders and their feelings towards financial assets.

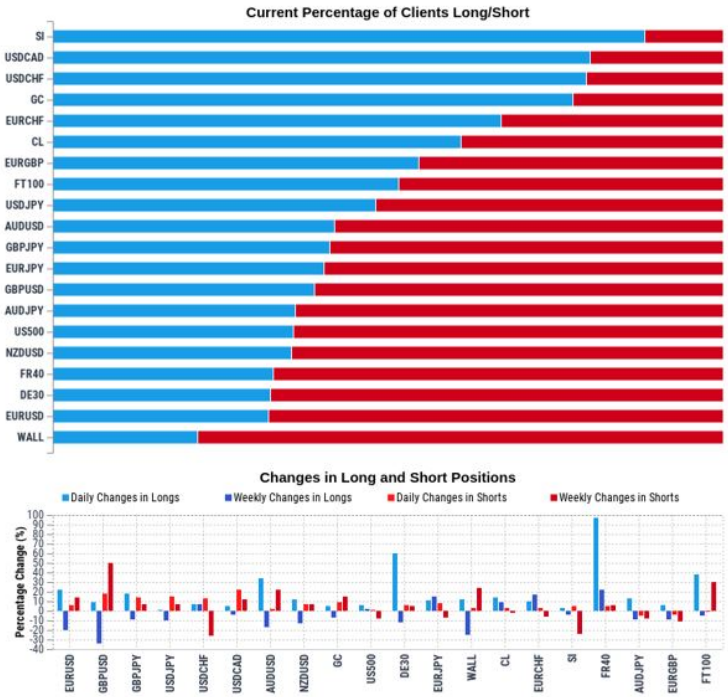


Figure 5: Daily-FX Long-Short Ratio Graph from Sentiment Report [8]

Within financial markets, sentiment data often comes in the form of social media content, the most popular of which being Twitter. The intuition behind Sentiment Analysis is to leverage the opinions of specific demographics to predict trends in the markets they stake in. Intuitively, if most retail traders express the same opinion about

a currency pair, they are more likely to hold a position relative to that opinion.

Other sentiment data can be found in the form of long-short ratios in which large groups or enterprises are able to visualize the trading positions held by their users. Figure 5 shows a long-short ratio graph where the positions held for a range of currency pairs are visualized. Datasets of this nature are relatively difficult to acquire due to their lucrative insights and so are often not released as raw data.

2.4 Fundamental Analysis

The most obvious form of analysis within financial market forecasting pertains to the synthesis of financial factors and economic indicators. Fundamental Analysis embodies this process in which the economic, social, and political forces that may influence exchange rates are analyzed.

Governments often release reports in which the economic performance of a country is explored, producing economic indicators. A fundamental economic indicator is Gross Domestic Product (GDP) which reflects the total market value of a country's services and goods. Furthermore, by determining the health of a countries economic power, the strength of their currency can be predicted.

Other than the raw analysis of a countries economic health, news headlines and articles can also be analyzed to determine a countries sociopolitical condition. This follows from sentiment analysis as news articles become more readily accessible online and through social media.

2.5 Algorithmic Trading

Classical approaches to forex trading involve the use of Technical Analysis within strategies that utilize rigid rule sets involving a set of indicators. Applying hard coded rules to technical indicators to place orders on the forex market is called Algorithmic Trading. Such strategies often produce signals to indicate to retail investors whether to buy,

sell or hold a particular currency pair. This type of analysis is a form of manual trend prediction as the predictive model is unable to learn.

An example of an algorithmic trading strategy is the SMA Crossover. This involves two moving averages, one short term and one longer term average. If the short term SMA crosses above the longer term SMA, an uptrend signal is generated. Otherwise, if the short term SMA crosses below the longer term SMA, a downtrend signal is generated. Figure 4 demonstrates this concept for a 15-day and 30-day SMA's on sample Forex data. Moreover, algorithms are often employed by Expert Advisors to attain automation, this involves converting strategies into programs and passing them to proprietary bots to run. These bots can push signals to traders or integrate into trading platforms to enforce automated trading.

Apart from technical indicators, more complex algorithmic trading strategies may leverage fundamental and sentiment indicators within their predictions. Similarly, advanced statistical models such as Autoregressive Integrated Moving Average (ARIMA) may be used to conduct further financial time series analysis. This model utilizes lagged series as predictors within a linear regression model in which predictions depend on lagged prices and forecast errors. Furthermore, algorithmic trading utilizes the benefits of programming and statistics to create predictive models for financial markets.

2.6 Machine Learning

Machine learning has been applied to many contemporary problems as it attempts to incorporate autonomy through learning across a multitude of industries. This science can be categorized into supervised and unsupervised machine learning to differentiate the types of problems it attempts to solve. Financial market applications of machine learning often refer to the supervised category in which classification or regression tasks are approached. Regression pertains to the prediction of continuous variables whereas classification deals with discrete variables.

Within classification problems, feature vectors are passed to classifiers as part of feature matrices in which output labels are predicted. These classifiers train on sets of pre-labelled training data that contain associated ground truth labels for each observation. After the classifiers are trained, they are introduced to new unlabeled test data and attempt to predict the appropriate output labels for each feature vector. The notation for a k-dimensional input feature vector X and corresponding row vector of ground truth labels y is as follows:

$$X = \begin{bmatrix} x_{11} & x_{12} & ... & x_{1k} \\ x_{21} & x_{22} & ... & x_{2k} \\ ... & ... & & ... \\ x_{n1} & x_{n2} & ... & x_{nk} \end{bmatrix}_{n*k}, Y = \begin{bmatrix} y_1 \\ y_2 \\ ... \\ y_n \end{bmatrix}_{n*1} \tag{2.2}$$

For machine learning models to learn, the Independent and Identically Distributed (IID) assumption is made in which training and test data are assumed to:

- Originate from the same data generating distribution.
- Be sampled independently, such that order does not matter.

The IDD assumption makes learning possible for many problems as training samples are likely to resemble test examples, however, this project deals with time series data. Time series data is temporally dependent and contains autocorrelations which means it no longer satisfies the IID assumption. This presents a difficult research problem as classical approaches to processes such as model validation cannot be used due to the biases introduced by time dependance. Thus, data must be carefully manipulated to ensure correct analysis and innovative methods are employed to achieve success.

2.7 Literature Review

Financial market prediction has been researched extensively within the past two decades and continues to be explored due to its potential applications and lucrative nature.

Within the domain, a range of techniques and approaches have been well documented and can be classified into machine learning and non-machine learning techniques. Moreover, research has been further divided into the type of problem they attempt to model, which can be categorised into classification, regression, and other approaches. The following literature review provides a critical analysis of the current literature and highlights research areas from which innovation can be sought.

It is important to note that the literature presents approaches for a range of financial markets, such as Forex and Stocks, therefore some methods may not be universally applicable due to the nuances of individual markets.

2.7.1 Price Prediction - Regression

Price prediction refers to the forecasting of real price values within financial markets, such as stock values or closing prices of exchange rates. Here, a regression problem is modelled in which a range of input features are used to predict a continuous output variable. The work of Sidehabi et al in [9] demonstrates this by comparing the effectiveness of supervised machine learning models against statistical models for predicting future exchange rates. Specifically, the study found that deep learning models such as Genetic Algorithm Neural Networks (GA-NN) outperformed classic statistical methods when predicting longer periods. The intuition is that genetic algorithms can accelerate the learning of their underlying neural networks by leveraging their stochastic nature. However, methods such as Adaptive Spline Threshold Autoregression (ASTAR), a non-linear time series model that utilises lagged variables, performed better on short term predictions. Nonetheless, Sespajayadi et al [4] and Wang [10] support the idea of using hybrid models such as GA-NN's when predicting financial markets by predicting shorter granularities and attaining low RMSE values.

An alternative approach to exchange rate prediction is demonstrated by Pradeepkumar and Ravi [11] in which a Chaos and Quantile Regression Random Forest model is adopted. Here, Chaos theory is used to convert price data into phase space using lags

and embedding dimensions for feature extraction. Firstly, Saida's method is utilized to detect chaos within a historic price series. Cao's method then establishes the minimum embedding dimension for the series and Akaike Information Criterion (AIC) is used to find the optimum lag. Results showed that the proposed method outperformed alternative chaos-based approaches and supports the notion of employing hybrid solutions for Forex prediction.

Other approaches to exchange rate prediction are detailed by Islam et al [2] in which a holistic view of the current literature is provided. The most popular method for regression was found to be Support Vector Regression (SVR) models. An example of this can be seen by Li and Suohai in [12] where an SVR is used with an RBF kernel to predict next-day prices. It was found that applying an optimization technique such as genetic, particle swarm or artificial fish swarm algorithms performed better than the default model. Moreover, the research supports the employment of a hybrid approach when predicting prices and concluded that parameter optimizations are critical when improving prediction accuracy.

Furthermore, it can be observed that both neural networks and SVR models are the most applied techniques in the literature [2]. Shiao et al [13] provide an analysis of both approaches by comparing an LSTM to an SVR model, discovering that the recurrent neural network was able to predict longer periods more efficiently. Zhelev and Avresky [14] support these results by analysing the performance of an LSTM model when using different data granularities as features and yielding positive results. Similarly, the work of Lakshminarayanan and McCrae in [15] provides a comparative study of these methods by implementing both an SVR and LSTM model for stock market prediction. It was concluded that the LSTM model outperformed SVR in all environments, especially when using moving averages. Despite these findings, SVR's continue to be employed due to their affordances in computational requirements as well as performing well under high dimensional data [12] [16].

Weerathunga & Silva [17] illustrate a more robust hybrid approach by implementing a DRNN to predict next-day exchange rates. In conjunction, an ARIMA model is enforced to forecast deviations between predictions and estimated values. Final predictions were then obtained by combining the two forecasts. Here, the DRNN was utilized to overcome the inherent non-linearity and non-stationarity of Forex data whilst the ARIMA model made more reliable forecasts. Furthermore, the approach was determined to outperform standalone models by combining machine learning and time series analysis to build an architecture for prediction. This is contrary to the approaches adopted by Eng et al [18] and Zhelev & Avresky [14] in which an ANN and LSTM model are respectively used as black boxes for prediction yet achieve similar prediction accuracies. An explanation for these results is illustrated by Shiao et al [13] in which the use of RMSE and similar evaluation metrics within the Forex domain are criticised. This is due to the small price deviations within the Forex market, demonstrated in Table 2, making it hard to interpret model performance through standard regression evaluation metrics. Moreover, Li & Suohai [12] highlight how using a single error indicator such as RMSE is not comprehensive enough to evaluate model performance and so propose a range of new metrics. These metrics include absolute RMSE, total error, relative error, and correct trend rate (CTR). Specifically, CTR uses next day trend labels to observe how effectively the models predict market movements. The study showed that even when RMSE and total errors were low (8.3 and 0.0478 respectively) for SVR models, the CTR was 0.37 (37%). Furthermore, the research provides evidence as to why more exhaustive evaluation metrics are necessary considering much of the literature employs the same naïve metrics [4], [9], [13], [18].

Date	Open	High	Low	Close
28.02.2011	1.01867	1.02028	1.01257	1.01351
01.03.2011	1.0135	1.01823	1.00864	1.01708
02.03.2011	1.01721	1.01886	1.01267	1.01467
03.03.2011	1.01469	1.01511	1.00778	1.01419
04.03.2011	1.01419	1.01419	1.01419	1.01419
...	...	...	...	...

Table 2: Sample AUD/USD OHLC Exchange Rate Data

Another limitation of the current approaches is presented by Weerathunga & Silva [17], where it is commented that the nature of the market makes it difficult to determine what is significant from noise. This follows from the problem of attempting to disprove the Random Walk and Efficient Market hypothesis (EMH). The Random Walk hypothesis states that the stock market moves in relation to a random walk and is unpredictable, whereas the EMH states that prices reflect all relevant data. Modelling financial prediction in its raw form of price forecasting directly challenges these concepts, making the problem inherently more arduous. Therefore, alternative forms of prediction and modelling must be approached to gain more ground within the research domain [2].

Overall, a range of models and techniques have been applied to price prediction within financial markets, however, there remains space for exploration. Islam et al [2] discuss how, for the regression problem, models such as lasso regression and multivariate regression are profoundly unexplored. Similarly, the literature fails to provide thorough documentation regarding data analysis, labelling and pre-processing. Chion et al [19] motivate this by highlighting the significance of pre-processing steps for the prediction accuracy of models. Furthermore, a holistic approach to Forex prediction which encapsulates the entire data analysis stack is non-existent within the current literature and would prove beneficial to the research domain.

2.7.2 Trend Prediction - Classification

Trend prediction within financial markets refers to the forecasting of future trends/-movements of an asset's value or price. This form of prediction models a classification problem in which input variables are utilised to determine a discrete output label. Like in the regression problem, Islam et al [2] suggest that the most common approaches to trend prediction are neural networks or Support Vector Machine's (SVM). The former has been shown to be more efficient due to their memory attributes, however, the complexity and computational demand of multi-layered neural networks make them difficult to reasonably explore [15]. Nevertheless, SVM's continue to be the most cited models,

most utilised in conjunction with genetic algorithms [2]. These operate on a population of individuals and enrol a heuristic function to select a subset of features from which an offspring can be generated from. The offspring are generated from values within their parent features' domains and undergo random mutations which can be beneficial. Such a stochastic algorithm can help improve the efficacy of machine learning models when the correct feature subsets are selected [20].

The work of Baasher & Fakhr [21] explores the forex trend prediction problem in which the next day trends of daily OHLC data are predicted. Here, an SVM was compared to a Multi-Layer Perceptron (MLP) over a range of feature selection and extraction techniques. The SVM model attained a maximum classification accuracy of 78%, outperforming the MLP. These results contradict previous research as a single, un-optimized model was able to achieve relatively high accuracies. Nguyen et al [22] presented similar findings when evaluating an SVM for forex trend prediction. After manual testing, a polynomial kernel was shown to outperform an RBF kernel in terms of classification accuracy and after a financial backtest. The financial backtest involved running the classifier on historic data and calculating financial statistics when executing orders according to the models' predictions. This form of evaluation is uncommon within the literature, which is profound considering the nature of the research. Furthermore, the study was able to better evaluate the financial viability of its models through backtesting analysis. For example, it was found that increasing the number of input features from 137 to 196 failed to increase the SVM's accuracy, however, it significantly reduced the maximum drawdown of the strategy when backtesting, making it more financially sound.

Another approach to trend prediction is presented by Lee et al [23] in which a Hidden Markov model (HMM) was used to forecast next day Forex trends and achieved a classification accuracy of 95%. The HMM was trained using a Baum Welch algorithm and prediction was calculated through a feed-forward algorithm. Accuracy of this magnitude can be explained by the proprietary Linear Regression Line (LRL) function

that was used to segment input data into up/downtrend labels. Such a process suggests that the model would not hold the same results when enforced live on the Forex market or even on a financial backtest. Moreover, comparing the efficacy of models becomes even more difficult as classic performance metrics do not reflect the pre-processing steps and assumptions made before prediction. This highlights the importance of techniques such as backtesting in [22] as more transparent indicators are required due to the disparity in results across the literature.

Hybrid approaches have also been found to be beneficial within trend prediction. For example, Jubert et al [24] employ a genetic algorithm with an SVM to optimize investment solutions and trade with appropriate levels of leverage. Using a genetic algorithm allowed for flexibility when backtesting solutions by adjusting the leverage used and tuning feature weights. Again, it was shown that this hybrid approach outperformed standalone models. The work of Chiong et al [19] follows a similar approach but employs a particle swarm optimization algorithm along with an SVM. It was concluded that the combined method performed better than alternative deep learning networks for stock trend prediction. Finally, Khan et al [25] present a holistic approach for forex trend prediction by employing a range of data analysis techniques and evaluating standard classifiers. The research found that the Random Forest tree-based classifier provided consistent results across all cases and was superior to the likes of SVMs, K-Nearest Neighbours and MLP's. Further testing revealed that an ensemble model, consisting of a voting classifier that aggregated independent classifiers, performed the best. Furthermore, this shows the undeniability of hybrid approaches effect on model performance.

It is important to note that the trend prediction problem generates two archetypal classification methods, namely binary classification, and multi-class classification. In binary classification, as presented by Baasher & Fakhr [21], labels are either generated as uptrends or downtrends. This naïve method for binary labelling is called "Fixed Horizon Labelling" as labels depend on the price change some constant time step into the

future. Tang & Chen [26] implement a solution for binary classification by combining an LSTM and Convolutional Neural Network (CNN) for stock market trend prediction. The model attained a 54% classification accuracy, beating a range of baseline machine learning models. Objectively, such a result is difficult to differentiate from a random walk or coin flipping method, however, Gerlein et al [5] provide an explanation for this. In the study, various low complexity machine learning classifiers were used to predict exchange rate trends for a range of currencies. The models attained modest classification accuracies between 50-58%, however, when financially backtested, the same models proved advantageous. For example, a Naïve Bayes classifier achieved a mere 52.99% classification accuracy yet attained a maximum cumulative return of 142.89% when backtested. Moreover, this research highlights a potential limitation of binary classification as financial markets exhibit non-trending regions as well as up/downtrends. The introduction of a third-class label allows for the measure of real up/down trend classifications as well as non-trends.

The work of Jubert et al [24] demonstrates the multi-class classification problem by using an SVM model to classify Forex OHLC data into up, down, and non-trending regions. Through this approach, the model was able to attain an average accuracy of 85.64% when employing price sequence analysis. As predicted, these results show great improvement compared to those found in the earlier binary classification studies. The research, however, notes that the non-trending (sideways) regions were the hardest to classify and exhibited poor results, which contradicts previous conclusions. Therefore, the binary label case presents a simpler abstraction of the problem by enforcing rigid assumptions. Baasher & Fakhr [21] support this idea and illustrate how it is more important to correctly classify strong trends rather than intricate market movements. Moreover, allowing for simplicity becomes crucial within financial market prediction as the steps taken in data pre-processing heavily influence the predictive power of models.

As well as the type of labels being predicted, the labelling methods implemented also influence model results. The "Fixed Horizon Method" was explored in [21] and

is commonplace amongst the current literature. A limitation of this approach is that trends are defined as the difference in price between two observations in a time series. This means that deviations between these two points are unaccounted for. Prado [27] proposes an alternative data labelling method called the "Triple Barrier Method", which provides a solution for the problems associated with Fixed Horizon labelling. Despite this, data labelling approaches are relatively undocumented in the current research and so a nuanced area for potential exploration is provided. Similarly, Islam et al [2] state that SVM's have not been fully explored within the literature when used in conjunction with neural networks. They also convey how Natural Language Processing as a concept for financial text analysis is completely unused. Finally, although LSTM's have been implemented [26], they are relatively underutilised for trend prediction despite their memory abilities being very beneficial for time series analysis.

2.7.3 Feature Generation

A crucial element within financial market forecasting using machine learning is feature generation. This is where features are crafted, typically from existing input variables, to generate new information and abstractions about a dataset. The approaches to feature generation are ubiquitous across both the stock and forex market, the archetypal method being Technical Analysis. This technique utilises statistical methods to produce indicators that identify patterns within historic price action to help predict future price movements [6].

Baasher & Fakhr [21] demonstrate how technical analysis is used to construct technical indicators as inputs into machine learning models. Over 81 features were generated from leveraging technical analysis including Moving Average Convergence Divergence (MACD), Commodity Channel Index (CCI) and Williams Accumulation Distribution Line (WADL). These indicators are commonplace in the literature and are used identically in [4], [5]. Considering technical analysis is used universally in the same way, more research can be conducted into different forms of technical analysis or using alternative indicators. The work of Gerlein et al [5] touches on this when proposing the use of

a Restricted Boltzmann Machine for extracting features from technical indicators in future work. The intuition behind this is to discover new and advantageous features from redundant indicators.

Jubert et al [24] build more of an abstract approach to technical analysis by enforcing a genetic algorithm to adjust the weights of each indicator, allowing for important indicators to take precedence during prediction. Despite this, it was found that using price sequences as input features yielded far superior results than technical indicators, indicating that technical analysis should be used as a supplementary method. Moreover, it is concluded that the applications of technical analysis for feature generation are rigid within the literature and there exists room for innovation.

Another primary form of feature generation is Sentiment Analysis. This form of analysis follows the nuances of the human psyche and retail trader sentiment to generate features, typically through the processing of online social media data. Various methods exist for extracting sentiment scores from bodies of text; however, Natural Language Processing underpins most approaches.

The impact of social media, specifically Twitter sentiment, on indexes is illustrated by Rao & Srivastava [28] in which tech-savvy influencer accounts are shown to hold predictive power over the markets. Similarly, in the research of Tang & Chen [26], sentiment data is derived from web scraped Reddit posts and used to predict trends of the Dow Jones Industrial Average (DIJA). In this study, the top 25 posts from the last 24 hours were scraped from a range of subreddits. Results showed that the models that leveraged social media data outperformed those that did not, further illustrating the effectiveness of Sentiment Analysis. On the other hand, the work of Oliveira et al [3] suggests that trader sentiment does not correlate with stock returns. Here, social media data from Twitter was used to generate two indicators, namely derived sentiment and posting volume. Although trader sentiment was not shown to correlate with stock returns, posting volume displayed promising results. A reason for this is described in

the paper in which the model's ability to recognise financial jargon within social media posts is criticized. This indicates that the tuning of sentiment models is critical as domain-specific syntax determines the derived scores of individual posts.

The final fundamental approach to feature generation within the literature follows from Sentiment Analysis and is called Fundamental Analysis. This technique differs from Sentiment Analysis in that it focuses on the processing of financial and economic factors surrounding a security or currency pair [29]. These factors can often be found in economic calendars or, more generally, in news items and articles.

Like in the previous analysis, NLP models are typically utilized to derive quantitative scores from fundamental data. Cheng et al [30] proposes a different method for classifying sentiment within news headlines by fusing semantic information into vectors. Scores are then derived from the vectors by associating syntax weights and passing them through a model. It was determined that this approach outperformed classic machine learning classifiers.

Hybrid approaches for feature generation are often employed in which both Fundamental and Sentiment analysis are used concurrently. Khan et al [25] demonstrate this when analysing online Twitter data and New headlines to predict trends in the stock market. The paper found that models that utilised this alternative data performed better than those that used technical analysis or price sequences alone. Moreover, it was concluded that social media analysis and news headline processing increased the average classification accuracy of models. Chiong et al [19] obtain similar results for an identical problem and found that models using Fundamental & Sentiment analysis outperformed baseline neural networks. The work of Khan et al [25], however, highlights the limiting factor of these processes being the ability of models to recognize domain-specific syntax. This supports the discussions of Oliveira et al [3] and highlights the importance of thorough model tuning. The use of a wider range of social media is also proposed as future work to allow for a comprehensive analysis of online sentiment.

Similarly, the work of Chantarakasemchit et al [31] comments on the lack of available fundamental data, specifically financial indicators, for higher data frequencies. Here, 372000 records of minute-by-minute Forex OHLC was used in which only 12-16 corresponding records existed for each financial indicator. This meant that linear interpolation was performed to fill missing values, resulting in poorer data. Patil et al [32] exhibited similar problems with fundamental data and suggested more could be done to acquire richer datasets.

It is important to note that the overwhelming amount of research into Sentiment & Fundamental Analysis for financial market prediction pertains to the Stock market alone. Islam et al [2] state that only 6% of all research within Forex trend prediction involves any form of Sentiment Analysis. Furthermore, there exists room for the exploration of feature generation techniques in Forex market prediction.

2.7.4 Optimization Techniques

Optimization techniques often refer to the tuning of parameters within models in hopes of incrementally improving prediction accuracies. Genetic Algorithms tend to be the most popular optimization technique in the literature, typically as part of hybrid machine learning pipelines. Jubert et al [24] demonstrate the use of a genetic algorithm with an SVM for hyperparameter tuning and found it to outperform unoptimized models. Sidehabi et all [9] and Sespajayadiet et al [4] follow similar approaches and attain promising results when combining genetic algorithms with neural networks.

Alternatively, the work of Li & Suohai [12] compares a range of optimization techniques such as genetic, particle swarm and artificial fish swarm algorithms. These were used to tune parameters within an SVR model for Forex price prediction. The Artificial Fish Swarm algorithm mimics the movements of fish populations, incorporating their intricate social behaviours and "colony" ideologies [33]. Moreover, the algorithm displayed rapid convergence, robustness, and accuracy, allowing it to outperform alternative approaches. The work of Chiong et al [19] supports this by showing an SVM

with Particle Swarm Optimization (PSO) outperforming deep learning networks in financial market trend prediction. The SVM-PSO technique achieved a 60% classification accuracy on test data, displaying significant improvement from standalone classifiers.

Similarly, Islam et al [2] provide a review of the optimization techniques enforced within the literature, concluding that swarm optimizations are far superior to the rest. It is also noted that algorithms such as ant colony, whale optimization, route optimization and Gray wolf optimization are not explored in the current literature. The paper, however, provides an explanation for this and claims that optimization-based techniques are not universal. It is suggested that optimizations are system and dataset dependent and so should cater to the model's application domain. This may explain why approaches to model tuning are generally ubiquitous within the research. Moreover, it can be concluded that there is room for further research into optimization techniques for financial market prediction.

Another fundamental optimization technique within supervised machine learning is feature selection. This is where a subset of an initial feature set is selected to retain important information and prune redundant features. Gerlein et al [5] motivate the use of feature selection techniques, saying how they are critical to a model's performance. Moreover, Baasher & Fakhr [21] explore feature selection by comparing Tree-based selection to Recursive Feature Elimination (RFE). The Tree-based technique utilized the ranking capabilities of decision tree classifiers to prune redundant features below a chosen node. Meanwhile, RFE was used as a wrapper in which features were recursively removed until an ideal number of features was found. The study concluded that the Tree-based feature selection method beat RFE when predicting Forex trends for 3 out of the 4 tested currency pairs.

Khan et al [25] provide a different optimization technique, specifically catered to social media sentiment data. In the study, a spam tweet reduction algorithm was employed to filter input data from Twitter prior to model training. Results showed

that applying this pre-processing step alongside standard feature selection techniques, such as Principal Component Analysis (PCA), improved model results drastically.

2.7.5 Other Approaches

As well as the classic machine learning approaches to prediction discussed above, there exists a range of other techniques that are present in the literature. For example, the work of Carapuço et al [34] presents a Reinforcement Learning model for Forex trading. Within the study, agents were employed under a Q-learning algorithm and were used to make informed decisions of when to buy, sell or hold a currency pair. This unorthodox approach was proven to be financially viable when backtested, achieving annual returns of 16.3+-2.8% and beating baseline buy-and-hold strategies.

Alternatively, the work of Fonseca et al [35] proposes a non-machine learning approach to optimizing investment portfolios in the stock market. It is interesting to note that both fundamental and technical analysis was employed identically to the earlier machine learning approaches for feature generation. Fundamental indicators such as growth ratio and annual earnings were used within rigid rule sets to rank stocks by a defined "attractiveness". Technical indicators, such as Relative Strength Index (RSI), were then used in rigid rules to determine when to long/short the chosen stocks. Finally, a genetic algorithm was employed to help further decide when to make trades. This approach beat baseline strategies and indicates the potential for exploring more hybrid techniques which implement both machine learning and non-ML components synchronously.

Finally, alternative prediction variables are shown by Oliveira et al [3] in which Stock behaviour was analysed in terms of volume, returns and volatility. The paper looked at how sentiment data can be used for modelling market movements and illustrates a different abstraction of the problem. Results showed that Twitter sentiment did not correlate with next day stock returns. This opposes the results of Rao & Srivastava [28] which showed that Twitter sentiment positively correlated with stock prices. Moreover, this suggests that more effort can be exerted on exploring the prediction of various

output variables to better understand financial markets.

2.7.6 Conclusions

The current research has been thoroughly reviewed in which a range of approaches and techniques have been analysed. From this, several conclusions can be drawn pertaining to the current literature and potential points of exploration:

1. Trend prediction is more transparent than price prediction due to the noise associated with raw price series and the benefits of simplifying assumptions.
2. Data pre-processing and labelling techniques are generally unexplored within the literature.
3. LSTMs as neural networks and hybrid SVM's have not been used for exchange rate forecasting.
4. Hybrid and ensemble approaches have been proven to be successful in financial market prediction and can outperform standalone models.
5. Technical Analysis is applied rigidly within the literature without looking at contemporary indicators.
6. Sentiment and Fundamental Analysis has been proven to be beneficial within the Stock market but has not been efficiently applied to the Forex market.
7. The evaluation metrics currently used are opaque and present an area for potential innovation.
8. Optimization techniques are critical to improving model performance.
9. Alternative output prediction variables provide new insights into market movements.

2.8 Project Requirements

The specification requirements for this project follow from the discussions and analysis of the current literature and motivating material. These requirements have been split

into functional and non-functional requirements and ranked according to the MoSCoW method. It is important to note that this work entails a research project and so a final system or interface was not initially proposed, thus the following requirements detail points of research and exploration. An interface has, however, been detailed later in this report but has been explicitly separated from the overarching research to highlight the experimental results and research components.

M – Must have

S – Should have

C – Could Have

W – Will not Have

Functional:

1. Parse an inputted dataset of historic Forex OHLC data, sanitise it, perform EDA, and prepare it for feature generation. (M)
2. Study data labelling techniques for Forex time series. (M)
3. Research technical analysis for feature generation and generate a data frame containing a range of technical indicators. (M)
4. Research Sentiment Analysis for the Forex Market, evaluate its efficiency within the market, and generate a set of sentiment features. (M)
5. Explore Fundamental Analysis for the Forex Market, evaluate its efficiency within the market, and generate a set of fundamental indicators. (M)
6. Research data transformation techniques to achieve stationarity of time series data. (M)
7. Implement a range of feature subset selection techniques. (M)
8. Implement a range of feature extraction techniques that can generate features from generic OHLC data. (S)

9. Implement a range of machine learning models for binary classification in which the next day trends of currencies pairs are predicted. (M)
10. Implement a range of machine learning regression models that can predict the closing price of a given currency pair on the next day of trading. (W)
11. Explore neural networks, explicitly LSTMs for Forex price/trend prediction. (C)
12. Analyse different evaluation metrics including measures of model financial viability. (M)
13. Visualise the performance of each implemented technique and be able to compare the efficacy of each model. (S)
14. Implement hyperparameter optimization methods for the best models to improve the accuracy of the top performing approaches. (S)
15. Explore hybrid/ensemble models to build a pipeline for prediction. (S)
16. Implement an automated trading system that enters the trades signalled by the best models on the live forex market (W)

Non-Functional:

1. Implement classifier that attains an 80%+ classification accuracy when predicting next day trends of a currency pair. (C)
2. Store pricing data in an SQL database and synchronously update data with the most recently available (C)
3. Pull pricing data on the fly via a REST API (W)
4. Attain over 40% returns on financial back tests with best model (C)

3 Methodology & Tools

A range of technologies and tools were utilized during the research stage of this project. Due to the analytical and data heavy aspects of the specification, Python was used as the main programming language. Using Python allowed for rapid implementation of algorithms as well as data analysis and visualization through Python Dataframes and graphing libraries such as Matplotlib. Moreover, Python libraries such as Scikit-learn and Keras/Tensorflow provided convenient out of the box machine learning models and methods for evaluation and tuning. Similarly, research was predominantly conducted through a series of Python Jupyter Notebooks. These notebooks enabled both detailed documentation and a modularized implementation environment which synergized well with the nature of the project. A full list of the technologies and resources used can be seen below:

- *Dukascopy* - A world renowned swiss bank which provides free historic Forex data for both major and minor currency pairs. The bank allowed access to over two decades of historic pricing data for different granularities, as well as an API for developers, making it an easy choice as a data source. Alternative data sources provided similar functionalities; however, most resources were subscription based. Moreover, Dukaskopy provided the most reliable and flexible data for a range of currency pairs whilst being free of charge.
- *Git* – For continuous version control.
- *GitHub* – To store a backup of the project, enabling access to project files from anywhere.
- *Trello* – Used to visualize objectives in a Kanban with new boards for every sprint to break down objectives into tasks. This allowed for prioritization of tasks and accountability, making project management streamlined and more efficient.
- *Python* – Main programming language used for this project's development. This is because of its vast range of libraries with out of the box applications. The

availability of libraries and open-source tools in Python makes developing machine learning models more intuitive and readable.

- *Jupyter Notebook* – Used throughout development as it allowed for dynamic coding and documentation within a single environment. This enabled data visualization of results, graphs, and equations whilst maintaining thorough documentation.
- *TKinter* – Used to build the dashboard user interface that was implemented after the project research concluded.
- *Medium* – Website containing various data science blogs which was used to gain inspiration pertaining to applications of machine learning for financial time series. These articles would often explore nuanced ideas that were beneficial for the analysis of financial markets.

4 Research

4.1 Design Overview

The design of this project closely relates to the fundamental components of the data analysis and supervised machine learning stack. This workflow can be broken down into seven critical modules: data pre-processing, labelling, feature generation, feature selection, model testing/training, evaluation, and model tuning. Figure 6 depicts this workflow in more detail and provides a transparent view of the partial order imposed by the project. One key aspect of this flowchart is the self-loop from the financial backtest & evaluation module back to data acquisition. The self-loop encompasses the iterative nature of the project and how, at every completed cycle of research, more is learned from the data, thus the potential for further discovery grows. This is a crucial concept that holds throughout this work and illudes to the continuous opportunity for exploration within the research domain.

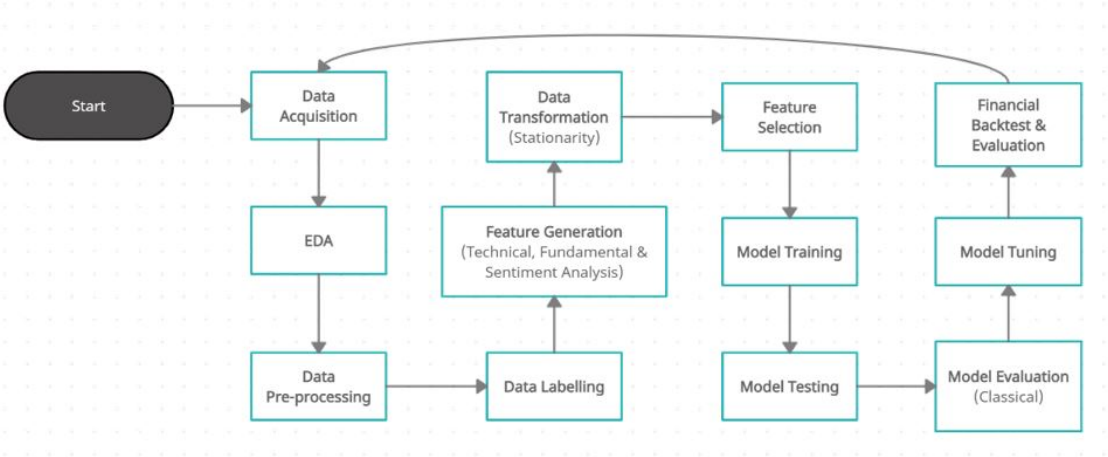


Figure 6: Project Workflow

4.2 Data Pre-processing

4.2.1 Data Acquisition

Figure 7: Daily-FX EUR/USD Trader Positioning from sentiment report [8]

Initially the EUR/USD currency pair was used as a ubiquitous dataset due to its popularity as a major currency pair, making it a more documented price series. However, upon further analysis, it was found that such major currency pairs possessed inconvenient attributes. Specifically, the EUR/USD pair held a 24% (2019) market share in the Forex market [36]. This percentage has grown since and makes the pair the most liquid and of the highest interest to the big banks. The big banks are important as they are the prime mover within most financial markets, often known as "whales", they hold the largest cumulative market share and can swing markets with their financial power alone [37]. Figure 7 shows the positions held by a representative sample of retail investors for the EUR/USD pair from August 2020 to January 2021. The chart demonstrates how the exchange rate moved in relation to trader sentiment. It can be observed that when trader sentiment was net short (red), the general price trend was still increasing. This illustrates the influence of the big banks on major currency pairs as they hedge their trades against general retail sentiment, contrary to the common outlook of the market. Similarly, this could suggest an inverse relationship between trader sentiment and price trends, which will be explored later in this work. Moreover, it became imperative to

acquire data from less biased datasets and to adjust predictions by considering market influences holistically. Thus, the AUD/USD currency pair was used for analysis as it held a mere 6% market share in 2021 [38] whilst being a well reported and documented pair.

After acquiring historic Forex data from Dukascopy in the form of CSV files, they were loaded into Jupyter notebooks by leveraging the Pandas DataFrame structure. Figure 8 shows the output of a Pandas DataFrame holding 8 years of daily AUD/USD data.

Date	Open	High	Low	Close	Volume
2011-01-02	1.02070	1.02267	1.01582	1.01687	69778.4109
2011-01-03	1.01687	1.01716	1.00294	1.00517	124906.5397
2011-01-04	1.00517	1.00770	0.99613	0.99961	119343.0942
2011-01-05	0.99961	1.00159	0.99338	0.99449	99106.5343
2011-01-06	0.99448	0.99932	0.99087	0.99614	101413.4171
...	...	...	...	...	...
2018-12-25	0.70428	0.70723	0.70333	0.70693	77848.7096
2018-12-26	0.70693	0.70786	0.70173	0.70352	193196.9914
2018-12-27	0.70338	0.70698	0.70287	0.70412	147956.1087
2018-12-30	0.70450	0.70721	0.70334	0.70534	108426.1493
2019-01-01	0.70542	0.70554	0.69826	0.69853	190119.7498

Figure 8: Pandas DataFrame with sample data

The final aspect of the data that was considered was the data frequency used. Forex OHLC data comes in tick by tick, minutely, hourly, and daily granularities. Depending on the frequency used, both the size of the dataset acquired, and the amount of information encoded within the data changes. Tick data shows aggregated prices on an order-by-order basis from a range of exchanges. This form of data holds the most information as it is the closest to the raw Forex price series. Intuitively, as data granularities become larger, the amount of information retained in their values is reduced.

Although it may seem obvious to use tick data for analysis, issues arise when modelling the classification problem of smaller frequencies. For example, predicting short periods becomes impractical as trades are not able to match the frequency of prediction. Similarly, the smaller the granularity, the smaller price deviations become between observations in the time series. This makes predictions close to a random walk and futile for research. On the other hand, resampling techniques or predicting longer periods can compensate for this, whilst maintaining the benefits of smaller granularities.

Another issue with smaller granularities such as tick or hourly data stems from the project environment and limitations surrounding the project. That is, due to the global pandemic surrounding research time, processing was constrained to a local workstation. This bounded the computational capacity of the research and meant that processing millions of observations through short frequencies was intractable. As a result, daily data was used throughout the project. Baasher & Fakhr [21] provide a supporting argument for using daily data for Forex trend prediction as it allows for the simplification of the classification problem. By using daily data, next day trends could be predicted which abstracted the complexity of the noise generated by shorter data granularities. Moreover, using daily data meant that true trends within market movements could be predicted.

4.2.2 Exploratory Data Analysis (EDA)

Exploratory Data Analysis and pre-processing are vital initial steps for analysing data sets as it enables the identification of high-level patterns and trends. Now that data has been acquired and stored in a DataFrame, EDA can be performed to better understand its distributions, and statistical properties. Various EDA methods exist, and this section provides an overview of the employed techniques within the research and an evaluation of their viabilities.

Filling Missing Values -

Before any stylized facts can be discovered about the data, it first must be cleaned by filling missing values. Within the raw price series, values can be missing for various reasons such as data filling, aggregation, or resampling errors. Furthermore, caution must be taken when selecting missing values within time series data due to its time dependency. The simplest approaches for this are the fill forward and the fill backward methods. These methods iteratively use the previous or next values from the missing data point as the new observation. Although these methods provide simple solutions, large jumps within the data become apparent if there are copious amounts of missing values.

Moreover, more convenient approaches can be used such as linear interpolation. Interpolation pertains to the adjustment of a function to fit a dataset in order to extrapolate missing values. Linear interpolation models a straight line in a series and attempts to fit missing values to it. Letting y = f(x), the value given to the missing value at x is given by:

$$f_{LI}(x) = f(x_1) + \frac{f(x_2) - f(x_1)}{x_2 - x_1} * (x - x_1) \tag{4.1}$$

Linear interpolation allows for rapid calculation of missing values however suffers in accuracy. This is because functions are not differentiable at x and so the associated interpolation errors are proportional to the square of the distances between data points.

Another type of interpolation is Polynomial Interpolation, which generalizes linear interpolation and can often produce more accurate results. N-th order polynomials are infinitely differentiable smooth functions that can fit any set of data points, making them convenient for this problem. The Python library Scipy provides functionality for many of these techniques, however, Langrage Polynomials were specifically used for this project. The Lagrange Interpolating Polynomial is the polynomial P(x) of degree $\leq (n-1)$ for the dataset$\{x_i, y_i\}, i = 0, 1, \ldots, n$ and is given by [39]: $P(x) = \sum_{j=1}^{n} Pj(x)$

where,

$$P_j(x) = y_j \prod_{i=1, i!=j}^{n} \frac{(x - x_i)}{(x_j - x_i)} \tag{4.2}$$

and $P_j(x)$ are Kronecker Delta functions in which values are elements of the set [0,1]. Moreover, Polynomial Interpolation provides a good approximation for certain data points that sit on the fitted curve. On the other hand, it suffers from the same problem of linear interpolation in that it cannot account for the chaos/unpredictable intricate market movements of financial data.

In reality, Forex data does not stick to a curve or a single straight line and so modelling the function of a historic price series is difficult. Similarly, if there are large amounts of missing data, accuracies dramatically decrease as all interpolated prices follow the interpolating polynomial. A simpler solution may be to remove observations completely from the series that contain missing values. Such a naïve solution reduces complexity but comes at the cost of losing information as the temporal dependence chain is weakened. Furthermore, there exists a fundamental trade-off between the accuracy of interpolated values and the computation overhead of calculations.

Fortunately, the datasets acquired from Dukaskopy contained minimal missing values. The only column that contained any such values was the trading volume field. Trading volume was not imperative for calculations and so polynomial interpolation was employed to deal with the missing data. Similarly, it was noticed that some records within the data observed zero trading volumes which suggested that trading was halted for those days. Looking at the previous days OHLC prices confirmed this as prices remained unchanged, therefore records with zero trading volume were pruned from the time series to maintain consistency.

Removing Outliers -

The removal of noise/extremities from data is a key aspect of data cleaning. Within a dataset, outliers may exist due to extreme or rare conditions causing sporadic results or even by propagated errors. The identification of these values is important to determine whether they should be removed from the data. This is especially true for Forex data as influxes in price values may skew future predictions, resulting in wider errors and potentially monetary loss.

Rigid rules can be employed to remove all values outside of a defined boundary without discretion. For example, let σ be the standard deviation of the daily returns of a historic forex dataset where

$$\sigma = \sqrt{\frac{\sum (x_i - \mu)^2}{N}} \tag{4.3}$$

and μ be the mean of the daily returns for N observations. A boundary can be defined by n-standard deviations of the mean of the daily returns. Figure 9 demonstrates this concept for the AUD/USD currency pair in which outliers are defined as values with a daily return outside of 2 standard deviations of the mean. The red points indicate values that lay outside of the defined boundaries in which 32 outliers are detected from 2522 points of daily data. By changing the value of n for the number of standard deviations used to 3, the number of points classified as outliers' drastically increases to 148 (Figure 10). Moreover, this approach simplifies the cleaning process as outliers can easily be identified, however, the variable sensitivity may cause important information to be deemed as extreme and be unwillingly removed. This is because the context of the time series data is not accounted for by a global mean and standard deviation. Furthermore, n is a hyperparameter that can be tuned to change the cleaning methods sensitivity and must be determined to best suit the dataset being processed.

After identifying outliers within the time series, they can be removed to swiftly clean the data or more processing can be applied to preserve data points.

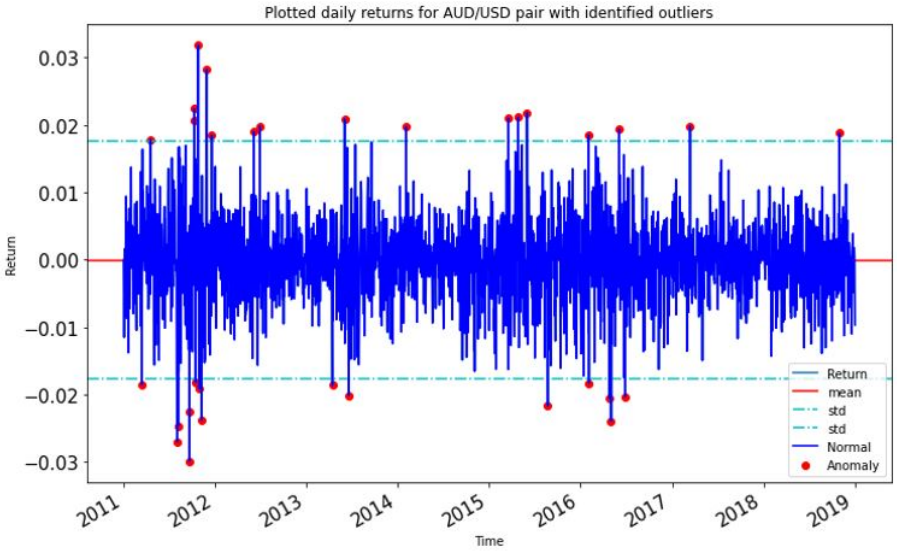


Figure 9: AUD/USD Pair with n-standard deviation outlier detection method

```
0    2490
1      32
Name: Outlier, dtype: int64
```

```
0    2374
1     148
Name: Outlier, dtype: int64
```

Figure 10: Number of outliers detected within AUD/USD dataset when n = [2,3]

Winsorization is a technique that supports the preservation of data by transforming potentially spurious extreme values into a defined and limited range. By doing this, the effect of outliers and extreme values is reduced without having to lose data points. Synonymous to clipping within signal processing, Winsorization can adjust defined outliers to become equal with an accepted extreme. For example, in Figure 9, the outliers in red can be set to the equivalent values at the chosen n-standard deviations boundary (dashed line). This technique is particularly useful when dealing with multi-dimensional data as outliers are defined by simple returns alone, thus completely removing identified points can result in significant losses in information.

Moving averages can also be employed when defining boundaries for identifying outliers within a Forex time series. In this approach, the mean and standard deviation of the data depend on a rolling window rather than across the entire dataset. By doing

this, the calculated statistical properties follow the most recent trends and so outliers can be detected relative to nearby values. Figure 11 demonstrates this approach in which the mean and standard deviation values were generated using a 30-day rolling average of the daily returns, mimicking a typical trading month. The standard deviation barriers were then defined with n=2 to maintain consistency with the previous method. The results in Figure 12 suggest that this new approach had no effect on the number of outliers detected despite the increase in complexity.

More complex Exponential Moving Averages (EMA) can also be used to apply more weighting to the most recent values, introducing more contextual knowledge to the data cleaning process. However, regardless of what moving average technique is used, a new hyperparameter is introduced in the form of the number of periods used within the rolling average. This must be tuned as well as the number of standard deviations used as a boundary to acquire the best results. Moreover, a trade-off exists within data cleaning and outlier detection between model sensitivities and complexity. Therefore, a simple n-standard deviations discriminatory approach presents a good approximation to efficient outlier detection whilst minimizing computational costs.

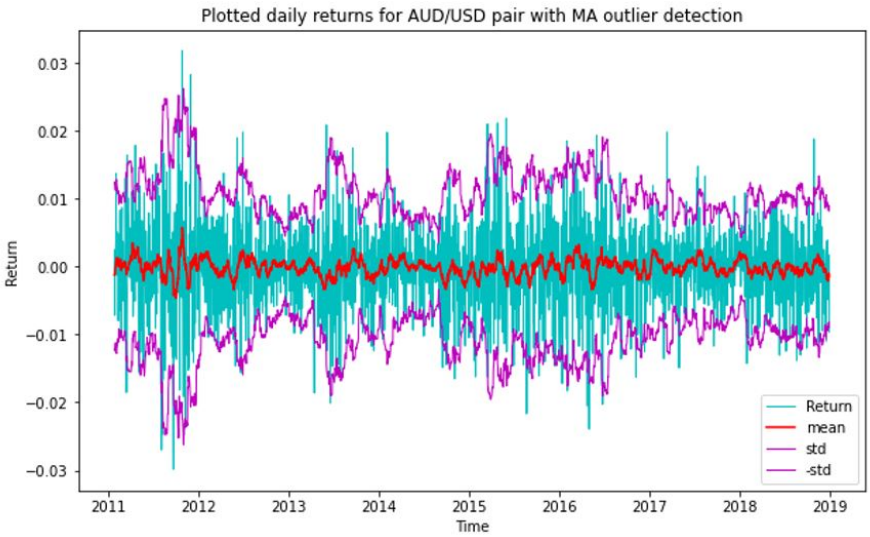


Figure 11: Outlier detection using moving averages for AUD/USD dataset

Figure 12: Number of outliers detected using MA method with n=2

Stylized Facts -

Statistical properties and empirical facts can be acquired from analysing cleaned data. By observing the attributes of data, more features and historic patterns can be observed and leveraged for the prediction of future trends.

An important feature to be generated for analysis is log returns. These returns exhibit beneficial attributes such as stationary in which the mean, variance and autocorrelation of the price series becomes constant over time. This is critical for most statistical forecasting models. Moreover, initial data analysis within the project heavily depended on observing patterns within the simple and log returns of historic Forex data. Log returns are defined as the following:

$$r_t = ln\left(\frac{P_t}{P_{t-1}}\right) = ln\left(P_t\right) - ln\left(P_{t-1}\right) \tag{4.4}$$

where P_t and P_{t-1} are the prices at time t and $t-1$ respectively. Python provides a convenient method for acquiring instantaneous statistical properties pertaining to a DataFrames contents. Figure 13 illustrates these properties for the previously cleaned AUD/USD dataset after removing noise. It can be noted that log returns hold very similar statistical properties to simple percentage returns thus can be utilized in the same way. It can also be seen that the mean returns for the series are both negative and small in magnitude, implying that daily market movements are small. This could suggest that next day trend prediction could be difficult to approach, and more flexible prediction periods should be explored.

	Open	High	Low	Close	Volume	Return	Log Return
count	2371.0000	2371.0000	2371.0000	2371.0000	2371.0000	2371.0000	2371.0000
mean	0.8697	0.8727	0.8662	0.8696	108899.4752	-0.0001	-0.0002
std	0.1265	0.1268	0.1260	0.1265	83408.7241	0.0045	0.0058
min	0.7019	0.7045	0.6983	0.7019	0.0000	-0.0119	-0.0488
25%	0.7564	0.7583	0.7534	0.7562	73359.1727	-0.0026	-0.0029
50%	0.8213	0.8236	0.8169	0.8206	107190.9692	0.0000	0.0000
75%	1.0092	1.0130	1.0037	1.0084	140444.3099	0.0026	0.0027
max	1.0779	1.0818	1.0733	1.0776	513169.4427	0.0116	0.0286

Figure 13: Basic statistical properties of cleaned AUD/USD dataset

The (Gaussian) Normal Distribution is a probability function that features heavily as an assumption upon which many financial models depend on. In this distribution, the mean, median and mode are identical as there exists symmetry about the mean value. Within financial market analysis, returns modelled by a normal distribution are highly sought after as they allow for the least investment risk. This is because the expectation of returns on an investment hang about the mean, making market predictions more desirable. The normal probability density function (PDF) is defined as follows:

$$f\left(x\right) \ = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}, \tag{4.5}$$

where σ and μ are the standard deviation and mean of returns, respectively.

Figure 14 shows a histogram and PDF for the percentage returns, log returns, and high low-price average of the cleaned AUD/USD dataset. It can be observed that all three distributions deviate from the standard bell curve with high peaks and narrow tops for the distribution of returns and high skew within the adjusted close distribution. To test for normality of these series, a kurtosis value can be calculated in which the tails of a distribution are weighted and combined, relative to its centre. Moreover, kurtosis describes how data is clustered within the tails or peak of a distribution and whether it resembles a bell curve. When kurtosis is zero, the distribution is identical to that of a normal distribution, also known as mesokurtic. If kurtosis is negative, then

the distribution is flat with short tails, indicating fewer extreme values. This is also known as a platykurtic distribution. Finally, if kurtosis is positive then the distribution possesses thick tails and a high peak. Large positive kurtosis values suggest that more data points are distributed about the tails rather than the mean, potentially exhibiting more frequent extreme behaviour. This distribution is also called Leptokurtic [40]. Moreover, Kurtosis is calculated by dividing the fourth central moment by the variance squared, as follows:

$$kurtosis \ = \frac{1}{N} * \frac{\sum_i^N (x_i - \mu)^4}{\sigma^4} - 3, \tag{4.6}$$

where N is the number of data points, μ is the mean value and σ is the standard deviation. It is important to note that a value of three is subtracted from the calculated Kurtosis value to attain the "excess kurtosis". This ensures that a standard normal distribution has a kurtosis value of zero, rather than three, as prefaced earlier.

Skewness can also be used as a measure of distortions from a bell curve in a probability distribution. For a normal distribution, skewness is near to zero whereas negative and positive values of skewness indicate that data are skewed left and right respectively. The formula for Skewness used in the project follows the Fisher-Pearson coefficient of skewness, which is defined as follows:

$$skewness = \frac{1}{N} * \frac{\sum_i^N (x_i - \mu)^3}{\sigma^3}, \tag{4.7}$$

where N, μ and σ are the number of datapoints, mean and standard deviation, respectively.

Fortunately, the SciPy Python library provided tests for Kurtosis and Skewness and so was utilised to test for normality in the AUD/USD dataset in Figure 15. The library also provided methods for both tests as well as normality in which null hypotheses were tested, and p-values were produced. For skewness, the function tested the null hypothesis that the skewness of the data was the same as that of a corresponding normal distribution. For Kurtosis, the function tested the null hypothesis that the kurtosis of

the dataset was also that of the gaussian distribution. Finally, SciPy provided functionality for normality that tested a similar null hypothesis for normal distributions. Furthermore, P-values less than 0.05 were enough to reject either null hypothesis and conclude that the distributions were either skewed, non-mesokurtic or not a normal distribution.

A comprehensive set of results for each of these tests can be seen in Figure 15. It can be observed that all the distributions have positive kurtosis values and are Leptokurtic. Similarly, they all attain P-values < 0.05 on the Kurtosis test which is enough to reject the null hypothesis that the simple percentage return, log return or adjusted close price have kurtosis matching those of a Gaussian distribution. This is further reflected in the normality test in which all P-values indicate a rejection of the null hypothesis. Furthermore, it can be concluded that none of the distributions are normal.

The value of Skew for the log returns distribution is negative with a P-value < 0.05, thus rejecting the null hypothesis that the log returns have a skewness identical to that of a bell curve. Again, the value of Skew for the adjusted closing price (high low average) is positive and rejects the null hypothesis with skew towards the right tail. Contrarily, for the distribution of simple percentage returns, the Skew test displays a P-value > 0.05, thus accepting the null hypothesis that the returns distribution has a skewness matching that of a normal distribution. This is interesting as the log returns acquired a similar skew score and is visually alike, however, rejects the null hypothesis.

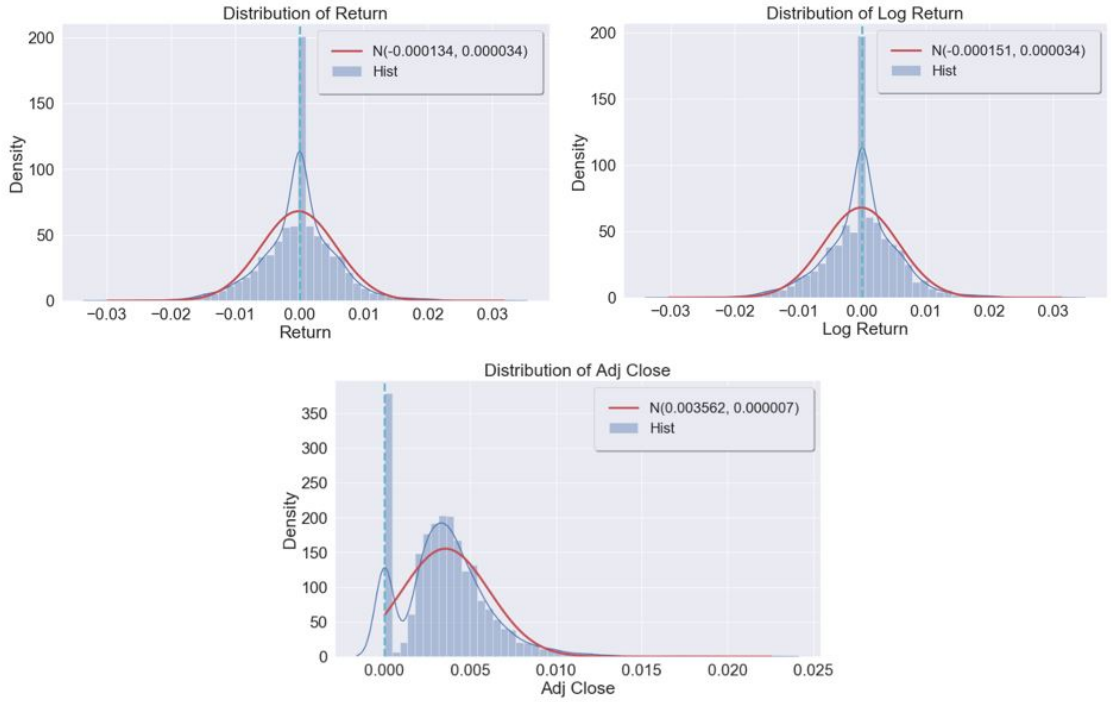


Figure 14: Distributions and Histograms for simple returns, log returns and high-low average

	skew	skew_test (p-val)	kurtosis	kurtosis_test (p-val)	normal_test (p-val)
Adj Close	1.089735	3.175465e-76	3.647202	1.788210e-49	2.455087e-122
Return	-0.069047	1.562392e-01	2.321321	1.555796e-32	8.535517e-32
Log Return	-0.107018	2.823949e-02	2.318565	1.700940e-32	2.296172e-32

Figure 15: Results from Skewness, Kurtosis & Normality tests

Although these results suggest that daily Forex returns do not model a gaussian distribution, Aggregational Gaussianity, a well-recognised stylized fact within financial data, can be used to estimate normality [41]. Aggregational Gaussianity states that by increasing the period in which returns are calculated, their distributions become more alike to a normal distribution. To demonstrate this, a series of lags were generated for the log returns feature of the AUD/USD dataset. For each of these lags, histograms and probability distributions were plotted, as shown in Figure 16. It can clearly be observed that as the size of the period used to calculate the log returns increase (lag number increases) the peak of the distribution decreases to become more alike a gaussian distribution. The lag_5_log_rtn feature used the widest window when calculating returns and consequently had the most normal looking distribution.

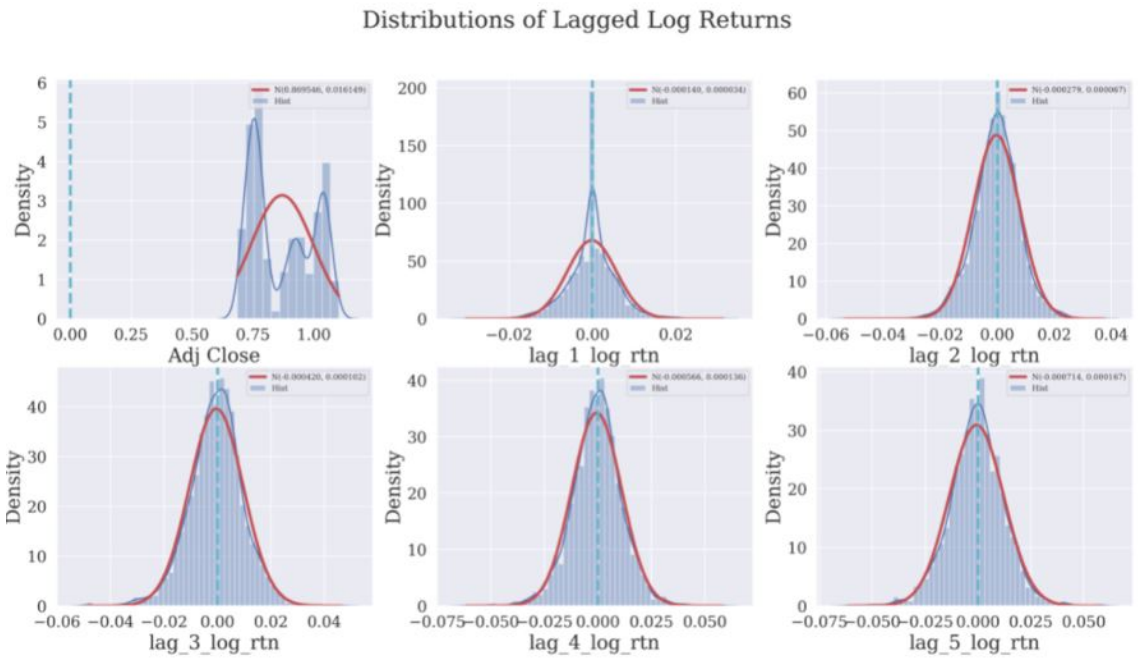


Figure 16: Aggregational Gaussianity through distribution plots of lagged log returns

In conclusion, it has been shown that normality cannot be assumed for the returns within this specific historic Forex dataset. Moreover, by leveraging facts such as Aggregational Gaussianity, more normal distributions can be achieved. Furthermore, this suggests that using wider periods for calculating returns may prove more beneficial for prediction than daily returns as they model a less erratic distribution. Thus, this must be considered when labelling data and choosing the output variable for prediction and training machine learning models.

Decomposition -

The distribution of the adjusted close prices, shown previously in Figure 14, appears to be the furthest from a normal distribution. This is due to the seasonality and time dependence of forex time series data. As returns and log returns attain stationarity, this attribute is abstracted, allowing for a more normal distribution. To observe the seasonality components of the raw price series, a seasonal decompose can be performed.

Time series data that is time dependant can be decomposed into base level, trend, seasonality, and residual/noise components. The base level illustrates the average values in the series whilst the trend is the underlying direction in which the data moves. Furthermore, seasonality is the repeated cycles that appear in the series and the noise/residual is the excess random variation. It is thought that a series is an aggregate of these components either in an additive or multiplicative model [42]. An additive model is linear in which seasonality expresses constant frequency and amplitude of cycles and a linear trend. Alternatively, a multiplicative model is non-linear with variable seasonality cycles and a non-linear trend. By decomposing the time series, more information can be gained about the data and it provides insights on what must be considered during forecasting and trend prediction.

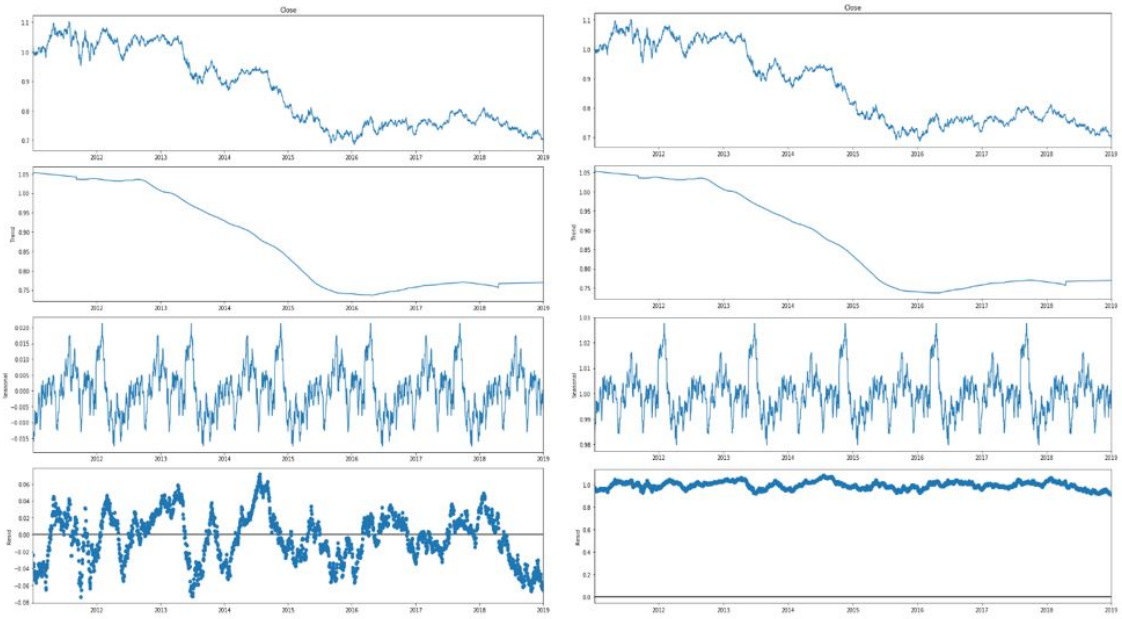


Figure 17: Additive and Multiplicative seasonal decompose of AUD/USD dataset

The statsmodel Python library provided convenient implementation for seasonal decomposition of time series data and was utilised during this analysis. Figure 17 depicts the additive and multiplicative decomposition for the AUD/USD dataset. Both decompositions display the trend of the data as a downtrend with clear seasonality. The obvious discrepancy between the two methods is the residual component in which

the additive decompose retains more information that may include excess patterns. On the other hand, the multiplicative residual plot is more constant without cyclic trends. Moreover, both plots demonstrate the general trend of the dataset and confirm the implicit seasonality of the series that must be made stationary before model training.

Correlation -

Correlation is the analysis of how two variables are linked together in their movements. Within data analysis, correlation can be between input and output variables or between a series and its own lags. Autocorrelation is the measure of correlation between a series and its own lags. Here, if a series is found to be heavily autocorrelated, it indicates that lagged values (previous values) of the series can be leveraged for predicting the current value. Partial Autocorrelation exhibits similar behaviours; however, it differs in that it is the raw correlation between a series and its own lag, without intermediate lags. By removing components already analysed from previous lags, the lags that are correlated with the residual are identified.

The Forex time series is said to be an Autoregressive (AR) process. This is because the current value of the series can be obtained by using previous values of the same series i.e., lags, as a weighted average. An Autoregressive process of order p is shown by the formula [43]:

$$y_t = c + \beta_1 y_{t-1} + \beta_2 y_{t-2} + \ldots + \beta_p y_{t-p} + \epsilon_t \tag{4.8}$$

where y_t is the current value in the series, ϵ_t is some white noise and y_{t-1} and y_{t-2} are lags of the series.

The value for p determines the number of lags that can be used as optimum features when forecasting an Autoregressive process. This value can be identified from a partial autocorrelation plot as it identifies lags that correlate with components independently. An autocorrelation plot would be unable to capture this as most lags will exhibit cor-

relation with the process, which can cause multicollinearity issues if a large value for p is chosen.

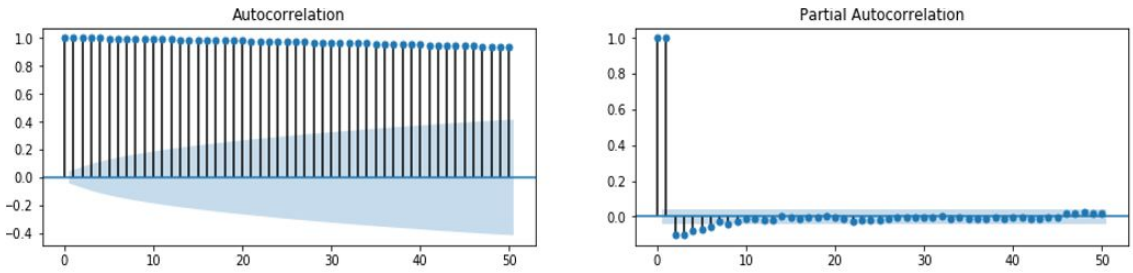


Figure 18: Autocorrelation and Partial Autocorrelation plots for AUD/USD dataset

Like the decompose functionality, the statsmodels library provided methods for visualising autocorrelation and partial autocorrelation of Forex pricing data. Figure 18 illustrates these graphs for the AUD/USD dataset. It can be observed that there are high levels of autocorrelation within the series by the slow decrease in correlation values as the number of lags increases in the autocorrelation plot. Within the partial autocorrelation plot, the correlation of new lags significantly decreases after the first two lags. This indicates that the first two lags of the series can best predict current values and can be determined as an optimal value for p.

Finding values for the order p is important as models such as ARIMA (Autoregressive Integrated Moving Averages) can be used to forecast the time series. This model utilises the attributes of Autoregressive and Moving Average models as well as a differencing function to make predictions. Models such as the ARIMA model have been proven to be beneficial for financial market prediction as shown by the work of Weerathunga & Silva [17]. Furthermore, analysing the autocorrelation and partial autocorrelation of historic Forex datasets provides valuable insights into suitable predictive models and powerful features.

Other Approaches –

Other approaches to EDA include correlation heatmaps of input variables, pair plots and lagged plots which all follow from the concepts explored within this section. Moreover, many of these visualisations are investigated further when performing feature selection on the finalized datasets. This again demonstrates the iterative nature of the research as data analysis is not confined to one section or context of the problem.

In conclusion, EDA can be utilised through various methods depending on type of data being analysed. Furthermore, EDA is not a one size fits all process and so numerous rounds of analysis must be performed to uncover as much information about the data as possible. For the AUD/USD dataset, missing values were filled through interpolation, followed by outlier detection in which extreme values were removed. The distribution of the dataset was then analysed in which it was concluded that the returns of the series did not model a normal distribution. Finally, the dataset was decomposed to identify the seasonality components within the data and autocorrelation plots were generated to determine how lags could be used for prediction.

4.2.3 Data Normalisation/Standardisation

Since multivariate time series data was being classified by models, care had to be taken when preparing the data for training, testing and validation. The fundamental step into data preparation was normalization, where values were fitted to a common scale. This was important as the biases of features with large absolute values were removed such that variables could be compared equally. Without normalization, graph-based models such as SVM's or KNN classifiers would suffer as distant measures would be affected by variable scales. Min-Max scaling and the standard scaler were two widely adopted methods for data normalization/standardization and were provided as functions within the Scikit-learn pre-processing libraries.

Normalization via the MinMaxScaler is the process of rescaling features to a range bounded by 0 and 1. This can be achieved for each feature column by subtracting the

minimum variable value from each observation and dividing by the range such that:

$$x_i = \frac{x_i - x_{min}}{x_{max} - x_{min}} \tag{4.9}$$

Alternatively, Standardization through the StandardScaler rescales the distributions of a series such that it has a zero-mean and unit variance, modelling a Gaussian distribution. This process involves subtracting the mean of each corresponding feature column from every observation and dividing by the standard deviation such that:

$$x_i = \frac{x_i - \mu_x}{\sigma_x} \tag{4.10}$$

The StandardScaler was primarily used within the project as it maintained information carried by outliers. On the other hand, the MinMaxScaler was highly influenced by extreme minimum and maximum values which could subsequently bias the data. Similarly, standardization ensured that distance measuring could be applied equally across graph-based models due to the achieved mean centring and unit variance.

Moreover, when normalising the Forex time series, it was important to fit the StandardScaler on training data before applying and transforming the test data. This was a common mistake within current approaches in which a look ahead bias occurred. By using the statistical properties across the entire series when standardizing, values from the test data would be used on the training data which lead to model overfitting and deceiving results.

4.3 Data Labelling

To train and test machine learning models for classification, class labels must be generated for input data as ground truths. Within this project, a binary classification problem is approached in which binary labels are generated. These labels can either indicate uptrends or downtrends depending on the way in which the exchange rate moves in the future. This is as opposed to the multiclass problem in which non-trending regions can be identified and labelled as well. The binary case has been chosen as it

provides a simpler abstraction of the Forex prediction problem. Similarly, the work of Jubert et al [24] suggests that predicting non-trending regions is difficult within the multi-class case. Furthermore, the multi-class problem introduces a new hyperparameter in the form of a threshold that is used to differentiate trending and non-trending patterns. Deciding and tuning this threshold is counterintuitive and so binary labels are employed.

The most widely used labelling technique is the "Fixed Horizon" method. In this method, labels are generated by comparing the current return at time t to the return n periods into the future at time $t + n$. The generated label is determined to be an uptrend (or a 1) if the return increases from r_t to r_{t+n} and a downtrend (or a 0) if this value decreases. To calculate the return for each value and the appropriate label, the following equations are used:

$$Return = r_t = \frac{p_t}{p + t - n}, \quad Label = \begin{cases} 1 & if\ r_{t+n} > r_t \\ 0 & if\ r_{t+n} \leq r_t \end{cases} \tag{4.11}$$

Although the Fixed horizon method provides a simple labelling approach, it fails to capture the intricate Forex market movements and trends over large periods. This is because the returns are generated considering only the current price at time t and the price n steps in the past at time t-n. Moreover, the entire price series between time $t - n$ and t is disregarded. Despite this, the Fixed Horizon method is still the most adopted approach for financial time series labelling.

Prado [27] proposes an alternative data labelling method called the "Triple Barrier Method". This technique provides a solution for the problems associated with Fixed Horizon labelling by incorporating a series of threshold barriers. Whichever barrier the closing price hits first between the current observation at time t and the price at time t+n indicates the generated label. Figure 19 demonstrates this process for labelling a single point. Specifically, a single vertical barrier and two horizontal barriers are implemented. If price crosses the top barrier first then the generated label is an uptrend

(or a 1), otherwise, if the bottom barrier is hit, a downtrend label is chosen (or a 0). Finally, if the end barrier (vertical) is reached without hitting the horizontal barriers, this indicates a non-trending region. In this case, a third-class label can be determined for multi-class labelling. Alternatively, the Fixed Horizon Method could be enrolled to decide whether to label an uptrend or downtrend. Moreover, it is clear to see how this alternative labelling method considers all price points in the series when defining trends and is more suitable for financial applications. Furthermore, such a method is coherent with the Forex market as it supports standard practices such as stop losses and take profits by incorporating the entire price series.

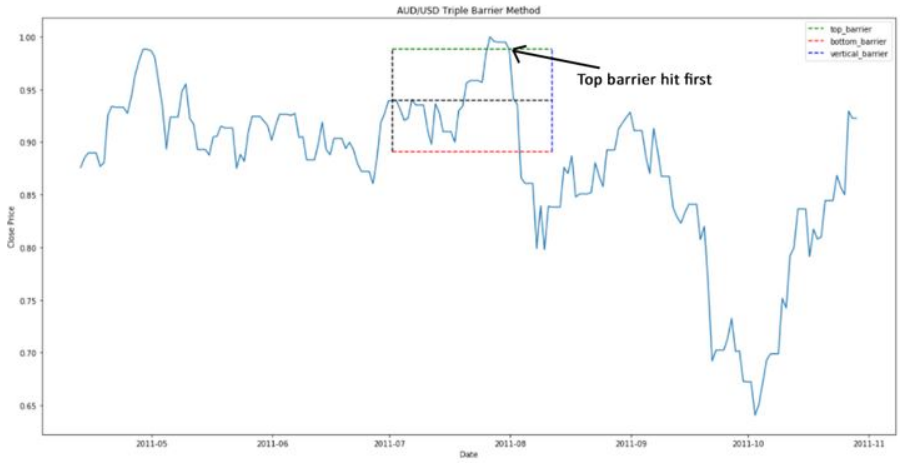


Figure 19: Triple Barrier Labelling Method for single data point in AUD/USD data set

```
df = Load AUD/USD dataset
volatility = ATR(df)
labels = []
holding_period = 10
def triple_barrier(df, volatility, sl_multiplier, tp_multiplier):
  barrier = [] //hold barrier values for each day
  for each day in df:
    if day + holding_period < len(df):
      vertical_barrier <- date(day + holding_period)
    top_barrier = df.Close[day] + df.Close[day] * tp_multiplier * vol[day]
    bottom_barrier = df.Close[day]      df.Close[day] * sl_multiplier * vol[day]
    barriers[day] = [df.Close[day], vertical_barrier, top_barrier, bottom_barrier]
  for i in range(len(barriers)):
    take_profit_range = (barriers.price[barrier.index[i]:barriers.vertical_barrier[i]]
                                          ) >= barriers.top_barrier[i]
    stop_loss_range = (barriers.price[barrier.index[i]:barriers_vertical_barrier[i]])
                                          <= barriers.bottom_barrier[i]
    if take_profit_range.any(): labels[i] = 1
    else if stop_loss_range.any(): labels[i] = -1
    else: labels[i] = FixedHorizonLabel()
  return labels
```

Figure 20: Triple Barrier Method

The logic behind the Triple Barrier method is demonstrated in Figure 20 in which the pseudocode for the technique is described. An important implementation detail is the use of Average True Range (ATR) as a market volatility measure. This technical indicator shows how much the exchange rate of a currency pair moves within a day and is used as a multiplier to calculate the horizontal take profit and stop loss barriers. By using ATR as a multiplier, the size of the barriers become dynamic and can fit the context of the series for different observations, making labelling more financially sound. Another important implementation detail is the use of the Fixed Horizon method when casting non-trending regions to a binary label. Alternative methods have been shown by Prado [27] such as generating real values and rounding to the nearest label, however, leveraging the Fixed Horizon method simplifies this process and supports a hybrid approach.

The Fixed Horizon (FH) and Triple Barrier (TB) methods are compared when labelling observations for the AUD/USD dataset. Figure 21 shows the generated class labels for both methods in which the TB method resulted in 1435 uptrends and 1476 downtrends, whilst the FH attained 1427 uptrends and 1495 downtrends. These labelling results are much closer than originally estimated, although the FH method

is more sensitive towards downtrends within the data. Despite this, both techniques capture the overall trend of the dataset as being a bearish price series.

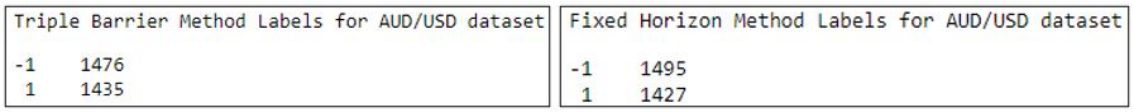


Figure 21: Labels generated by Triple Barrier and Fixed Horizon methods on AUD/USD dataset

Figure 22 provides further information pertaining to the classifications within the TB method. Here, the number of non-trending regions that were then classified using the FH method was 676, approximately 23% of the dataset. This is a significant proportion of the data and so several considerations must be made. Firstly, the multipliers used to calculate the height of the horizontal barriers (take profit and stop losses) and the prediction period must be tuned. Also, the method used to classify non-trending regions into a binary label must be carefully chosen to prevent naïve selection when non-trends hold a significant share of the time series.

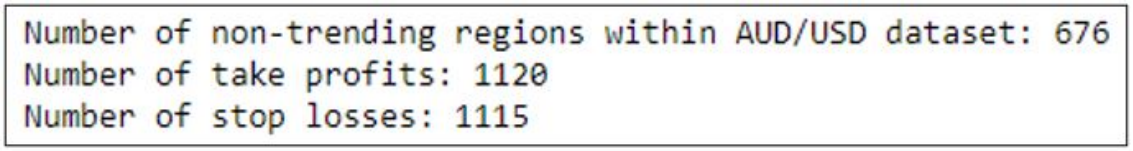


Figure 22: Number of non-trending regions, TP's, and SLs from Triple Barrier Labelling on AUD/USD dataset

Moreover, applying these methods to the AUD/USD dataset has confirmed the complexity that comes with the TB approach as various hyperparameters are introduced. Furthermore, employing TB labelling has not shown any adverse results from standard FH labelling and so was not utilized in the formal testing and evaluation of machine learning models.

4.4 Feature Generation

Feature generation/engineering is concerned with the production of new variables from existing ones to acquire advanced information and abstractions about a dataset [44]. Within this project, a range of feature generation techniques were enrolled to acquire

rich feature vectors for Forex trend prediction. This section details the feature engineering methods used and evaluates their applicability for the Forex market.

4.4.1 Technical Analysis

The primary source of feature engineering for financial market prediction using machine learning is Technical Analysis. This is where historic price action is analysed to identify patterns within a historic price series to make predictions from. Technical Indicators are calculated from this analysis and represent the engineered features from this process. An example of a Technical Indicator was previously demonstrated as moving averages, although there exists a vast number of indicators with unique attributes that can be leveraged for prediction.

Retail investors holistically analyse several indicators to make informed decisions on the positions they hold in the market. This process mimics the prediction process and so technical indicators complement the nature of features in supervised machine learning. The intuition is that if an investor can use a set of indicators to predict market movements, a machine learning model can enforce automation whilst using the same information to achieve higher accuracies.

Technical Analysis for feature generation has been used ubiquitously within the current literature, often with the same indicators being used throughout. Thuy & Vuong [22] adopt this approach in which standard indicators such as RSI, Moving Averages, Bollinger Bands and Average True Range (ATR) are used as features. Specifically, the RSI indicator (Relative Strength Indicator) is a momentum indicator that is used to identify overbought and oversold conditions within a market [45]. With a value ranging between 0-100, the oscillator is classically interpreted by defining values over 70 as the overbought condition and below 30 as oversold. Typically, when RSI passes 70, if it crosses back below 70 then a short position is signalled. Alternatively, when RSI drops below 30, if it goes back above 30 then a long position is signalled. Figure 23 demonstrates the RSI indicator and the associated thresholds.

Although RSI is easy to interpret and is widely employed, it has many limitations. Firstly, the idea of overbought and oversold conditions does not apply to the Forex market. These concepts are often applied to the stock market as assets exhibit intrinsic value. Exchange rates, however, do not have intrinsic value as they are regulated by governments and determined by the big banks how and when their prices move. Furthermore, attempting to apply such concepts to the Forex market is both naïve and contradictory. Similarly, the origins of the RSI indicator date back to 1978, when J. Welles Wilder Jr first created it [46]. This was almost two decades before the spot forex market was opened to retails investors in 1996 [47]. Thus, it can be argued that applying an outdated indicator to the Forex market is ineffective. Finally, the indicator is unable to efficiently capture non-trending regions within the market. This is because the "overbought" and "oversold" conditions are not able to predict when a ranging/non-trending period is eminent. Moreover, many of the standard indicators employed for technical analysis exhibit similar defects and so more consideration must be taken when performing technical analysis.

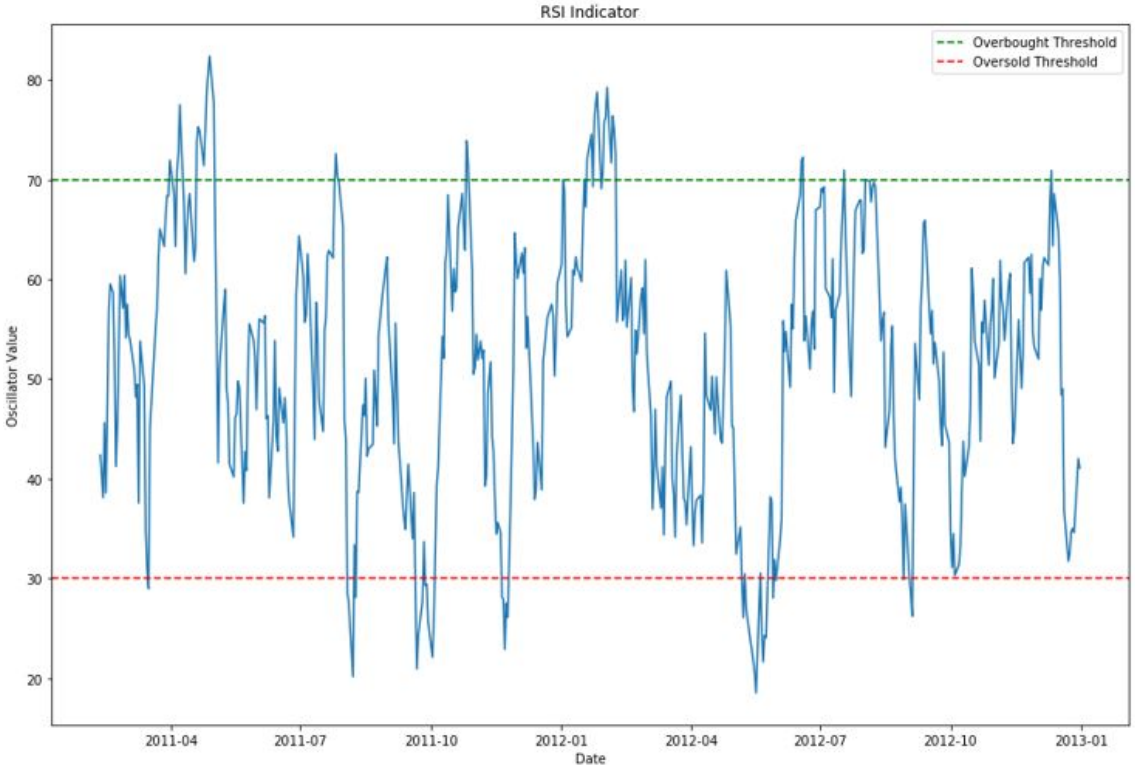


Figure 23: Example of RSI Indicator

Dozens of Technical Indicators, specifically for the Forex market, are developed daily via MQL4 scripts by retail investors and programmers on websites such as TradingView [48]. Hence, it is naïve to use the same limited indicators and expect improved results considering the availability of contemporary, open-source scripts. Therefore, within this project, a wider range of indicators were calculated during the Technical Analysis process. To do this, the Python libraries TA-Lib and Pandas_TA were utilised which provided out of the box functionality for calculating various volume, momentum, volatility, and price indicators. These libraries provided convenient functionalities, however, failed to incorporate the contemporary scripts being made daily by retail investors. To account for this, many indicators that were deemed to be beneficial for prediction were manually implemented by converting MQL4 scripts to Python code.

An example of one of the manually implemented technical indicators is the Semaphore Signal Level (SSL) channel chart alert. This indicator illustrates the dynamics of ex-

change rate movements and emerging trends by combining moving averages, namely High and Low price Simple Moving Averages (SMAs) [49]. Figure 25 displays the SSL channel for sample AUD/USD data in which an SSL up and SSL down line is visualised. When the SSL down line is above the up line, a down trend is indicated. Otherwise, an uptrend is indicated. Moreover, the indicator can show the forming of price trends over large periods through the convergence of two indicator lines. The calculation for the SSL channel indicator can be seen in Figure 24 in which the logic behind the up/down lines are demonstrated. Moreover, the documentation for this indicator was sparse and so implementation required careful consideration and planning as various MQL4 scripts of similar indicators were studied.

```
def ssl_channel_indicator(df, period):
    df['SMA_High'] = SMA(df.High, timeperiod=period)
    df['SMA_Low'] = SMA(df.Low, timeperiod=period)
    hlv = DataFrame(columns=['HLV'], index=df.index)
    sslDown = DataFrame(columns=['SSL_Down'], index=df.index)
    sslUp = DataFrame(columns=['SSL_Up'], index=df.index)
    for i in range(0,len(df.Close)):
        if df.Close.iloc[i] > df.SMA_High.iloc[i]:
            hlv.iloc[i] = 1
        elif df.Close.iloc[i] < df.SMA_Low.iloc[i]:
            hlv.iloc[i] = -1
        else:
            hlv.iloc[i] = hlv.iloc[i-1]
    for i in range(0,len(hlv)):
        if hlv.HLV.iloc[i] <= 0:
            sslDown.iloc[i] = df.SMA_High.iloc[i]
            sslUp.iloc[i] = df.SMA_Low.iloc[i]
        else:
            sslDown.iloc[i] = df.SMA_Low.iloc[i]
            sslUp.iloc[i] = df.SMA_High.iloc[i]
    return sslDown, sslUp
```

Figure 24: SSL Indicator Code Snippet

Although it can capture long term trends, the SSL indicator can also lead to dead ends as lines rapidly and successively converge, as demonstrated between July and August in Figure 25. Therefore, it is common practice to combine the SSL channel with a confirmation indicator that captures market volume, volatility, or momentum such as Average True Range (ATR).

Figure 25: SSL Channel Indicator

Another manually implemented indicator is the Volatility Index (VI) indicator. As the name suggests, the VI is a volatility measure for Forex data and can be used as a confirmation for trend indicators such as SSL channels. The VI indicator works by calculating volatility as the absolute difference between high and low prices, working backwards from the most current observation in the price series. Figure 26 demonstrates this by plotting this data as a bar chart with an SMA of volatility that is overlayed and used as a threshold to define high volatility. Furthermore, when the VI bars cross the SMA, high levels of volatility are signalled, as demonstrated frequently within the graph. Similarly, Figure 27 demonstrates this logic through pseudocode and captures the calculations behind volatility and the SMA.

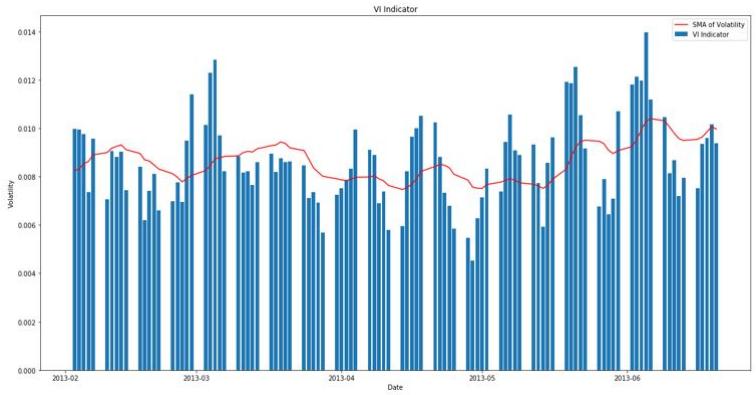


Figure 26: Volatility Index Indicator

Many additional unique indicators were implemented such as QQE and Linear WMA in a similar fashion. An important parameter for the calculations of most indicators was the period. The period defined the number of previous observations used to generate instances of each indicator. For example, a 5 period SMA would simply be the average of the previous 5 values in the series. Hence, different values for the period, as well as other parameters, could be used to generate multiple features from standalone indicators. Using this approach, the resultant DataFrame after technical analysis, shown in Figure 28, was highly dimensional with over 60 engineered features. The intuition behind generating multiple versions of each indicator was to allow for the tuning of technical analysis parameters during feature selection.

```
def volatility_index_indicator(df, period):
    volatility = DataFrame(columns=['VOLATILITY'], index=df.index)
    for i in reversed(range(0,len(df)-1)):
        if df.Close.iloc[i] > df.Open.iloc[i]:
            volatility.iloc[i] = abs(df.High.iloc[i] - df.Low.iloc[i+1])
        elif df.Close.iloc[i] < df.Open.iloc[i]:
            volatility.iloc[i] = abs(df.Low.iloc[i] - df.High.iloc[i+1])
        else:
            volatility.iloc[i] = volatility.iloc[i+1]
    volatility = SMA(volatility.VOLATILITY, timeperiod=5)
    sma_vol = SMA(volatility, timeperiod=15)
    return volatility, sma_vol
```

Figure 27: VI Indicator Code Snippet

Date	Open	High	Low	Close	Volume	Adj_Close	ATR	ADX	AROON_Down	AROON_Up	...	RSI10	RSI15	KAMA15
2011-02-10	1.00454	1.00489	0.99621	1.00217	119120.8795	1.005602	0.009501	25.433869	0.0	40.0	...	42.375267	43.229323	0.999384
2011-02-12	1.00083	1.00116	0.99806	0.99825	3685.7300	1.004841	0.009052	24.168516	90.0	30.0	...	38.104076	40.568553	0.999376
2011-02-13	0.99855	1.00753	0.99843	1.00307	106895.0516	1.004658	0.009071	22.301349	80.0	20.0	...	45.595885	45.026230	0.999416
2011-02-14	1.00307	1.00589	0.99447	0.99655	141942.1788	1.003824	0.009264	20.583584	100.0	10.0	...	38.577779	40.611491	0.999381
2011-02-15	0.99654	1.00571	0.99596	1.00367	173804.4501	1.003808	0.009304	19.037596	90.0	0.0	...	48.243778	46.723333	0.999487
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...

2060 rows × 63 columns

Figure 28: Sample data from DataFrame of Technical Indiactors (Technical Analysis)

A limitation of using multiple periods was the production of NaN values. These values were generated when the number of available observations before a datapoint was less than the size of the period. Records containing these NaN values were pruned from the dataset to ensure data richness, resulting in information loss. To limit this data loss, the range of periods used was bounded and applied to appropriate indicators. For the AUD/USD dataset, the number of data points remaining after technical analysis dropped from 2091 to 2060. A loss of this magnitude was insignificant as most of the time series was preserved, with loss only occurring at one end of the data. An alternative method for dealing with these values would be to use back/forward-filling, like how missing data was dealt with during data pre-processing. However, this approach could result in dirty data if the size of the periods used is sufficiently large.

Through leveraging technical analysis for feature generation, a comprehensive set of features were generated for the AUD/USD dataset. Many of the engineered indicators were correlated or summarized similar information and so multicollinearity became a problem. This is where the statistical significance of certain features is dampened by the effect of other highly correlated features [50]. Multicollinearity problems can be demonstrated through a simple regression problem in which the output variable Y is given by:

$$Y = \beta_1 X_1 + \beta_2 X_2 \tag{4.12}$$

where β_1 and β_2 are coefficients of features X_1 and X_2. If X_1 and X_2 are highly correlated, it becomes difficult to distinguish the effect of X_1 on Y from the effect of X_2 on Y as changing the value of X_1 equivalently changes X_2. Similarly, as the indicators outputted values of different magnitudes, there was a bias in which statistically larger features had more influence over prediction. Furthermore, it became imperative to employ feature selection techniques for dimensionality reduction. Similarly, it was important to enforce alternative feature generation techniques in a hybrid model to collect further abstractions and information about the dataset.

4.4.2 Sentiment Analysis

Sentiment analysis pertains to the study of the opinions of retail traders towards the forex market, often through the processing on online social media. It was previously shown how sentiment analysis had been successfully implemented as a feature generation technique within the stock market, however, was generally unexplored for forex. This is fascinating considering the vast amount of social media data available as well as the interesting relationship between exchange rates and trader sentiment demonstrated in Figure 7. Thus, this section provides a novel approach to Sentiment analysis for feature engineering within the forex market.

Currently, there exists no freely available trader sentiment datasets that can be downloaded over large periods of time. This meant that a new solution had to be proposed in which social media was leveraged and scraped to build bespoke DataFrames. The most accessible and well-documented social media was Twitter, which provided posting data from a multitude of profiles dating back to 2006. Initially, there were many Python libraries that allowed the scraping of historical tweets by leveraging the Twitter API. However, upon further research, many of these libraries were deprecated and no longer operated due to updates within the API. Moreover, the only functioning tool was found to be the Python library "snscrape", which enabled the scraping of general social media and filtering through search queries. Although snscrape provided convenient web scraping automation, the tool was highly undocumented and required extensive testing

to understand its behaviour.

Figure 29 depicts the record structure of pulled tweets pertaining to the GBP/JPY pair which observe the following structure: Date, Tweet ID, Tweet Text, and Username. These fields were searchable and so were optimized within scrape queries to filter and clean the acquired data. The initial plan for acquiring a complete dataset was to pull the top 100 tweets concerning a given currency pair for each day. This would then be aggregated to find a quantitative daily mean polarity score/value. Unfortunately, this became quickly insufficient due to the high noise ratio within Twitter posts, with most tweets originating from bot/spam accounts. To combat this, an optimized list of verified retail Forex trader accounts was collated for use within search queries. The list was obtained by analysing high profile Forex accounts with frequent posting schedules and is a variable that can be optimized. Furthermore, the full list of twitter accounts used can be found in Appendix A. Alternatively, logic could have been employed such that accounts could only be used once within the dataset, however, this would result in exposure to large proportions of unverified profiles.

	Datetime	Tweet Id	Text	Username
0	2013-01-04 22:58:43+00:00	287332257398681600	PSlope for GBP/JPY on m15: A=141.763,B=141.619...	fxtradersystem
1	2013-01-04 06:51:59+00:00	287088973468291072	GBP/JPY Daily Outlook: With 4 hours MACD stayi...	EuropeanMarkets
2	2013-01-04 06:34:55+00:00	287084676806479872	GBP/JPY Daily Outlook With 4 hours MACD stayin...	LATEST_FX_NEWS
3	2013-01-04 06:25:46+00:00	287082374121013248	GBP/JPY Daily Outlook: With 4 hours MACD stayi...	COLLICAFeed
4	2013-01-04 05:59:40+00:00	287075807623532544	GBP/JPY Daily Outlook: With 4 hours MACD stayi...	ForexOnestop
5	2013-01-04 05:30:03+00:00	287068352558555136	Forex GBPJPY stalls an 8 week run up below 141...	MarketInvestors
6	2013-01-04 05:14:55+00:00	287064543866273792	GBP/JPY Daily Outlook: With 4 hours MACD stayi...	tomfindlay1976
7	2013-01-04 05:14:54+00:00	287064542054326273	GBP/JPY Daily Outlook: With 4 hours MACD stayi...	ultimateforex
8	2013-01-04 05:14:54+00:00	287064538426273792	GBP/JPY Daily Outlook: With 4 hours MACD stayi...	rightonthemoney

Figure 29: Sample Scraped Tweets using snscrape library

A fundamental issue with the use of a static list of accounts was the assumption that traders tweeted every day. This was in fact false and resulted in only 40% of data between 2011-2019 for the AUD/USD dataset being account for by tweets. Consequently, the acquired DataFrame was sparse and methods such as interpolation for data filling were inefficient and unreliable. To solve this problem, a layered approach was adopted

in which tweets were first scraped from the list of reputable accounts that satisfied a search query. In the case that there were no tweets, the top 100 tweets for that day were pulled from twitter, constraining accounts to a singular tweet contribution per day. Algorithm 1 demonstrates this hybrid approach to generate the raw sentiment data shown in Figure 30 for the AUD/USD dataset between 2011 and 2019.

During the scraping of Twitter posts, several issues were realised about the snscrape library. The foremost problem was the time complexity in which it would take an average of 7.6 seconds to fulfil a single day of tweet scraping, equivalent to a single datapoint in the AUD/USD dataset. Performing this between 2011-2019 required 2920 days of scraping which took approximately 3.8 hours to complete. Although this was inconvenient, it was not unreasonable, however, when performing pull requests, the library started to throw exceptions. It became evident that the Twitter API would block requests being made after an average of 130 days' worth of successive requests within a short period. After such exceptions, a 15-minute cooldown period would be imposed before scraping could continue. The inability to automate this process due to sporadic and undetectable exceptions meant that acquiring data took almost two days to fulfil. As a result, the DataFrame of tweets shown in Figure 30 was the final dataset for the process and any optimizations that could be applied to the search query or hyperparameters would not be possible. This is something to be considered when evaluating the accuracy of derived sentiments and the performance of engineered features.

	Date	ID	Tweet	User
0	2018-12-31 14:18:13+00:00	1079743531382722560	The AUD/USD is driving me crazy, $USD goes dow...	petergo99037185
1	2018-12-30 22:44:42+00:00	1079508604124721152	AUSTRALIA currency policy of minus 25% to USD ...	ArthurKokontis
2	2018-12-29 23:51:39+00:00	1079163064933470208	The AUD/USD exchange rate is about to make a r...	wijaranakula
3	2018-12-29 17:00:13+00:00	1079059527289442304	AUD/USD is perilously close to the October low...	breaksnrevs
4	2018-12-28 11:46:46+00:00	1078618254816497664	AUD/USD .7064!!! So everything bought in the ...	bodgejob2
...	...	...	...	...
4616	2011-01-04 17:50:05+00:00	22349059058565120	AUD/USD is the biggest loser so far in 2011. ...	GaryDeduke
4617	2011-01-04 17:14:30+00:00	22340105045807104	FX Levels and Outlooks - USD/CAD AUD/USD Gold:...	boposlav
4618	2011-01-04 06:26:57+00:00	22177143656022016	I think a sell of aud/usd pair with TP at 0.99...	ashwinksharma
4619	2011-01-04 04:49:26+00:00	22152602259034112	Not sure the AUD/USD is readyto climb yet	Poppyjerry
4620	2011-01-03 17:14:26+00:00	21977700684005376	My trade room is long AUD/USD from 1.0180. Ema...	NewstraderFX

4621 rows × 4 columns

Figure 30: Twitter Sentiment Data for AUD/USD dataset between 2011 and 2019

To obtain a more comprehensive and diversified understanding of trader sentiment, Reddit was also scraped as an alternative social media. Like Twitter, Reddit has open-source API's that allow for the pulling of posts from subreddits and active querying as well as supplying several supporting libraries. The Python Reddit API Wrapper (PRAW) was the main library used for searching and pulling posts via unique posts IDs as well as providing additional statistics such as upvote ratios and post scores. These statistics were used to apply pre-processing steps to post data by pruning those that had an upvote ratio/score below a threshold. Thus, allowing for a cleaner and more reliable dataset at the cost of losing information. The list of subreddits traversed included those that related to the Forex market and general economics such as r/Forex and r/Economics. Similarly, a list of keywords was inputted as a parameter to the search query to ensure relevant posts were pulled.

A limitation of PRAW was the inability to search for posts within a data range. This was a problem as sentiment data had to be acquired between 2011-2019 to match the records found within the AUD/USD dataset. To resolve this, the Pushshift.io open API was used which allowed for date querying of reddit posts and conveniently returned the unique post ID's that satisfied the query. Moreover, the Pushshift API was used first to

acquire a list of ID's that were then passed through PRAW in which all corresponding posts and statistics were received. Furthermore, Figure 31 depicts the records of data received from Reddit which observed the following structure: Date, Post Title, Content, Post Score, Upvote Ratio. 6245 records of data were collected from Reddit for the 2011-2019 period in approximately 35 minutes. This was significantly faster than Twitter scraping due to cleaner API's, however, was limited in the number of sources used. Furthermore, by acquiring a range of statistics as well as post titles, a range of features could be engineered through Sentiment analysis.

Algorithm 1 Tweet Scraping Algorithm

```
0: procedure GetTweets
0:     accounts ← list of verified twitter accounts
0:     tweet_list ← structure to hold pulled tweets
0:     users ← list of users pulled from
0:     for acc ∈ accounts do
0:         query ← initialise query with acc
0:         pulled ← PullTweets(query)
0:         for tweet ∈ pulled do
0:             Add tweet to tweet_list
0:     noDate ← list of dates with no tweets
0:     for date ∈ noDate do
0:         query ← initialise query with dates
0:         pulled ← PullTweets(query)
0:         for tweet ∈ pulled do
0:             if tweet.user ∈ users then
0:                 continue
0:             Add tweet to tweet_list
0:             Add user to users
0:     return tweet_list
=0
```

After acquiring the initial datasets from both Twitter and Reddit, cleaning was performed to remove noise, extremities and ensure text could be parsed by a sentiment derivation model. The cleansing of data required the reformatting of text to remove special characters, punctuation, and social media dependant syntax. For example, for both tweets and reddit posts, hashtags were removed as well as tagged users, basic punctuation, special characters, and links.

	Date	Titles	Content	Score	Upvote Ratio
0	2011-01-01	Bill Gross Telling Bloomberg To "Avoid Dollar ...		1	0.54
1	2011-01-02	Clueless teenager here: Is outsourcing from on...	I know there is enough debate to get a million...	34	0.76
2	2011-01-04	China's Grey Swan is changing colors; Hyperinf...	[removed]	2	0.75
3	2011-01-05	Dollar Fade - World Bank issues its 1st yuan b...		36	0.72
4	2011-01-06	2011 Market Outlook	At [FXIB](http://www.fxibonline.com) we have i...	0	0.50
...	...	...	...	...	...
6240	2018-12-29	The Dollar Store Backlash Has Begun		72	0.82
6241	2018-12-31	Is Dollar a Sisyphus?	[deleted]	1	1.00
6242	2018-12-31	Is Dollar a Sisyphus?	[removed]	1	1.00
6243	2018-12-31	U.S. dollar share of global currency reserves ...	[deleted]	43	0.85
6244	2018-12-31	2019 - The Year of the "Human Spring"... It is...		0	0.46

6245 rows × 5 columns

Figure 31: Reddit Sentiment Data for AUD/USD dataset between 2011 and 2019

There are two classical methods for deriving sentiment scores from bodies of text, namely rule-based systems, and machine learning. Rule-based systems enrol sets of hand-crafted rules that attempt to identify the opinions, context, and polarity of individual words. These approaches can often utilize Natural Language Processing (NLP) techniques such as tokenization and tagging. An example of a rule-based approach to sentiment analysis includes the use of a lexicon that contains a list of words and their polarizations. The number of positive and negative words within a given body of text are counted and the greater of the two is returned as derived sentiment. Clearly, this method is naïve as it is unable to understand the context of words and their combined meanings, however, provides a simple and cheap method for sentiment analysis. Such an approach can be made more robust through the addition of more rules but comes at the cost of increased complexity as the system becomes harder to maintain and tune.

Machine learning is the second archetypal method for sentiment analysis in which automation is provided. In this method, a classification problem is approached where text inputs are categorised into sentiment classes such as positive, negative, and neutral. Unlike rule-based systems, machine learning does not require hand-crafted rulesets but instead extracts features from input text and learns patterns within data to attain autonomy. Machine learning models prefer structured and meaningful data, therefore

feature extraction is used to acquire this through a quantitative representation of input text. This process is called text vectorization and is typically performed through techniques such as bags-of-words or bags-of-ngrams [51]. In these approaches, the order and structure of words are abstracted with the intuition that similar input data will have similar contexts, therefore conclusions can be drawn from words alone.

In this project, the Python libraries TextBlob and Vader Sentiment were used for processing textual data and completing NLP tasks. These libraries implemented rule-based approaches to sentiment analysis and satisfied the following process:

1. Tokenization of input text
2. Tagging and Identification of each token e.g., noun, verb etc.
3. Polarity assignment
4. Aggregation for calculating final sentiment over all tokens.

Similarly, both libraries were built on top of or within the Python Natural Language Toolkit (NLTK) and provided clean APIs for a range of functions. Although both libraries functioned in similar ways, the Vader Sentiment module was more widely adopted due to its dynamic lexicon and ease of tuning. This was pivotal as both models struggled to initially recognise financial jargon such as "bearish" and "bullish", which are commonplace within the forex market. Furthermore, it has been suggested that Vader Sentiment performs better than TextBlob when analysing slang words [52], making it more suitable for financial contexts. An example of using the Vader Sentiment model to derive scores from news headlines can be seen in Figure 32. The model provided an overview of polarity in terms of a negative, neutral, positive, and compounded score. It can be observed that the unoptimized model was unable to capture the context of the Forex news headline as it derived a positive score for the falling of the EUR/JPY currency pair. To solve this problem, the lexicon of the model was manually updated to include basic Forex syntax as well as corresponding sentiment scores. This

demonstrates a fundamental limitation of such rule-based systems as manual tuning is required to befit the application domain.

```
HEADLINE USED: EURJPY Price May Fall as US PCE Data Encourages Dollar Higher
VADER SENTIMENT RETURN: {'neg': 0.0, 'neu': 0.775, 'pos': 0.225, 'compound': 0.4404}
```

Figure 32: Vader Sentiment model used to derive sentiment for example news headline for EUR/JPY currency pair

Although Vader Sentiment was used for sentiment score derivation, the TextBlob library provides a range of useful NLP methods that can be used for further analysis. This is because TextBlob is built on top of the Python NLTK, allowing for easy access to techniques such as n-grams, speech tagging and tokenization. These approaches allow for the transformation of textual data as an initial step towards feature extraction. By performing these transformations, the TextBlob library can be used to easily prepare text inputs for machine learning sentiment classification. This approach for sentiment analysis was not fully explored as it expanded the scope of this research and so remains to be an area of potential study.

Once sentiment scores had been derived for each observation within the sentiment dataset, values were aggregated, and the mean was calculated for each day in the series. This way, an average sentiment was acquired for each day corresponding to the AUD/USD price series. In addition to this, various alternative features were generated such as sentiment return, moving averages of sentiment and lagged sentiment. By producing alternative abstractions of daily sentiment, more information was gathered about the social media data with the hopes of achieving greater predictive power. Figure 33 shows the result of this sentiment analysis process and the various extracted features within a single DataFrame.

Reddit Sentiment	Twitter Sentiment 2	Reddit Sentiment 2	News Sentiment 2	Twitter Sentiment Return	Reddit Sentiment Return	News Sentiment Return	Twitter Sentiment MA	Reddit Sentiment MA	News Sentiment MA
0.000000	-0.361200	0.790600	0.075496	0.179621	0.000000	0.030991	-0.435250	-0.158120	-0.066623
0.790600	0.440400	-0.152933	0.171122	-2.219269	-1.193440	1.266631	-0.238710	-0.083293	0.004306
-0.152933	0.452500	-0.093375	0.047328	0.027475	-0.389440	-0.723427	-0.039750	0.003445	0.050475
-0.093375	0.464600	-0.033817	-0.076467	0.026740	-0.637840	-2.615683	0.138020	0.102095	0.058141
-0.033817	0.476700	0.025742	-0.181353	0.026044	-1.761212	1.371660	0.294600	0.107243	0.007225
...	...	...	...	...	...	...	...	...	...

Figure 33: Sentiment Features for Reddit posts and Twitter data pertaining to AUD/USD dataset between 2011-2019

To evaluate and motivate the use of sentiment analysis as a feature generation technique for forex trend prediction, the derived sentiment data was compared to the actual returns of the AUD/USD price series. Standardisation was applied to both series through a Standard Scaler to achieve distributions with a standard deviation equal to one 1 and a mean value of 0. By scaling values between 0 and 1, the datasets could be plotted together and analysed synchronously. Figure 34 illustrates this through a plot of actual returns (blue) and twitter sentiment (red). A relationship can be observed between twitter sentiment and actual returns as they follow similar trends with a correlation of 0.4. Note that the magnitude of the sentiments does not accurately represent the magnitude of the returns. Rather, they both seem to capture the same general trends within the market. Furthermore, the proposed approach for sentiment analysis has been proven to encapsulate trends within the forex market and produces beneficial features for the classification problem.

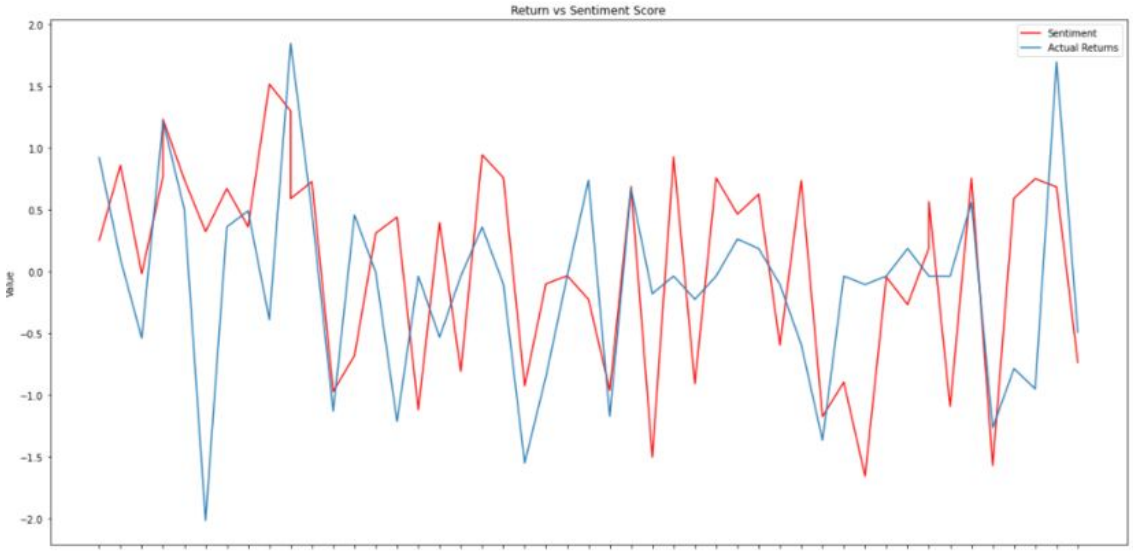


Figure 34: Actual Returns vs Twitter Sentiment for AUD/USD dataset between 2011-2019

4.4.3 Fundamental Analysis

The final technique for feature generation used within the project was Fundamental Analysis. This approach looks at the social, economic, and political influences on the Forex market such as the employment rates, financial statements, and GDP of a country. The work of Eng et al [18] supports the use of fundamental data for financial market prediction and motivates its use as tool for feature generation.

Like with sentiment analysis, there existed no readily available fundamental datasets that covered substantial periods. To further this, relevant fundamental data pertaining to specific countries or currency pairs was also unavailable. Moreover, the clearest and most reliable medium for acquiring fundamental data was news headlines. This was because they could easily be filtered to satisfy search queries relating to specific countries or currency pairs. The initial idea for acquiring such news headlines was through the scraping of RSS feeds. Unfortunately, it was quickly realised that standard RSS feeds were dynamically updated and so archived xml files could not be acquired, limiting data to a single page of articles. Consequently, the only feasible method for acquiring historic news articles was through manual web scraping. This required the

heavy processing and careful parsing of individual webpages to build a sufficiently large DataFrame. The website investing.com was used as the primary source of fundamental data due to its convenient indexing, allowing for a simplified scraping algorithm.

To perform manual web scraping, HTTP requests were made to each page of the Forex news section of the investing.com website. Pages were indexed by number and so were iteratively requested and processed until articles with a date matching the beginning and end of a search query were found. The raw HTML code for each article was then parsed to extract headlines and filter out noise such as sponsors or ads. The Python library BeautifulSoup made this possible by providing simple HTML parsing methods such as tag filtering and searching. It can be noted that headlines can be filtered such that only those that relate to specific countries or currencies are retained. Moreover, news headlines were scraped for the AUD/USD dataset between 2011-2019 and can be seen in Figure 35. The initial dataset pulled included general Forex news and so a search query was enrolled using a list of keywords as a filtering step. Specifically, the list of keywords contained the following: USD, AUD, Australia, USA, US Dollar and AUD/USD. This list was non expansive and so can continue to be optimized, however, provided a sufficient approximation of relevant fundamental data, as seen by the 18427 records remaining in Figure 35.

After attaining a fundamental dataset, daily sentiment scores were derived through aggregation, identical to the process used in Sentiment Analysis. This involved the use of the Vader Sentiment model which produced a series of mean sentiments shown in Figure 35. It can be observed that only 2003 records remained after aggregation which is less than the number of days in the 2011-2019 period. During sentiment analysis, a layered approach was adopted to ensure data was available for each day of searching. This was not the case for Fundamental Analysis due to the constraints of a single news outlet. To solve this, various alternative new sources could have been used, though this would require bespoke scraping algorithms for each site due to differences within HTML structures. Furthermore, due to time constraints, interpolation was used to fill

the missing values 920 values. This remains to be a suboptimal solution and can be improved upon in future work.

Date	Headline
2011-01-02	Forex - AUD/USD weekly outlook: January 3-7
2011-01-02	Forex - USD/CAD down during the Asian session
2011-01-02	Forex - USD/JPY up during the Asian session
2011-01-02	Forex - AUD/USD down during the Asian session
2011-01-02	Forex - NZD/USD down during the Asian session
...	...
2018-07-05	Forex - EUR/USD Strength Knocks Dollar; Fed Mi...
2018-07-09	Forex - Dollar Turns Positive After GBP/USD Sl...
2018-07-13	Forex - Dollar Retreats From 2-Week Highs as G...
2018-07-27	Speculators lift net long USD bets, highest si...
2018-12-30	USD Flat In Asia As US-China Trade Talks Prog...

18427 rows × 1 columns

Date	Sentiment
2011-01-02	-0.083418
2011-01-03	-0.256967
2011-01-04	0.010917
2011-01-05	-0.144652
2011-01-06	-0.005096
...	...
2018-07-05	0.493900
2018-07-09	0.557400
2018-07-13	0.790600
2018-07-27	0.000000
2018-12-30	0.421500

2003 rows × 1 columns

Figure 35: Scraped News Headlines from investing.com relating to the AUD/USD pair between 2011 and 2019 and derived sentiment scores

The effectiveness of the Vader Sentiment model, after lexicon tuning, can be demonstrated through word clouds. These visualisations illustrate the frequency of certain words within the classes generated by a model. Figure 36 shows the word clouds for the positive and negative classes generated by the Vader Sentiment model when classifying the sentiment of the scraped news articles in Figure 35. It can be observed that within the positive classed news articles, Forex specific syntax was captured such as higher, rises, and gain. Similarly, for the negative class, jargon such as slips, lower and slump were realised. This shows how the rule-based model was able to adapt to the changes within its lexicon and effectively classify news articles. Figure 37 supports this claim when calculating the classification accuracy of the model on 100 pre-labelled news articles in which an 88% accuracy was achieved. Thus, the Vader Sentiment model presents a simple and effective method for sentiment derivation within Fundamental and Sentiment analysis for feature generation.

Figure 36: Positive and Negative word clouds for classified news sentiment - showing frequency of words within each class.

A range of features were also generated from fundamental data sentiment. This included lagged news sentiment, sentiment return and moving averages. Figure 33 shows these features after the sentiment and fundamental analysis process. In conclusion, Fundamental Analysis was utilized to capture deeper analysis and abstractions of Forex data to aid the holistic approach adopted for trend prediction. By capturing factors external to the raw price series, a more comprehensive set of variables were attained.

	precision	recall	f1-score	support
-1	0.88	0.82	0.85	17
1	0.88	0.91	0.89	23
accuracy			0.88	40
macro avg	0.88	0.87	0.87	40
weighted avg	0.88	0.88	0.87	40

Figure 37: Classification report for Vader sentiment model when tested on 100 manually labelled news articles

4.4.4 Final Dataset

Feature generation was conducted through Technical, Sentiment and Fundamental Analysis. This resulted in a vast number of engineered features within a finalised DataFrame. Figure 38 depicts a sample from the final DataFrame in which 75 features were generated. Each feature captured a different aspect of the AUD/USD currency pair and produced values with differing distributions. Furthermore, additional processing and data management was required to ensure features were not undermined due to the statistical significance of others.

date	Open	High	Low	Close	Volume	Adj_Close	ATR	ADX	AROON_Down	AROON_Up	...	Reddit Sentiment	Twitter Sentiment 2
2011-02-16	-2.059794	1.271955	-1.496021	2.261647	145581.8402	0.220081	-0.003171	18.583211	80.0	20.0	...	0.000000	-0.361200
2011-02-17	2.255490	0.218701	3.491674	0.221342	127347.1819	0.220086	0.004869	18.817948	70.0	10.0	...	0.790600	0.440400
2011-02-19	0.137879	-0.098105	0.421135	-0.041208	2514.3501	0.150574	0.014352	19.029211	60.0	0.0	...	-0.152933	0.452500
2011-02-20	0.040588	0.069997	-0.417820	-0.321397	84486.1273	0.046669	0.001557	17.416606	50.0	80.0	...	-0.093375	0.464600
2011-02-21	-0.330830	-0.359630	-1.426833	-1.106428	186337.6789	-0.116143	-0.008484	17.918501	40.0	70.0	...	-0.033817	0.476700
...	...	...	...	...	...	...	...	...	...	...	...	...	...

Figure 38: Sample of records from final AUD/USD dataset from 2011-2019 after feature generation

4.5 Stationarity of Time Series Data

4.5.1 Stationarity & Integer Differencing

Datapoints within a time series have temporal dependence and seasonal components, making it hard to identify underlying trends and forecast movements. This is because each observation exhibits properties dependent on the attributes of previous data. Moreover, stationarity ensures time invariant statistical properties such that the mean, variance, and autocorrelation of a series are constant over time. By enforcing consistency, the modelling and analysis of the classification problem is simplified as external components are removed from the raw price series. Figure 39 shows an example of a stationary process such that statistical consistency is observable.

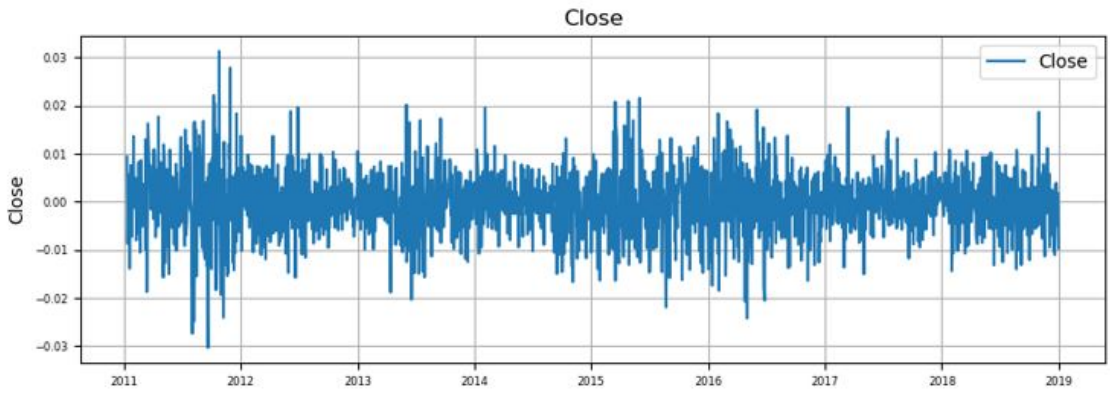


Figure 39: Closing price of AUD/USD made stationary through integer differencing

The most popular method for achieving stationarity is integer differencing. This is

where values are calculated by subtracting a lagged series of order p [53]. For example, first order differencing is where values are subtracted by their immediately preceding observations, also known as the *lag-1* difference. The first order differenced series y_t^1 is given by:

$$y_t^1 = y_t - y_{t-1} \tag{4.13}$$

where the differenced series has $t-1$ values as the first observation cannot be differenced. If the series is still non-stationary after this process, second order differencing can be applied, increasing the order used. The second order differenced series y_t^2 is subsequently given by:

$$y_t^2 = y_t^1 - y_{t-1}^1 = (y_t - y_{t-1}) - (y_{t-1} - y_{t-2}) = y_t - 2 * y_{t-1} + y_{t-2} \tag{4.14}$$

Note that the new differenced series y_t^2 has $t-2$ values. Hence, the series y_t^p, differenced by the order p, is given by $y_t^p = y_t^{p-1} - y_{t-1}^{p-1}$ and retains $t-p$ values. Furthermore, differencing can be iteratively applied until stationarity of a series is achieved.

Integer Differencing is a simple and quick method for attaining stationarity and is often used on price series data. A more robust use of differencing is on log prices. This is where prices are defined as the log values of their raw price. By differencing log prices, the differencing function can obtain log returns, which are defined as follows:

$$log\,(1 + r_t) = log\left(\frac{p_t}{P_{t-n}}\right) = log\,(p_t) - log\,(p_{t-n}) \tag{4.15}$$

Returns are desired as they apply normalisation to prices, allowing for comparable metrics between variables. Log returns provide even further benefits such as raw-log equality. This is where, if returns are small, which is common within the Forex market due to small daily price deviations, the log returns are approximately equal to raw returns such that [54]:

$$log\,(1 + r) \approx r,\ where\ r \ll 1 \tag{4.16}$$

Similarly, a frequently calculated statistic for price series data is the compounded return. This is where returns are aggregated across a period to find the cumulative return in the formula:

$$(1 + r_1)(1 + r_2)\dots(1 + r_n) = \prod_i (1 + r_i) \tag{4.17}$$

Performing this over raw percentage returns is problematic as the product of two normally distributed variables is not necessarily normal [55]. However, the sum of two normally distributed variables is normal [56], when they are both uncorrelated. The time-additive property of log returns provides a solution for this as it can be compounded through a simple summation as follows:

$$\sum_i log(1 + r_i) = log(1 + r_i) + log(1 + r_2) + \dots + log(1 + r_n) = log(p_n) - log(p_0) \tag{4.18}$$

Notice that this summation can be simplified to a differencing operation between the final and initial observations. Hence, integer differencing can attain log returns and cumulative log returns whilst ensuring beneficial statistical properties. This reduces the time complexity for computing compounded returns and presents a flexible approach to stationarity.

4.5.2 Preserving memory through Fractional Differencing

Although log returns are sufficient for attaining stationarity of time series data, there are various limitations of this approach. Prado [57] suggests that returns are memory-less and that raw prices have memory in their non-stationary and trending form. This presents a fundamental trade-off between stationarity and memory retention as statistical consistency is sought for at the loss of information. Furthermore, the problem becomes "What is the minimum amount of differentiation that makes a price series stationary while preserving as much memory as possible?" [57].

Prado [57] presents a remedy for this problem being Fractional Differencing. This extends differencing to fractional numbers, of order d, using fractional calculus. The intuition is that each value is a weighted sum of previous values in the time series, where the new stationary series X_t is given by:

$$X_t = \sum_{k=o}^{\infty} w_k X_{t-k} \tag{4.19}$$

where weights w are calculated as fractional derivatives and follow the expansion:

$$w = \left\{1, -d, \frac{d(d-1)}{2!}, -\frac{d(d-1)(d-2)}{3!}, ..., (-1^k)\prod_{i=0}^{k-1}\frac{d-1}{k!}, ...\right\} \tag{4.20}$$

And values for the time series X are given by:

$$X = X_t, X_{t-1}, X_{t-2}, \ldots, X_{t-k}, \ldots \tag{4.21}$$

It can be observed that when d is a positive integer the value of $\left(-1^k\right)\prod_{i=0}^{k-1}\frac{d-i}{k!} = 0$ for $\forall k > d$. This means that all values beyond $k = d$ are removed from the series and memory is said to be lost since $k - d = 0$. For example, for integer differencing with $d = 1$, the calculated weights are given by $w = \{1, -1, 0, 0, \ldots\}$, which is identical to returns as the values for $X = X_t, -X_{t-1}$ where only the current and immediately preceding values are used.

Note that when d becomes a real number, such as $d = 0.2$, as k remains to be an integer, the calculated weights for w will always be some value $\neq 0$. Thus, allowing memory to be preserved as weights are applied to each observation in the series. Furthermore, the weights given to the series can be expressed iteratively in the following form:

$$w_k = -w_{k-1} * \frac{(d-k+1)}{k} \tag{4.22}$$

Intuitively, calculating weights for every past observation in a large series can become computationally expensive as each datapoint is transformed by a fractional derivative.

Figure 40 demonstrates this by plotting the weights given by fractional differencing on the AUD/USD dataset with an order of $d = 0.1$. When $k > d$, the calculated weights converge asymptotically to 0, as the absolute values for weights decrease. Moreover, a threshold $\tau \in [0, 1]$ can be employed as a tolerance level such that calculations are terminated when weights are below the threshold. This bounds the length of memory retained and is a good approximation of the original series information.

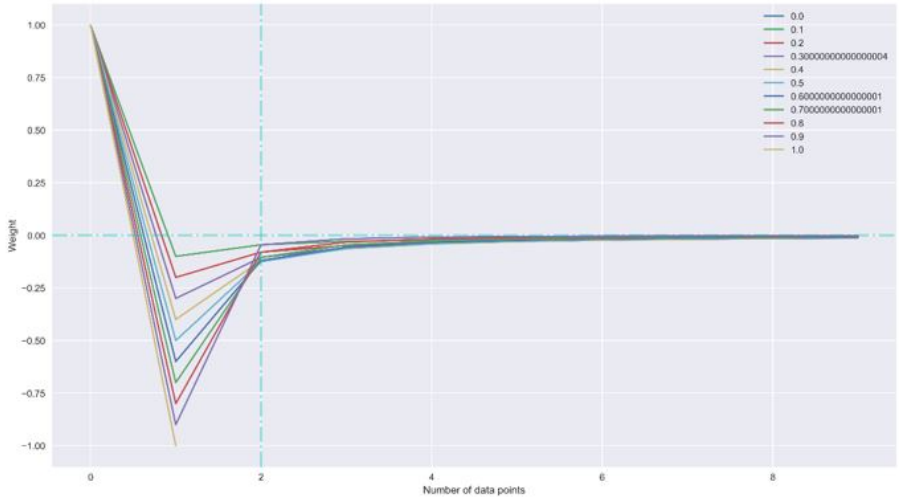


Figure 40: Convergence of Fractional Differencing weights on AUD/USD closing price series

To test for stationarity after differencing a series, statistical tests such as the Augmented Dickey-Fuller (ADF) can be employed. The ADF test is a unit root test in which the null hypothesis that a time series contains a unit root is tested. If the null hypothesis is accepted, then the series is not stationary and has a time dependent structure. Whereas, if it is rejected, then the time series is regarded as stationary and time independent. This test can be performed in Python via the statsmodels adfuller module in which a range of values are returned. Specifically, the ADF returns p-values which can be compared to an accepted threshold such as 5% (0.05). If the returned p-value is less that the threshold, the null hypothesis can be rejected. Similarly, the test returns an ADF statistic which can be interpreted by comparing it to given critical values. If the ADF statistic is less than the critical values, then the null hypothesis of a unit root can also be rejected.

Figure 41 depicts the results of the Augmented Dickey-Fuller test on the AUD/USD dataset for a range of fractional orders of d. The correlation of the resulting series to the original data was also calculated. Using these results, the order d used for fractional differencing can be optimized by searching for the highest correlation scores whilst attaining stationarity. When $d = 0.2$, the ADF statistic of -3.18 was strictly less that the 5% critical value of -2.86. The returned p-value (≈ 0.02) was also less that the 5% threshold. Thus, the null hypothesis of a unit root was rejected, and stationarity was assumed. Furthermore, the correlation with the original time series was 95% and so long-memory was attained whilst achieving stationarity.

Through this process, the optimal value for d was found and a final transform was performed on the data through fractional differencing. Although this allowed for the retention of memory, it was soon realised that the fractional differencing process came at the cost of data. This was because, when using a threshold to limit computation of weighted lags, the number of past values required to calculate a new observation was bounded to n. Thus, n-1 were removed from the beginning of the dataset. When the threshold value τ was small, the number of pruned records increased as more fractional derivatives were computed. This was a problem as the daily Forex datasets were limited to 8 years of past data and using a threshold of $\tau = 0.0001$ and $d = 0.2$ resulted in over 500 datapoints being pruned. Moreover, a second trade-off was realised between maximising the size of the time series and retaining memory through sufficient fractional differencing.

	adfStat	pVal	lags	N	5% Crit Val	Corr w/ OG
0.0	-0.836410	8.081922e-01	1.0	2711.0	-2.862607	0.998954
0.1	-1.947127	3.102261e-01	1.0	2209.0	-2.862849	0.989840
0.2	-3.177869	2.130198e-02	1.0	2215.0	-2.862846	0.950898
0.3	-5.059568	1.689507e-05	1.0	2324.0	-2.862784	0.886472
0.4	-7.936776	3.396946e-12	1.0	2430.0	-2.862730	0.804898
0.5	-12.551777	2.197900e-23	1.0	2512.0	-2.862691	0.656496
0.6	-17.535512	4.215334e-30	1.0	2572.0	-2.862664	0.500277
0.7	-22.587494	0.000000e+00	1.0	2615.0	-2.862646	0.352444
0.8	-27.395402	0.000000e+00	1.0	2648.0	-2.862632	0.232637
0.9	-31.696534	0.000000e+00	1.0	2674.0	-2.862621	0.138168
1.0	-35.877600	0.000000e+00	1.0	2710.0	-2.862607	0.023102

Figure 41: ADF test of AUD/USD dataset

In conclusion, Fractional differencing was used as alternative method for attaining stationarity that remedied the shortcomings of classical integer differencing. Despite the addition of memory retention, the complexity of applying fractional differencing, optimizing values, and dealing with trade-offs made the technique unattractive for rapid analysis. Similarly, there was no clear indication that preserving such memory could improve the performance of machine learning models. Therefore, the rest of the project utilized standard integer differencing and log returns to attain stationarity, however, an evaluation of the performance impact of fractional differencing remains to be an area of potential exploration.

4.6 Feature Selection & Dimensionality Reduction

Feature selection involves the dimensionality reduction of input matrices by selecting the strongest features for prediction and pruning the rest. This process can help improve the runtime of predictive models as the computational cost of processing high dimensional feature vectors is reduced. Similarly, the results of Baasher and Fakhr [21] suggest that selecting the best performing features can improve the performance of machine learning models when forecasting trends in the Forex market. This is because model performance can deteriorate if irrelevant features are used and so feature selection

can retain the most relevant variables. Moreover, as approximately 75 features were engineered for the AUD/USD dataset, feature selection was applied to secure these benefits.

4.6.1 Manual feature selection

The most basic type of feature selection method is manual selection. This is where the statistical properties of all features are analysed, and variables are removed if deemed irrelevant. An example of such an approach is through heatmap visualisations. These plots illustrate the cross correlation between input variables and can identify multicollinearity within the data. As discussed previously, multicollinearity can cause problems for regression models as variable coefficients become highly correlated and so such plots can be leveraged to solve this. The libraries matplotlib and seaborn were used as graph plotting tools to create such visualisations. Figure 42 shows the cross correlation heatmap for the AUD/USD dataset after feature generation. Initially, it is easy to see the high correlation areas between input variables in the top left (light) region of the plot. To further understand which features were most correlated, a naïve approach was enforced to remove and record all input variable pairs that had a correlation above a threshold. It was discovered that the majority of eliminated variables were technical analysis features. This was to be expected considering technical analysis captured patterns within mutual data which can easily overlap. Consequently, a total of 37 columns were removed in which all sentiment and fundamental analysis features were retained.

Another plot that was used for initial feature selection analysis was the input-output variable correlation plot. Intuitively, this plotted and ranked the correlation between features and output variables such as daily price returns. Figure 43 depicts this graph for the AUD/USD dataset. It was found that Twitter sentiment return, and lagged Twitter sentiment had the strongest correlation with price returns, however, this correlation was almost negligible at ≈ 0.07. On the other hand, when plotting feature correlation with market volume, the Twitter sentiment return was also one of the highest ranked features with a correlation of ≈ -0.1 and was ranked immediately after the volatility

indicators such as ATR and VOLA. These results are interesting as they support the conclusions made after sentiment analysis, reinforcing the use of social media indicators as predictive variables.

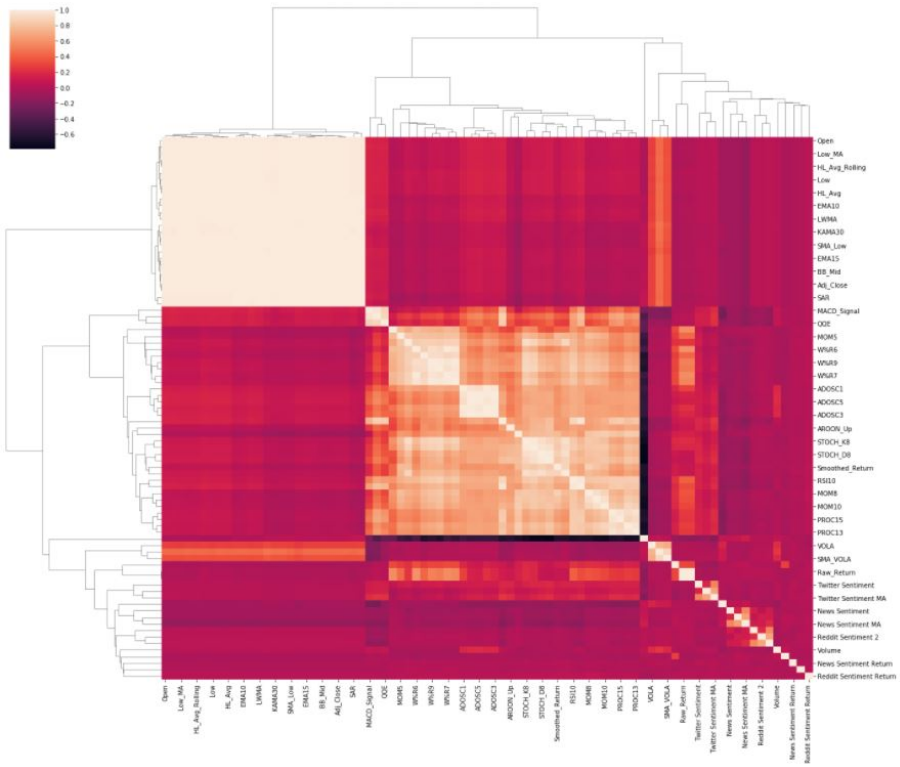


Figure 42: Feature cross correlation heatmap for AUD/USD dataset

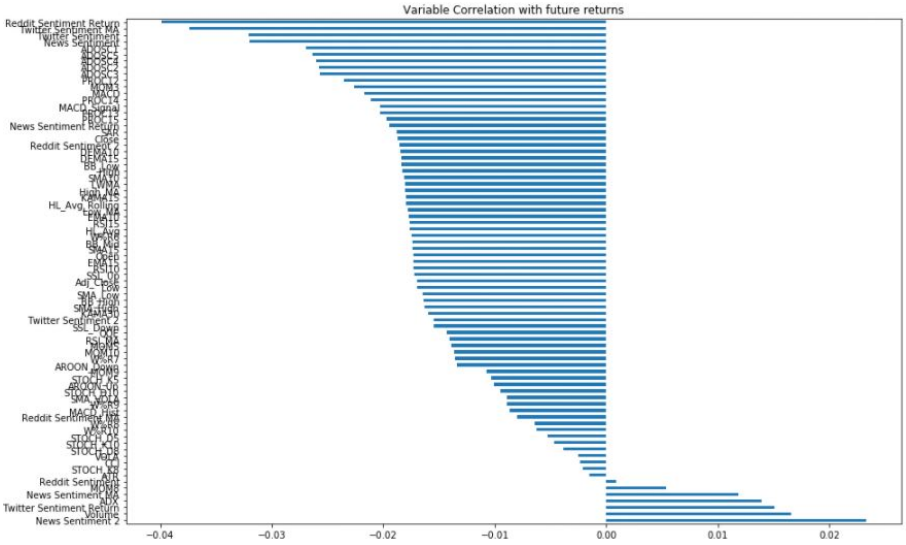


Figure 43: Correlation plot between input features and future returns of AUD/USD dataset

4.6.2 Machine learning for feature selection

The second approach for feature selection uses machine learning or statistical algorithms such as Principal Component Analysis (PCA). PCA is a dimensionality reduction method that extracts a small number of features through the transforming of the feature matrix into a smaller one, all whilst retaining the original information of the dataset. Before PCA can be performed, the dataset must be standardised. This is to ensure a standard range between variables such that each feature contributes equally to analysis. The Scikit-learn StandardScaler library was used for this process by subtracting the mean and dividing by the standard deviation for each observation. Following this, a covariance matrix was calculated to highlight the correlations between input features. The covariance matrix for a dxd matrix is given by:

$$\begin{bmatrix} Cov(x_1, x_1) & Cov(x_1, x_2) & ... & Cov(x_1, x_d) \\ Cov(x_2, x_1) & Cov(x_2, x_2) & ... & Cov(x_2, x_d) \\ ... & ... & ... & ... \\ Cov(x_d, x_1) & Cov(x_d, x_2) & ... & Cov(x_d, x_d) \end{bmatrix} \tag{4.23}$$

Where $Cov\,(x_n, x_n) = Var\,(x_n)$, therefore the top left diagonal of the matrix is equivalent to the variances of each feature. Similarly, the covariance operator is commutative such that $Cov\,(x_i, x_j) = Cov\,(x_j, x_i)$. Hence, the matrix is symmetric with respect to the diagonal of variances.

Using this covariance matrix, PCA can extract several features call principal components. These components are linear combinations of original features such that they are uncorrelated with each other yet retain the information of the original series. Each successive calculated principal component retains less information so that the first component captures the largest possible variance. To calculate these principal components, the eigenvectors and eigenvalues for the covariance matrix are found, where the eigenvalues represent the variance within each component. Ordering these eigenvectors by their eigenvalues returns an ordering of the most significant principal components.

By using PCA, high dimensional data can be visualised and plotted to observe patterns within the data. For example, Figure 44 shows the top 3 principal components of the AUD/USD dataset after feature generation and fixed horizon labelling of smoothed data. Data points are plotted according to their labelled class, namely blue for uptrends and yellow for downtrends. Here, PCA has been leveraged to observe clustering within the labelled datapoints to gauge the separability of observations. Similarly, PCA can be used to reduce multicollinearity within Forex data [58] as highly correlated features are replaced by principal components. This was beneficial for the AUD/USD dataset as technical indicators were shown to be highly correlated.

Despite the benefits of PCA for dimensionality reduction and data analysis, the interpretability of features was lost as they were replaced by principal components. This was detrimental to the project as it was critical to determine which specific features aided prediction the most and, subsequently, whether the explored feature generation techniques were beneficial to the application domain. Furthermore, other feature selection techniques were required to preserve feature states.

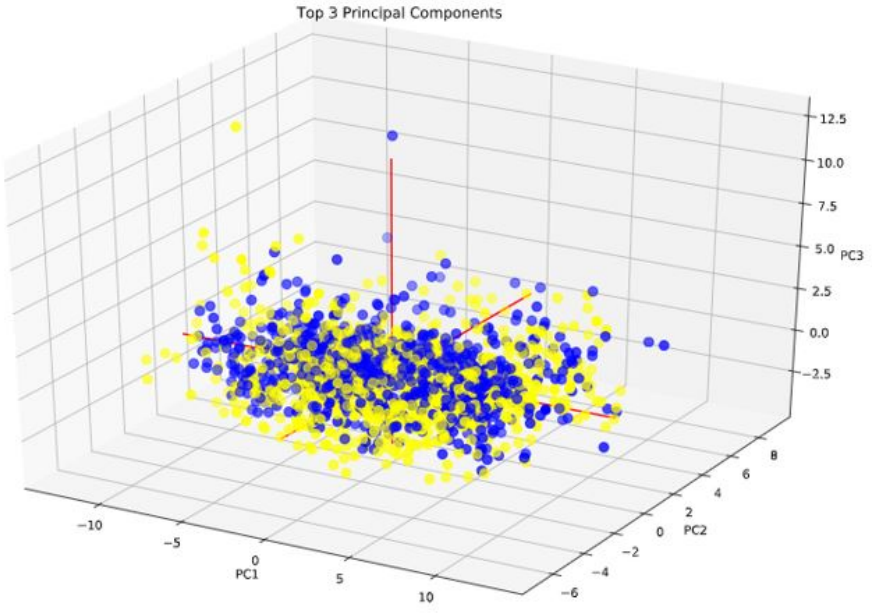


Figure 44: Top 3 Principal Components for AUD/USD Dataset using PCA

Recursive Feature Elimination (RFE) was also used as a machine learning feature selection method. This technique acts a wrapper over a given machine learning model that can provide feature important scores/probabilities [59]. RFE searches for an optimal subset of features by first using all dimensions and iteratively eliminating the least important variables by refitting the underlying model on new subsets. The subset with the highest classification accuracy or the best score according to a given heuristic is chosen as the final set of features. The logic behind this method can be seen in Algorithm 2. It is important to note that the estimator used to rank predictor variables can be independent of the model used for final prediction.

In the project, a logistic regression model was used to calculate importance scores. This was because the model utilized a sigmoid function in which feature weights could easily be used to reflect feature importance. Figure 45 displays the best model when predicting trends in the AUD/USD dataset. It was found that the best performing model selected a subset of 26 features, a reduction of 49 dimensions, and achieved a classification accuracy of 58%. Also, it can be observed that the lagged Twitter

sentiment, normal Twitter sentiment and news sentiment were all selected predictors. Thus, further supporting the use of Sentiment & Fundamental analysis for feature generation.

Alternatively, the work of Baasher & Fakhr [21] motivates the use of Support Vector Machine RFE (SVM-RFE) in which variable coefficients are used to rank features. This, however, is limited to a linear kernel and was found to have a similar performance to the logistic regression approach. Other models such as Bagging Trees are also used. Here, the out-of-bag mean squared error is averaged over all trees when removing features and is used to rank the best predictors. Nonetheless, the logistic regression model was found to be the most efficient technique for the given features and AUD/USD dataset.

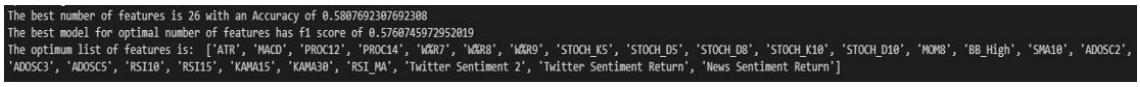


Figure 45: Selected features after RFE

Algorithm 2 Recursive Feature Elimination

procedure RFE
    train, test ← Split Data
    Train model on training data
    accuracies ← list of tuples
    Calculate feature importance an order from test data
    for `subset of size` s_i `from` $i = 1...num\_features$ **do**
        `top_features` ← s_i most important features
        `Train model of feature subset` s_i `on training data`
        acc ← model performance on test data
        `accuracies.append(`s_i`,acc)`
    Order accuracies by <value>
    best_feats ← s_i.columns from accuracies[top]
    best_num_feats ← len(best_feats)
    return `best_num_feats, best_feats`
=0

5 Testing, Results & Evaluation

After features had been generated, selected, and transformed to achieve stationarity, approaches to Forex trend prediction were evaluated. This section explains the testing, results, and evaluation methods for the project in which a plethora of standard classifiers were used to predict future trends of the AUD/USD dataset.

5.1 Machine Learning Models

Various machine learning classifiers were evaluated from tree-based methods to graph and ensemble models. The Scikit-learn library was used for this as it provided convenient out of the box classifiers that were easy to both tune and customize. Furthermore, the classifiers tested within this research included: Random Forest, Support Vector Machine (SVM), Decision Tree, K-Nearest Neighbours (KNN), Logistic Regression and AdaBoost. A Voting Classifier was also implemented as an ensemble method to test how combining the tested models would affect performance.

5.1.1 Decision Trees

The Decision Tree classier is a tree-based model that splits data using specific features through decision nodes and leaves. The leaves of the tree describe the outcome/class label, whilst nodes specify splitting conditions. As the problem being approached is a binary classification task, a binary tree is used in which each node has a maximum of two children.

There exist many algorithms for decision trees that determine the way in which splits are made, however, the Scikit-Learn library gives the choice of two criterion, namely the Gini Index and Entropy. Entropy measures the amount of uncertainty within a series such that:

$$Entropy = -\sum_{j=1}^{n} p\left(c_j\right) * log_2\left(p\left(c_j\right)\right) \tag{5.1}$$

where $p\left(c_j\right)$ is the probability of class c_l occurring within a node. Similarly, the Gini index also measures impurity within a node by calculating the frequency of mislabelled

elements within a dataset when randomly labelled, given by:

$$Gini = 1 - \sum_{i=1}^{n} p^2\left(c_i\right) \tag{5.2}$$

Moreover, Information Gain is the metric used to decide the best feature to serve as the next node within a decision tree. It measures the remaining entropy after splitting according to a feature's values and is calculated across all features. The feature with the most information gain is chosen such that the entropy of labels decreases the most, reducing the randomness within values. Using information gain as a heuristic allows for the reduction of disorder within target variables and is calculated by:

$$IG = Entropy\left(x_i\right) - \frac{n_{yes}}{n} * Entropy\left(x_i\right) - \frac{n_{no}}{n} * Entropy\left(x_i\right) \tag{5.3}$$

where $Entropy\left(x_i\right)$ denotes the impurity of feature values x_i through the chosen impurity criterion (Gini or Entropy) and n_{yes}/n_{no} is the number of positive/negative class labels. Furthermore, by splitting in terms of Information Gain, a decision tree can be built for the classification problem. Figure 46 depicts the first three levels of a Decision Tree classifier after predicting log return trends of a smoothed AUD/USD dataset.

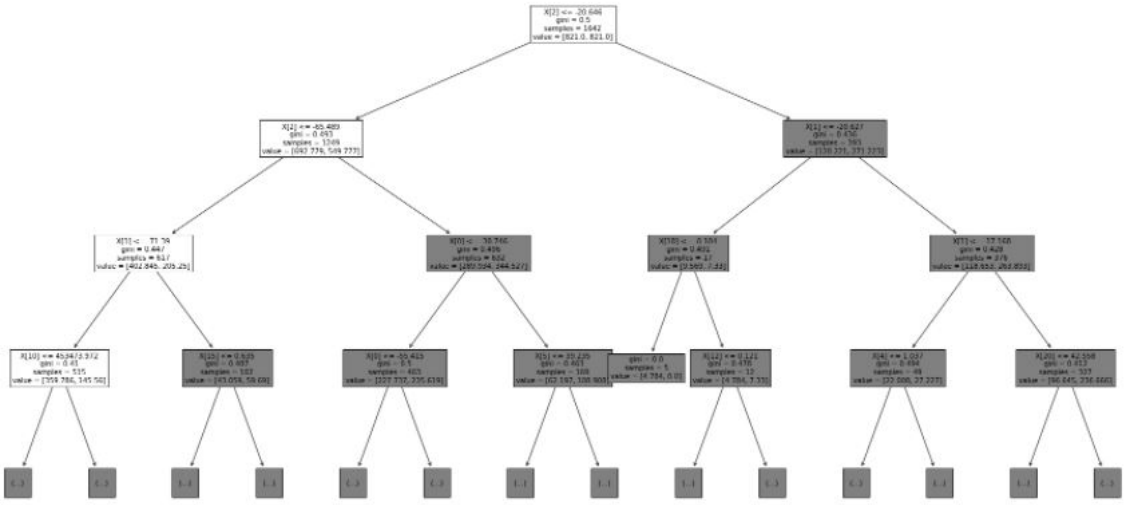


Figure 46: First three levels of Decision Tree model after predicting smoothed rolling high low average labels for AUD/USD dataset

There are many hyperparameters that exist for decision trees that must be manual tuned for model optimization. Conveniently, the *Sciki-learn* library allows for the choice of many of these values when initialising a decision tree. Table 3 shows the hyperparameters tuned for the decision tree classifier within this project and the corresponding value ranges used. These included the impurity criterion used *(criterion)*, the node splitting method *(splitter)*, the maximum depth of generated trees *(max_depth)*, the number of data samples required for splits *(min_samples_split)* and the class weights *(class_weight)*.

Hyperparameter	**Values**
Criterion	*['gini','entropy']*
Splitter	*['best','random']*
Max_depth	*[100,300,500,700,1000]*
Min_samples_split	*[2,4,6,8,10]*
Class_weight	*[None, Balanced]*

Table 3: Hyperparameters and values tested for Decision Tree classifier

5.1.2 Random Forest

The Random Forest classifier is another tree-based model that is built from multiple decision trees. Final predictions are generated by aggregating the predictions of individual decision trees via majority voting. Furthermore, the splits made by separate models follows from the calculations of normal decision trees. This is an example of an ensemble machine learning model in which a range of individual classifiers are joined under a single architecture for prediction. Exploring such a model provided valuable insights into how the performance of standalone models compared to an ensemble.

Like decision trees, the Random Forest classifier has hyperparameters that can be tuned such as the number of trees within the forest *(n_estimators)* and the number of samples require for each split *(min_samples_split)*. Table 4 illustrates these hyperparameters and their corresponding value ranges used within testing.

Hyperparameter	Values
n_estimators	*[100,200,400,600,800,1000]*
min_sample_splits	*[1,2,3,4,5]*

Table 4: Hyperparameters and values tested for Random Forest classifier

5.1.3 Support Vector Machine

Support Vector Machine's (SVM's) are graph-based models that attempt to optimize a hyperplane within an n-dimensional space such that all datapoints are distinctively classified. Within the binary classification case, a hyperplane is chosen to separate two classes as best as possible through the maximizing of the hyperplane margins. This is the maximum distance to the closest datapoints from each respective class.

The position of the chosen hyperplane is dependent upon Support Vectors. These are data points which lie on the boundary of soft margins. Removing a support vector would change the position and orientation of the hyperplane and so these points are used to optimize decision. Moreover, to help maximize the margin, $\frac{2}{||w||}$, between data points and minimize the number of violations within soft margins, a loss function is enrolled such as the Hinge Loss. The Hinge Loss is a non-degenerate, convex function which is given by:

$$max0, 1 - y_i W^T x_i \tag{5.4}$$

where y_i is the ground truth for observation i, W^T is a weight vector and x_i is the i^{th} feature vector in the dataset.

For linearly separable data, the SVM attempts to minimize the following objective function for all examples:

$$f\left(W\right) = \sum_{i=1}^{n} max0, 1 - y_i W^T x_i * \frac{\lambda}{2} \left|\left|W\right|\right|^2 \tag{5.5}$$

Here, the L-2 Regularized Hinge Loss is used and λ is the controlling parameter for the margin/misclassification trade-off.

For non-linearly separable datasets, the Kernel Trick is employed in which data is projected onto a higher dimensional space such that it becomes easier to define hyperplane boundaries and classify data points. The Kernel function, K, is given by:

$$K\left(x_i, x_j\right) = \phi\left(\langle x_i, x_j \rangle\right) = \phi\left(x_i\right) * \phi\left(x_j\right) \tag{5.6}$$

where $\phi\left(\langle x_i, x_j \rangle\right)$ is an inner product using the mapping function $\phi\left(*\right)$. Therefore, by using a Kernel function, the exact mapping function is not required. A commonly used kernel function is the Radial Basis Function (RBF):

$$K\left(x_i, x_j\right) = e^{-\gamma\left(x_i - x_j\right)^2} \tag{5.7}$$

where γ is a hyperparameter that defines the influence of the distance of a single data point.

Furthermore, the SVM classifier has a range of hyperparameters that are shown in Table 5. This includes the kernel function used, the value for C, which is penalty given to misclassified points, and values for gamma.

Hyperparameter	**Values**
gamma	$[2^{-15}, 2^{-13}, 2^{-11}, 2^{-9}, 2^{-7}, 2^{-5}, 2^{-3}, 2^{-1}, 2^{1}, 2^{3}]$
C	$[2^{-5}, 2^{-3}, 2^{-1}, 2^{1}, 2^{3}, 2^{5}, 2^{7}, 2^{9}, 2^{11}, 2^{13}, 2^{15}]$
kernel	*['rbf']*

Table 5: Hyperparameters and values tested for the Support Vector Machine

5.1.4 K-Nearest Neighbours

Another graph-based model is the K-Nearest Neighbours (KNN) classifier. This non-parametric model classifies new data points based on a similarity/distance function. The K data points that are closest to the new observation are selected and the mode of their attributed labels is returned as the generated class. An example of a distance

function is the Euclidean distance in which the similarity of points is measured by:

$$Euclidean\ Distance\ = \sqrt{\sum_{i=1}^{k} (x_i - y_i)^2} \tag{5.8}$$

Table 6 shows the tuneable hyperparameters and the corresponding tested values for the KNN model. The parameters included the value for K *(n_neighbours)*, the weights of points in each neighbourhood *(weights)*, the distance function used *(metric)* and the algorithm employed to compute the nearest neighbours.

Hyperparameter	**Values**
n_neighbours	*[5,7,9,. . . ,45]*
weights	*['uniform','distance']*
metric	*['euclidean','manhattan','minkowski','chebyshev']*
algorithm	*['auto', 'ball_tree', 'kd_tree', 'brute']*

Table 6: Hyperparameters and values tested for the KNN model

5.1.5 Logistic Regression

The logistic regression classifier models the probability that a class label occurs given a set of features and weights. Moreover, the model maps inputs to probabilities using the sigmoid function. This activation function scales values to a range between 0 and 1 and is defined as:

$$Sigmoid\,(z) = \frac{1}{1 + e^{-z}} \tag{5.9}$$

where z is the dot product of a set of input features and weights. Using this function, probability outputs can be classified into labels by defining a threshold as a decision boundary.

To learn and optimize feature weights, the cross-entropy loss is enrolled as a cost function. This function penalizes high probability misclassifications more than it rewards high probability correct classifications. Through this nature, the prediction accuracy of the model increases where the ideal loss is a value of 0. The cross-entropy loss function

is given by:

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m} -y_i * log\left(h_\theta\left(x_i\right)\right) + \left(1 - y_i\right) * log\left(1 - h_\theta\left(x_i\right)\right) \tag{5.10}$$

where $h_\theta(x_i)$ is the hypothesis of logistic regression $\frac{1}{1+e^{-w^T x_i}}$. Moreover, algorithms such as gradient descent can be used to minimize calculated costs and regularization can be applied to prevent extreme coefficient values.

The hyperparameters used with the logistic regression model includes the number of iterations taken for solver convergence *(max_iter)*, the inversed regularization strength *(C)* and the weight of individual classes *(class_weight)*. Table 7 demonstrates these parameters and the corresponding value ranges tested within the research.

Hyperparameter	**Values**
max_iter	*[100, 200, 400, 600, 800, 1000]*
C	$[2^{-5}, 2^{-3}, 2^{-1}, 2^{1}, 2^{3}, 2^{5}, 2^{7}, 2^{9}, 2^{11}, 2^{13}, 2^{15}]$
class_weight	*['balanced','None']*

Table 7: Hyperparameters and values tested for the Logistic Regression model

5.1.6 AdaBoost

Another ensemble model is the Adaptive Boosting (AdaBoost) classifier. In this model, several weak learners are employed in which weights are provided for both the models and training data. The weights are adjusted at each iteration of the algorithm, implementing a new classifier with the new weights. The intuition is that each subsequent model learns from the misclassification of previous weak learners. This way, several suboptimal (almost random walk) models are combined to generate meaningful predictions. Furthermore, after generating M classifiers, the final predictions are taken as weighted combinations of the models.

Define the weight for example n in weak learner m as w_n^m. Also define the cost function for the m^{th} classifier to be:

$$J_m = \sum_{i=1}^{n} w_i^m \left[\widehat{y_i^m} \neq y_i\right] = \sum weighted\ errors \tag{5.11}$$

During training, the initial weights for all data observations are equal ($w_i^1 = \frac{1}{n}, \forall i = 1 \dots n$). Each weak learner m is fitted to the weighted training data and predictions $\widehat{y^m}$ are made using cost function J_m. Following this, an error rate ϵ_m is calculated as

$$\epsilon_m = \frac{J_m}{\sum_{i=1}^{n} w_i^m} \tag{5.12}$$

and the quality α_m of each model is determined by

$$\alpha_m = log\left(\frac{1-\epsilon_m}{\epsilon_m}\right) \tag{5.13}$$

The weights of observations in the training data are subsequently updated such that the weights for the next weak learner w_i^{m+1} are given by

$$w_i^{m+1} = w_i^m * exp\left(-\frac{1}{2} y_i \alpha_m \widehat{y_i^m}\right) \tag{5.14}$$

Furthermore, for test data and new observations, the final predictions $\widehat{y^M}$ are generated using a weighted combination of all the previous weak learners, where

$$\widehat{y^M} = sign\left(\sum_{m=1}^{M} \frac{1}{2} \alpha_m * \widehat{y^m}\right) \tag{5.15}$$

Moreover, within this project, the weak learners used by the AdaBoost model were decision trees. Table 8 demonstrates the hyperparameter values tested which included the number of estimators at which boosting halts *(n_estimators)*, the weight given to each weak learner at each boosting iteration *(learning_rate)* and the base estimator used from which an ensemble was built *(base_estimator)*.

Hyperparameter	Values
num_estimators	*[50, 100, 200, 400, 500]*
learning_rate	*[0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1]*
base_estimator	*[DecisionTree(maxdepth=1)* – *DecisionTree(maxdepth=21)]*

Table 8: Hyperparameters and values tested for the AdaBoost model

5.1.7 Voting Classifier

The final model tested was the Voting Classifier. This model takes several estimators/-classifiers as input and computes final predictions as an aggregation of individually predicted class labels. Moreover, during testing, the previously explored models were all passed into a Voting Classifier in an ensemble strategy to combine the abilities of various algorithms.

Table 9 shows the sole hyperparameter tested for the Voting Classifier which was the voting type. The "Hard" voting type predicted labels using a majority voting rule over all inputted models. On the other hand, "Soft" voting predicted class labels as the argmax of the sums of predicted probabilities over all input classifiers.

Hyperparameter	Values
voting	*['Hard','Soft']*

Table 9: Hyperparameters and values tested for the Voting Classifier

5.2 Training & Testing Methods

5.2.1 Model Validation

Validation is the process of determining how well a model will perform on test data through error estimations. When training data, error residuals can provide information on how well a model performs. Using this as an approximation of test error can often lead to over/under fitting. Therefore, validation data can be made from a subset of the training data and used as an unbiased approximation of the test error. Validation error can also be used to tune model hyperparameters before final evaluations.

Moreover, cross validation is a technique employed to approximate test error when data samples are sparse. This approach involves the splitting of training data into k even folds and iteratively validating on each fold after training a model on the rest. The accuracy of the model is then calculated by taking an average over all iterations. This approach works best when data is shuffled which presents a clear problem for time series data as time ordering is broken.

To employ k-fold cross validation on time series data, the algorithm for the approach was altered such that training and validations splits were made to maintain chronological orderings. This was achieved by iteratively increasing the size of the training set, starting from the beginning of the series, and shifting the validation set in an expanding window approach. Figure 47 depicts this behaviour in which the size of the validation set is constant as the training size increases. Alternatively, a sliding window approach could be taken in which both the training and validation set shift in unison and are of a constant length.

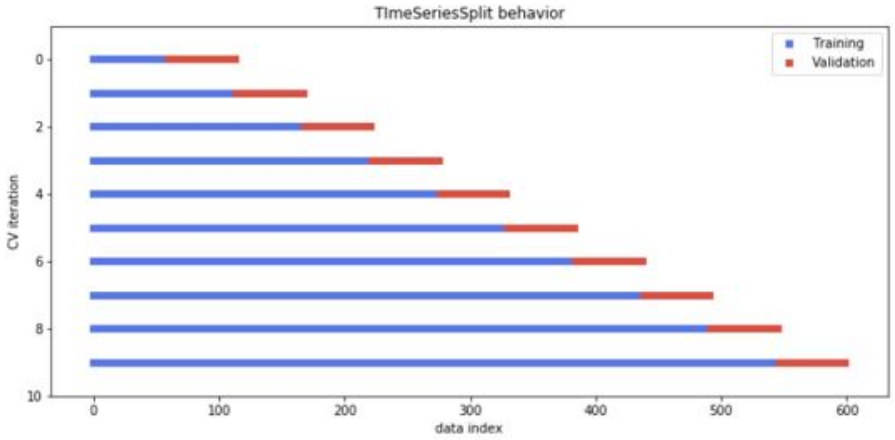


Figure 47: Time series split with expanding window visualisation

5.2.2 Optimizing Hyperparameters

Hyperparameters are parameters that are manually tuned to control the learning process of machine learning models. Unlike parameters such as feature weights, hyperparameters cannot be learned by the models themselves and so an independent optimiza-

tion process is required.

A Grid-Search is an optimization technique that can find optimal values for model hyperparameters through a brute force approach. This is where combinations of hyperparameter values are iteratively used to train models in which the best performing hyperparameters are returned. Different scoring functions/heuristics can be used to determine the performance of hyperparameters in a Grid-Search, the most common of which being classification accuracy. Moreover, this technique is computationally demanding as every hyperparameter combination is evaluated, however, guarantees to find the best values from a given grid. Furthermore, it has been shown that grid searches work best on datasets with few dimensions and a small number of hyperparameters [60].

Alternatively, the work of Jubert et al [24] suggests the use of a Genetic Algorithm for hyperparameter optimization. In this approach, the concept of natural selection is imposed where the fittest individuals amongst a population are selected to produce offspring with beneficial adaptations. For hyperparameter tuning, Genetic algorithms first initialize a population of hyperparameters with random values into a population vector. For each value in the population, a model is trained, and a fitness function is evaluated in which the best performing "parent" parameters are chosen. Crossover is then enforced where children hyperparameter values are chosen from their parents' domains. Finally, mutation occurs in which diversity is introduced by randomly changing one of the children's hyperparameter values. This approach has been shown work well when the ranges of hyperparameter values are unknown and is faster than brute-force alternatives [60].

Moreover, due to time constraints, a Grid-Search was employed for model tuning during the project. This was because the Scikit-Learn Python library provided a simple implementation of the Grid-Search algorithm which was both streamlined and customisable. It also allowed for time series k-fold cross validation when training with parallel

computation for splits. Similarly, with this approach, a range of scoring functions were afforded as well as the ability to optimize all the tested classifiers and their varying hyperparameters.

5.3 Evaluation Metrics

To be able to compare the performance of models and employed techniques, a set of evaluation metrics were created. It was important to consider both classical machine learning metrics as well as financial evaluations as commented by Li & Suohai [12] to obtain a holistic understand of model performance. Table 10 shows the list of metrics enforced as well as their formulae. It can be observed that standard classification metrics such as classification accuracy, precision, recall and f1-score are used. Precision refers to the proportion of correct positive classifications out of all positive labels. Recall, however, pertains to the how accurately a model can classify true positives. Finally, f1-score is the harmonic mean of the precision and recall of a model.

The other metrics used were concerned with the financial evaluation of models. This was important as it was unreasonable to base model performance on classification statistics alone considering the financial application of the project. Moreover, the set of financial statistics in Table 10 were calculated after financial back testing of each classifier. During back testing, each model was tested on historic OHLC data and predictions made by each model were used to calculate the strategies cumulative returns and equity. Using this data, financial evaluation metrics could be determined such as Sharpe's ratio. This metric measures the risk adjusted return of a strategy in which a value >1 is indicates a financially viable model. Maximum Drawdown was also calculated which describes the maximum observed loss from peak to trough through a models backtest. This metric can gauge the riskiness of a strategy; however, only captures the largest loss achieved and not the frequency of losses. Furthermore, it was important to use these metrics in unison to provide a comprehensive understanding model's financial viability.

Hyperparameter	**Values**	**Parameters**
Classification Accuracy	$\frac{TP+TN}{TP+TN+FP+FN}$	True/False Positive/Negatives
Precision	$\frac{TP}{TP+FP}$	
Recall	$\frac{TP}{TP+FN}$	
F1-Score	$\frac{2*Precision*Recall}{Precision+Recall} = \frac{TP}{TP+\frac{1}{2}(FP+FN)}$	
Sharpe's Ratio	$\frac{E\|r_p-r_{rf}\|}{std(r_p-r_{rf})} * \sqrt{N}$	$r_p = \ strategy\ return$, $r_{rf} = riskfree\ rate$, N=$Number\ of\ yearly\ samples$
Total Return	$\frac{E_t-E_0}{E_0}$	$E_t = Equity\ at\ time\ t$
Maximum Drawdown	$\frac{TroughValue-PeakValue}{PeakValue}$	

Table 10: Classification & Financial evaluation metrics

As well as metrics, a series of visualisations were used to provide a basic interpretation of results and model performance. The first method utilised was the confusion matrix which visualised the classification accuracy, precision, recall and f1-score of a model. Within a confusion matrix squares are labelled by the output classes, with values equal to the appropriate frequency of classifications.

Another plot used during testing was the validation curve. This graph shows the accuracy of models as the value for a hyperparameter changes for training and validation data. Such a plot can help identify if a model suffers from bias where under/overfitting occurs. Similarly, it can help determine initial values for hyperparameters to best fit the data.

Finally, after back testing, cumulative return visualisations were used to model the performance of model strategies. Here, the running return of each model was plotted against baseline buy-and-hold strategies. By doing this, it was clear to see whether implemented machine learning models could beat the most naïve of approaches to Forex trend prediction.

5.4 Results & Discussions

5.4.1 Testing Environment & Process

Initially, next day trends were going to be predicted for the AUD/USD time series, however, it became quickly apparent that this was an inherently hard problem due to do the stochastic nature of the market and its noise. Therefore, as much of the current literature suggested, smoothed data was predicted with raw log returns over larger horizon periods. This required the re-labelling of the dataset which used an altered Fixed Horizon Method over both smoothed and raw data. In this method, the log returns of the high-low price average was used and labelled with a 5-day horizon period. This meant that models would be predicting the log returns of the high-low average 5 days into the future as opposed to next day log/normal returns.

For the smoothed data, a 5-day simple moving average was applied to the AUD/USD price series. Labels were then generated with the same Fixed Horizon Method but using the log returns of a rolling high-low average. The intuition for producing two sets of labels was to evaluate the effect of predicting smoothed vs raw data on the classification accuracy and financial viability of machine learning models. The reason for this comparison stemmed from the conclusions of the literature review in which there was an apparent disparity between the accuracy of similar models. By investigating this issue, the most efficient approach to Forex trend prediction could be determined.

Furthermore, the process for testing each model was normalized to streamline evaluation which comprised of the following steps:

1. Splitting the dataset into 80% training and 20% test data, maintaining temporal ordering and standardizing values using *StandardScaler*.
2. Initializing the machine learning model with default hyperparameters.
3. Generating time series splits for cross validation using a value of $k = 5$.
4. Initializing grid of hyperparameter values to test & optimize.

5. Performing Grid-Search with 5-fold cross validation using the defined hyperparameter grid and F1-score as the evaluation function.

6. Extracting best performing model and evaluating on test data whilst calculating standard classification metrics.

7. Performing financial backtest and calculating financial evaluation metrics.

8. Generating performance visualizations.

5.4.2 Results and Evaluation

The prediction of smoothed trends within the Forex market was tested first in which models were trained on the AUD/USD dataset from 2011-2019. Table 11 shows the classification results for all models during this test in which the best standalone classifier was the Logistic Regression model with a 64.3% classification accuracy. This model also attained the highest F1-score in which the optimised hyperparameters included balanced class weights, 100 max iterations for algorithm convergence and a C value of 128. The SVM classifier attained similar results with a 64% accuracy when utilizing a Radial Basis Function (RBF) kernel.

On the other hand, the Decision Tree classifier attained the lowest classification accuracy of 54.8%. This could be due to overfitting that occurred during training or an indication that naïve tree-based models are unable to capture patterns within financial time series. Despite this, the Random Forest classifier achieved a relatively high F1-Score of 63.8%. This contradicts the results of the single tree model and demonstrates how the enrolment of multiple classifiers can better identify smoothed trends.

	SVM	RF	KNN	ADB	LOGR	DT	VC
Classification Accuracy (%)	64.0	63.9	60.2	61.2	**64.3**	54.8	**65.3**
F1-Score	63.8	63.8	60.0	60.7	64.4	52.3	64.2
Precision	63.2	64.3	59.7	62.6	64.5	54.6	65.4
Recall	64.1	63.2	60.5	58.9	63.2	50.5	63.1
Best Hyperparameters	C: 32, class_weight: 'balanced', gamma: 0.001953125, kernel: 'rbf'	min_samples_split: 4, n_estimators: 800	algorithm: 'auto', metric: 'manhattan', n_neighbors: 21, weights: 'distance'	base_estimator: DecisionTreeClassifier (max_depth=6), learning_rate: 0.8, n_estimators: 400	C: 128, class_weight: 'balanced', max_iter: 100	class_weight': 'balanced', criterion: 'entropy',max_depth: 500, min_samples_split: 4, splitter: 'random'	Type: Hard VC

Table 11: Classification Evaluation metrics for machine learning models when predicting smoothed data with rolling high-low average fixed horizon labelling

It can also be noted that the Voting Classifier achieved the highest overall classification accuracy of 65.3%, outperforming standalone alternatives. This model utilized a Hard-VC in which a majority vote was taken between the predictions of all classifiers. Moreover, such findings convey the effectiveness of ensemble methods for prediction as models are not used as silver bullets, thus support thing findings of Khan et al [25]. Furthermore, the general performance of all classifiers when predicting smoothed trends was strong, with the lowest accuracy belonging to the standalone decision tree model.

After initial classification analysis, the models were backtested on smoothed data with an initial equity of $1000. Table 12 shows the results of the individually backtested classifiers. It can be observed that the strongest classifier according to standard evaluation metrics, the Logistic Regression model, attained a small total return of 0.82% with a net profit of $8.21. This is interesting considering how effective the model was able to identify smoothed trends. In addition to this, the model obtained a Sharpe's ratio of 0.04 which is suboptimal and indicates a weak trading strategy.

	SVM	RF	KNN	ADB	LOGR	DT	VC
Equity before Backtest	1000.00	1000.00	1000.00	1000.00	1000.00	1000.00	1000.00
Equity after Backtest	1032.45	**1052.23**	**1043.40**	969.88	1008.21	854.64	1027.15
Net Profit	32.45	52.23	43.40	30.12	8.21	-145.36	27.15
Total Return	3.25%	**5.22%**	4.34%	-3.01%	0.82%	-14.54%	2.72%
Sharpe's Ratio	0.21	**0.36**	**0.30**	-0.25	0.04	-1.23	0.18
Max Drawdown	-10.88%	-9.89%	-9.13%	-14.71%	-10.26%	-15.75%	-12.02%

Table 12: Financial Evaluation metrics for machine learning models when predicting smoothed data with rolling high-low average fixed horizon labelling after backtest

The least profitable and riskiest model after backtesting was the Decision Tree classifier, with a total return of -14.54% and a maximum drawdown of -15.75%. This, however, was to be expected considering it was also the weakest model in terms of classification performance.

On the other hand, the strongest model after backtesting was the Random Forest classifier which attained a net profit of $52.23. Similarly, the KNN model obtained a net profit of $43.40 and a maximum drawdown of -9.13%. Despite this, the Sharpe's Ratio for both models was 0.36 and 0.30, respectively. These values are relatively low and are far off being optimal. Furthermore, none of the tested classifiers displayed promising financial statistics following the backtest.

The confusion matrices and cumulative return plots for all models can be found in Appendix B.

	SVM	RF	KNN	ADB	LOGR	DT	VC
Classification Accuracy (%)	50.1	50.0	50.0	**54.6**	52.4	49.7	**52.9**
F1-Score	33.8	49.5	49.5	54.3	52.3	49.8	50.8
Precision	33.2	49.3	49.2	54.8	52.1	47.6	53.3
Recall	49.5	49.8	49.9	53.5	52.6	50.4	50.5
Best Hyperparameters	C: 0.5, class_weight: 'balanced', gamma: 0.5, kernel: 'rbf'	min_samples_split: 2, n_estimators: 600	algorithm: 'auto', metric: 'chebyshev', n_neighbors: 7, weights: 'uniform'	base_estimator: DecisionTreeClassifier (max_depth=1), learning_rate: 0.8, n_estimators: 50	C: 32, class_weight: 'balanced', max_iter: 200	class_weight': 'balanced', criterion: 'entropy',max_depth: 300, min_samples_split: 4, splitter: 'random'	Type: Soft VC

Table 13: Classification Evaluation metrics for machine learning models when predicting raw data with high-low average fixed horizon labelling

Following the evaluation of smoothed trend prediction, models were trained with labels generated by the raw high-low average of the AUD/USD dataset. Table 13 shows the corresponding classification performance of each model. It can be quickly observed that the general classification accuracies across all models decreased with the AdaBoost model achieving the highest accuracy of 54.6%. This model enrolled 50 Decision Trees of depth one (decision stumps) as estimators and a learning rate of 0.8. In particular, the model's highest accuracy was over a 10% less than that of the best model on smoothed trends. Furthermore, the Voting Classifier performed similarly on raw trend labels with a 52.9% classification accuracy. Notably, despite the decrease in overall accuracy, the two ensemble models outperformed standalone alternatives on raw market trends.

Like previous results, the worst performing model on raw trends was the Decision Tree classifier which accomplished a 49.7% classification accuracy. This was worse than a random walk and reiterates the difficulty associated with predicting raw returns within financial time series over small horizon periods.

Contrarily, within the backtest results in Table 14, it is evident that there was a significant increase in model financial performance. Specifically, the total return and net profit across all models drastically improved with the AdaBoost classifier achiev-

ing a 22.17% return and -5.53% maximum drawdown. This follows from its previous performance during raw trend prediction. To further this, the model had a Sharpe's Ratio of 1.54 which is extremely high and implies a highly financially viable strategy. The Voting Classifier also achieved promising results with a net profit of $176.60 and a Sharpe's Ratio of 1.30.

	SVM	RF	KNN	ADB	LOGR	DT	VC
Equity before Backtest	1000.00	1000.00	1000.00	1000.00	1000.00	1000.00	1000.00
Equity after Backtest	914.93	1059.56	993.06	**1221.75**	1130.51	1086.66	**1176.60**
Net Profit	85.07	59.56	-6.94	221.75	130.51	86.66	176.60
Total Return	-8.51%	6.00%	-0.69%	**22.17%**	13.05%	8.67%	**17.66%**
Sharpe's Ratio	-0.67	0.42	-0.08	**1.54**	0.96	0.60	**1.30**
Max Drawdown	-14.89%	-5.78%	-12.01%	-5.53%	-8.74%	-7.60%	-8.14%

Table 14: Financial Evaluation metrics for machine learning models when predicting raw data with high-low average fixed horizon labelling after backtest

Figure 48 and Figure 49 show the cumulative returns of both the Voting Classifier and AdaBoost model during the entire backtest. Both models are plotted against the returns of a baseline buy-and-hold strategy which mimics the returns of the price series. In both cases, the models beat the baseline strategy by the end of the backtest. However, it can be observed that the AdaBoost classifier struggled to deal with bullish periods as shown at the start of the plot when returns were increasing yet strategy returns were negative. The Voting Classifier did not exhibit this same limitation as cumulative returns remained relatively positive for the majority of the backtest. Furthermore, this highlights the financial and classical benefits of ensemble methods for Forex trend prediction.

Moreover, it is also important to note the performance of the Decision Tree classifier during the backtest. This model obtained a positive return of 8.67% and a Sharpe's Ratio of 0.6. This is interesting as the worst performing model in terms of classification

accuracy across all tests was able to convey financial viability on raw trends. This emphasises the effect of changing the predicted output variable and labelling methods on both the accuracy and financial performance of machine learning models.

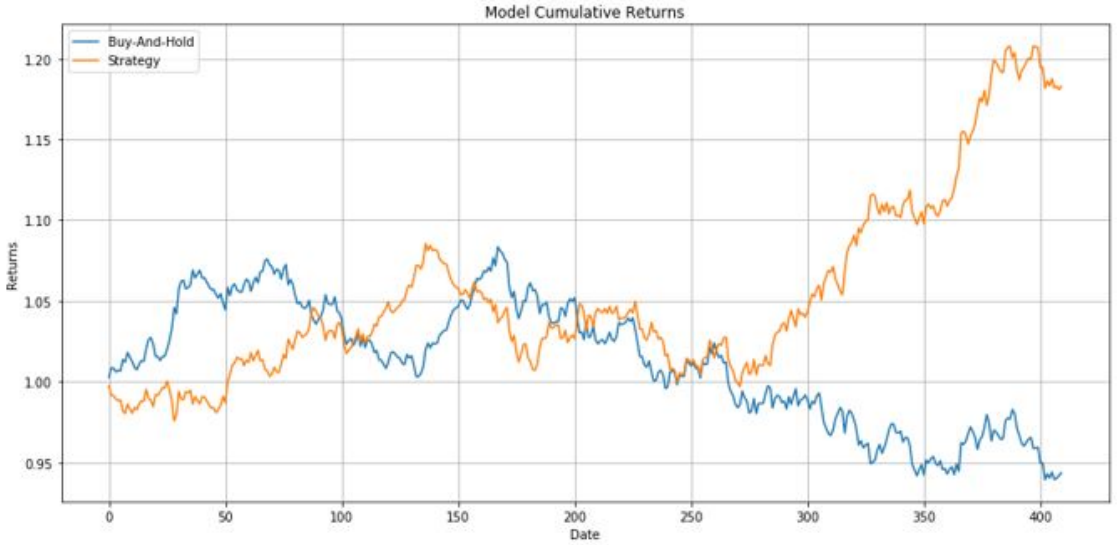


Figure 48: Cumulative Returns plot for Voting Classifier on raw trends vs baseline buy-and-hold strategy

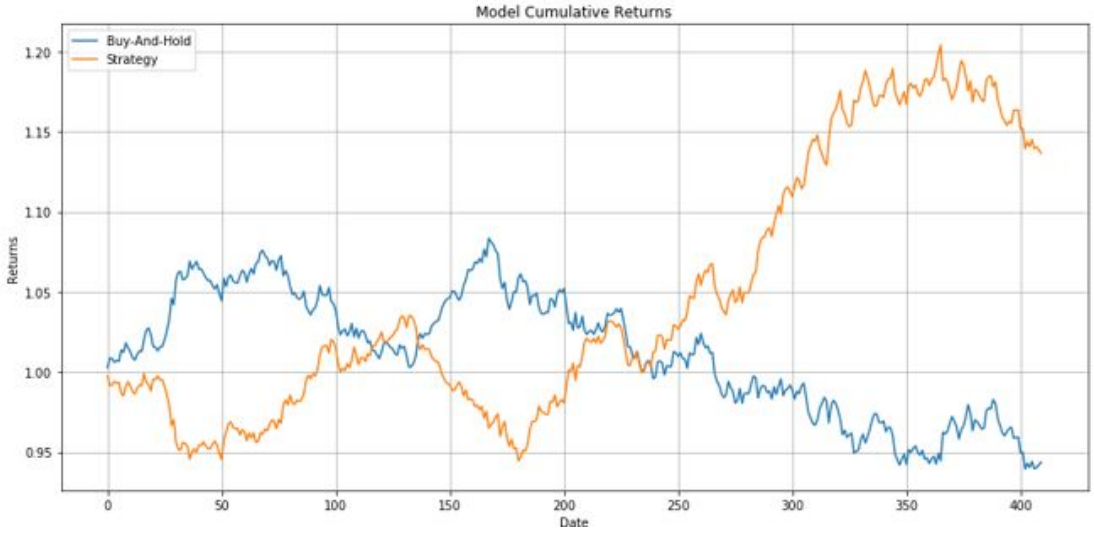


Figure 49: Cumulative Returns plot for AdaBoost classifier on raw trends vs baseline buy-and-hold strategy

5.4.3 Conclusions

Within this research, a range of techniques have been explored pertaining to the prediction of trends within the forex market. The current approaches have been critically evaluated from which a set of beneficial requirements were built. This began with the employment of several pre-processing methods to analyse the distributions and patterns within Forex data. Following this, the ubiquitous approach to data labelling was criticised and the Triple Barrier Method was evaluated as an alternative but showed no adverse performance or computational benefits.

Moreover, a novel approach to feature engineering has been proposed that consisted of Technical, Sentiment and Fundamental analysis. Popular engineered technical indicators were scrutinised in which new features such as the SSL and VI indicators were implemented from current trader theories and standard libraries. Within Sentiment and Fundamental analysis, trader opinions through social media were shown to hold predictive power over the Forex market and abstract sentiment features were engineered. Furthermore, methods for attaining stationarity of time series data were explored in which Fractional Differencing was considered superior to Integer Differencing due to memory retention, however, incurred greater computational costs and data loss.

After attaining a final dataset, both manual and machine learning feature selection techniques were employed to impose dimensionality reduction and deal with multicollinearity. Performing Recursive Feature Elimination (RFE) over a logistic regression model revealed that sentiment features such as Twitter and News sentiment return were optimal for prediction. Thus, further motivating the use of social media analysis for Forex market prediction. Principal Component Analysis (PCA) was also enrolled to visualise the separability of data and showed that data labelled by smoothed trends were more visually separable in three dimensions than raw trends.

Using the optimized dataset of selected features, both smoothed and raw trends were predicted within the AUD/USD dataset from 2011 to 2019 by range of machine

learning classifiers. Overall, the models trained on smooth data were able to better classify corresponding trends in the Forex market, however, the models trained on raw prices achieved far greater financial backtest results. This deduction follows from the research of Gerlein et al [5] in which standard classifiers performed poorly in terms of raw trend binary classification but excelled on financial evaluations. Ensemble methods for prediction were also shown to outperform standalone classifiers and beat baseline strategies when backtested, supporting the findings of Khan et al [25]. Furthermore, the two forms of evaluation were shown not to be positively correlated such that a highly accurate classifier did not necessarily exhibit financial viability. Finally, the limitations of employing standard classification evaluation metrics alone for models applied to financial markets was highlighted as well as the iterative nature of analysis within the application domain.

5.4.4 Project Limitations

Due to the nature of the project and the conditions surrounding the research period, there were many limitations of the implemented works. Firstly, due to being bounded by the computational limits of a local workstation, the size of the datasets that could be processed were restricted. Consequently, daily data was investigated which has been shown to be less rich than higher granularity alternatives such as tick data. Similarly, the range of evaluated classifiers was constrained to a tractable set, meaning that more complex and potentially beneficial models such as deep neural networks were not employed. Furthermore, the market intricacies and patterns that could be captured by machine learning models were also restricted due to data choices.

Moreover, many choices were made throughout the project to allow for simplicity and avoid the golden ratio problem. This is where the number of tuneable hyperparameters is too great such that the problem becomes an optimization task. As a result of this, several strict assumptions were enforced such as the labelling, stationarity and feature selection techniques used. The problem approached was also limited to binary classification rather than multi-class. By restricting these methods to simpler approximations,

a trade-off was observed between performance and complexity where the potential accuracy of implemented strategies was bounded by the assumptions made.

Similarly, during Sentiment and Fundamental analysis, the generated features were bounded by the quality of data being scraped from online. As there was no readily available high quality sentiment datasets, public social media posts were leveraged which were entirely unregulated and unverified. Despite introducing pre-processing steps, this data was not fully reliable and produced sparse feature series from which interpolation had to be performed.

Additionally, although ensemble methods were explored during testing and proved beneficial results, machine learning was still generally used as a silver bullet. This is because models were evaluated independently of each other rather than in a system for prediction which was originally proposed.

A final limitation pertaining to the enforced evaluation metrics was the lack of realistic costs. Within live Forex trading, various overheads exist such as the platform fees, commissions, and the latency associated with distributed systems. These factors are important as many of the positive returns revealed within the financial backtests of models would be diminished by the overheads of the live market. Thus, by not incorporating such costs, the true financially viability of machine learning models was not fully determined.

6 Future Work

6.1 Extending Models & Neural Networks

There exists a wide range of scope in which this project can be extended and methods further researched. The most intuitive area for further exploration is the types of models used to predict the Forex market. Specifically, Neural networks have been demonstrated by the likes of Tang & Cheng [26] and Zhelev & Avresky [14] to be beneficial for Forex market prediction when using LSTM's. Such deep learning approaches can learn both feature mappings and weights to extract the most beneficial patterns from time series data and would fit perfectly for the application domain. Similarly, alternative machine learning classifiers can be explored such as the Naïve Bayes classifier and further tuning can be performed on existing models.

Moreover, for Sentiment Analysis, machine learning can be explored for sentiment derivation and feature extraction from social media datasets. This would provide a more accurate sentiment derivation model that could leverage the explained benefits of the NLTK library such as tokenization, speech tagging and n-grams.

6.2 Acquiring Richer Datasets

Another critical area for expansion is the size and quality of processed datasets. Within the project daily historic Forex data was investigated which was limited in both size and encoded information. Therefore, richer data can be explored through the processing of higher frequency/granularity datasets such as hourly/minutely/tick OHLC data. This would enable the use of more data analysis techniques without the fear of information loss such as Fractional Differencing. Furthermore, evaluating data of different frequencies allows for a thorough analysis of the effect of granularity on prediction.

Sentiment and Fundamental analysis data can also be improved by acquiring data from a wider range of reliable sources. This could include the pulling of macroeconomic indicators or financial reports within certain periods or scraping data from alternative

social medias such as Facebook. The analysis process would also benefit from more rigorous data pre-processing and to ensure the integrity of pulled sentiment data.

6.3 Expansive Evaluation Metrics

As explained in the limitation section of this work, the employed evaluation metrics failed to incorporate the overheads of real Forex trading. A potential extension of this work would include the implementation of brokerage fees, latencies and commissions that would provide a more realistic evaluation of a strategy's financial viability. This follows from the problems realised in the work of Gerlein et al [5] in which there is a disparity between the results of various evaluation metrics. A way in which this extension could approached is through the formalising of the backtesting process. This involves building implemented strategies into a trading platform in which backtesting tools are provided. By having a regulated body evaluate a strategy, results can be compared relatively and more reliably.

6.4 Hybrid Models & Optimizations

The work of Khan et al [25] and Jubert et al [24] motivated the use of hybrid and ensemble methods within this research. This idea can be extended to the explicit case where a system for prediction is developed. Such a system could incorporate both machine learning and non-machine learning components in a prediction pipeline to diverge from the black box approach currently enforced. The results of this project support this extension as ensemble methods such as the Voting and Random forex classifier outperformed standalone models. Moreover, statistical models such as ARIMA, that have been proven to be effective in forex market forecasting [17], can be leveraged with manual or automated strategies to optimize strategy performance.

Other optimization techniques such as Genetic Algorithms and Artificial Swarm algorithms have been utilised to tune models in financial market prediction [4], [24], [60]. Such approaches can be investigated to streamline the tuning of hyperparameters of tested models and improve upon the runtime of the enrolled grid search.

6.5 Alternative Output Variables

As well as extending the approaches to forex prediction, the problem itself can be varied to extend the scope of this project. Within this research, forex returns trend prediction is explored as a binary classification problem. This is a simplification of financial market prediction and so alternative output variables such as volatility, volume or market sentiment can be forecasted as an abstraction of the problem statement. Furthermore, the regression problem can be approached as well as extending to the multi-class classification case in which non-trending regions are handled.

7 Data Analysis & Machine Learning Dashboard

The initial scope of this project was to holistically research Forex trend prediction using machine learning. By nature, this took a research heavy approach in which the data analysis and supervise machine learning stack were investigated. Moreover, it was not initially envisioned for a tangible system to be built. Despite this, due to efficient time management and planning, an application was created as a demonstration of the project's critical findings, displayed in the project roadmap in Figure 6, and was utilized during the project presentation.

Moreover, the system was made as a data analysis dashboard in which historic forex data could be pulled, visualised, and pre-processed in real time. Furthermore, machine learning components such as feature generation, selection and model training/testing were implemented in which classifiers could be initialised with hyperparameter values and evaluated. Figure 50 illustrates the dashboards user interface which is presented as a series of visualisation tiles and buttons. In this example, the AUD/USD dataset was loaded and visualised through the data tree view and a plot of the closing price. Data analysis and machine learning processes were then performed. Furthermore, the system allowed for both smoothed and raw labelling as explored during the research. The right-hand side of the dashboard displays the models results in which a Logistic Regression model was trained with default hyperparameters on smoothed trend labels. The confusion matrix, cumulative returns plot and validation curve were all returned with corresponding statistics.

In conclusion, by building such an application, the financial market prediction process was demonstrated. Consequently, the iterative nature of the research was embodied as new datasets could be pulled in sequentially after initial analysis and model results. Such a system could be extended through the previously explained ideas as a step towards automated trading.

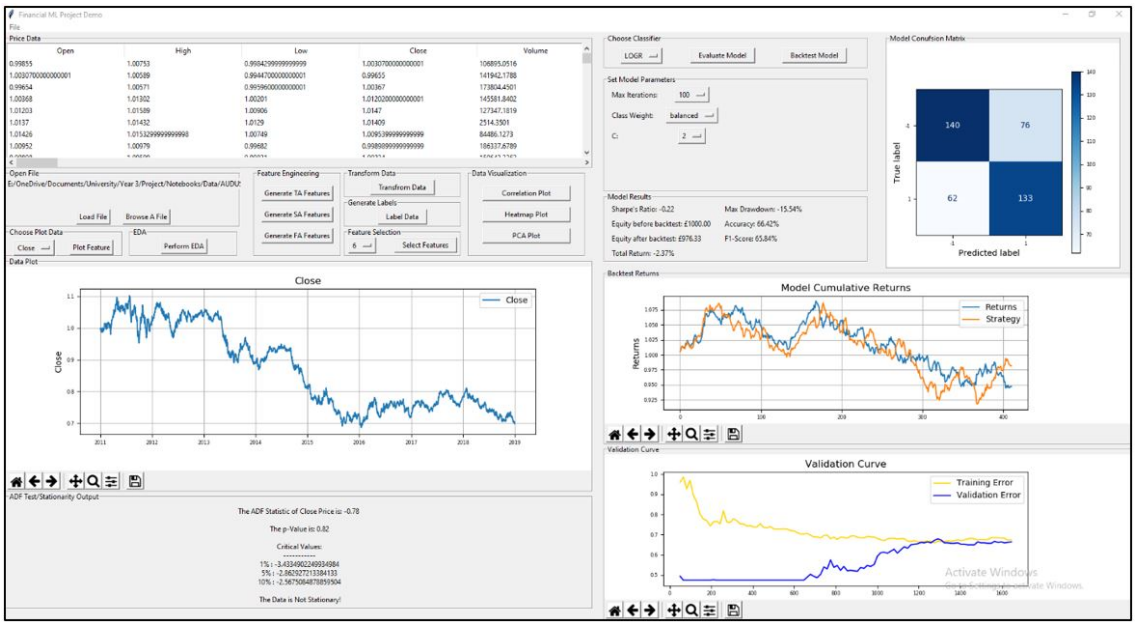


Figure 50: Implemented project dashboard with trained Logistic Regression model on AUD/USD dataset

References

[1] J. Chen, "Learn about trading fx with this beginner's guide to forex trading." https://www.investopedia.com/articles/forex/11/why-trade-forex.asp, 3 2021.

[2] M. Islam, E. Hossain, A. Rahman, M. Hossain, and K. Andersson, "A review on recent advancements in forex currency prediction," *Algorithms*, vol. 13, p. 186, 4 2020.

[3] N. Oliveira, P. Cortez, and N. Areal, "Some experiments on modeling stock market behavior using investor sentiment analysis and posting volume from twitter," Association for Computing Machinery, 2013.

[4] A. Sespajayadi, Indrabayu, and I. Nurtanio, "Technical data analysis for movement prediction of euro to usd using genetic algorithm-neural network," pp. 23–26, 5 2015.

[5] E. A. Gerlein, M. McGinnity, A. Belatreche, and S. Coleman, "Evaluating machine learning classification for financial trading: An empirical approach," *Expert Systems with Applications*, vol. 54, pp. 193–207, 2016.

[6] M. J. S. de Souza, D. G. F. Ramos, M. G. Pena, V. A. Sobreiro, and H. Kimura, "Examination of the profitability of technical analysis based on moving average strategies in brics," *Financial Innovation*, vol. 4, 12 2018.

[7] H. Tankovska, "Number of social media users 2025." https://www.statista.com/statistics/278414/number-of-worldwide-social-network-users/#:~:text=In2020,over3.6billion,almost4.41billionin2025., 1 2021.

[8] "Ig client sentiment report 2021-05-10 12:00." https://www.dailyfx.com/sentiment-report.

[9] S. W. Sidehabi, Indrabayu, and S. Tandungan, "Statistical and machine learning approach in forex prediction based on empirical data," pp. 63–68, 11 2016.

[10] H. Wang, "A study on the stock market prediction based on genetic neural network," pp. 105–108, Association for Computing Machinery, 2018.

[11] D. Pradeepkumar and V. Ravi, "Forex rate prediction using chaos and quantile regression random forest," pp. 517–522, 3 2016.

[12] M. Li and F. Suohai, "Forex prediction based on svr optimized by artificial fish swarm algorithm," pp. 47–52, 12 2013.

[13] Y. C. Shiao, G. Chakraborty, S. F. Chen, L. H. Li, and R. C. Chen, "Modeling and prediction of time-series-a case study with forex data," pp. 1–5, 10 2019.

[14] S. Zhelev and D. R. Avresky, "Using lstm neural network for time series predictions in financial markets," pp. 1–5, 9 2019.

[15] S. Lakshminarayanan and J. P. McCrae, "A comparative study of svm and lstm deep learning algorithms for stock market prediction," 2019.

[16] T. N. T. Thu and V. D. Xuan, "Using support vector machine in forex predicting," pp. 1–5, 2018.

[17] H. P. S. D. Weerathunga and A. T. P. Silva, "Drnn-arima approach to short-term trend forecasting in forex market," pp. 287–293, 9 2018.

[18] M. H. Eng, Y. Li, Q. Wang, and T. H. Lee, "Forecast forex with ann using fundamental data," vol. 1, pp. 279–282, 12 2008.

[19] R. Chiong, Z. Fan, Z. Hu, M. T. P. Adam, B. Lutz, and D. Neumann, "A sentiment analysis-based machine learning approach for financial market prediction via news disclosures," pp. 278–279, Association for Computing Machinery, 2018.

[20] S. D. Immanuel and U. K. Chakraborty, "Genetic algorithm: An approach on optimization," pp. 701–708, 7 2019.

[21] A. A. Baasher and M. W. Fakhr, "Forex trend classification using machine learning techniques," 2011.

[22] T. N. T. Thu and V. D. Xuan, "Forex trading using supervised machine learning," *International Journal of Engineering & Technology*, vol. 7, 10 2018.

[23] Y. Lee, L. T. C. Ow, and D. N. C. Ling, "Hidden markov models for forex trends prediction," pp. 1–4, 5 2014.

[24] B. J. de Almeida, R. F. Neves, and N. Horta, "Combining support vector machine with genetic algorithms to optimize investments in forex markets with high leverage," *Applied Soft Computing*, vol. 64, pp. 596–613, 2018.

[25] W. Khan, M. ali Ghazanfar, M. A. Azam, A. Karami, K. Alyoubi, and A. Alfakeeh, "Stock market prediction using machine learning classifiers and social media, news," *Journal of Ambient Intelligence and Humanized Computing*, 4 2020.

[26] J. Tang and X. Chen, "Stock market prediction based on historic prices and news titles," pp. 29–34, Association for Computing Machinery, 2018.

[27] M. Prado, *Advances in financial machine learning*. John Wiley & Sons, Inc, 2018.

[28] T. Rao and S. Srivastava, "Modeling movements in oil, gold, forex and market indices using search volume index and twitter sentiments," pp. 336–345, Association for Computing Machinery, 2013.

[29] T. Segal, "Fundamental analysis." `https://www.investopedia.com/terms/f/fundamentalanalysis.asp`, 4 2021.

[30] Z. Cheng, T. Qi, J. Wang, Y. Zhou, Z. Wang, Y. Guo, and J. Zhao, "Sentiment evaluation of forex news," pp. 197–201, Association for Computing Machinery, 2019.

[31] O. Chantarakasemchit, S. Nuchitprasitchai, and Y. Nilsiam, "Forex rates prediction on eur/usd with simple moving average technique and financial factors," pp. 771–774, 6 2020.

[32] V. Patil, N. Somani, A. Tadvi, and V. Attar, "Algorithmic forex trading using combination of numeric time series and news analysis," pp. 1–5, 10 2018.

[33] M. Neshat, G. Sepidnam, M. Sargolzaei, and A. N. Toosi, "Artificial fish swarm algorithm: a survey of the state-of-the-art, hybridization, combinatorial and indicative applications," *Artificial Intelligence Review*, vol. 42, 12 2014.

[34] J. Carapuço, R. Neves, and N. Horta, "Reinforcement learning applied to forex trading," *Applied Soft Computing*, vol. 73, pp. 783–794, 2018.

[35] J. Fonseca, R. Neves, and N. Horta, "Optimizing investment strategies based on companies earnings using genetic algorithms," pp. 1731–1732, Association for Computing Machinery, 2013.

[36] "Top 10 most traded currency pairs." `https://www.ig.com/uk/trading-strategies/top-10-most-traded-currency-pairs-191206`.

[37] R. Snow, "How central banks impact the forex market." `https://www.dailyfx.com/education/forex-fundamental-analysis/how-central-banks-impact-forex.html`.

[38] F. F. S. Board, "The most liquid currency pairs." `https://fxssi.com/most-liquid-currency-pairs`, 4 2021.

[39] H. Jeffreys and B. Swirles, *Methods of mathematical physics*. Cambridge Univ. Press, 2001.

[40] S. Mcleod, "What is kurtosis?." `https://www.simplypsychology.org/kurtosis.html`.

[41] R. Cont, "Empirical properties of asset returns: stylized facts and statistical issues," *Quantitative Finance*, vol. 1, pp. 223–236, 2001.

[42] J. Brownlee, "How to decompose time series data into trend and seasonality." `https://machinelearningmastery.com/decompose-time-series-data-trend-seasonality/`, 12 2020.

[43] J. Salvi, "Significance of acf and pacf plots in time series analysis." `https://towardsdatascience.com/significance-of-acf-and-pacf-plots-in-time-series-analysis-2fa11a5d10a8`, 3 2019.

[44] I. Cohen, "Optimizing feature generation." https://towardsdatascience.com/optimizing-feature-generation-dab98a049f2e, 11 2019.

[45] J. Fernando, "Relative strength index (rsi)." https://www.investopedia.com/terms/r/rsi.asp, 4 2021.

[46] J. Fernando, "Relative strength index (rsi)." https://www.investopedia.com/terms/r/rsi.asp, 4 2021.

[47] "Retail foreign exchange trading." https://en.wikipedia.org/wiki/Retail_foreign_exchange_trading, 3 2021.

[48] "Trading strategies, signals and technical indicators - tradingview - uk." https://uk.tradingview.com/scripts/.

[49] "A trading strategy using the ssl indicator." http://www.forex-central.net/SSL-indicator-trading-strategy.php.

[50] M. P. Allen, "The problem of multicollinearity," 1997.

[51] J. Brownlee, "A gentle introduction to the bag-of-words model." https://machinelearningmastery.com/gentle-introduction-bag-words-model/, 8 2019.

[52] B. White, "Sentiment analysis: Vader or textblob?." https://towardsdatascience.com/sentiment-analysis-vader-or-textblob-ff25514ac540, 5 2020.

[53] R. J. Hyndman and G. Athanasopoulos, *Forecasting: principles and practice*. OTexts, 2018.

[54] "Why log returns." https://quantivity.wordpress.com/2011/02/21/why-log-returns/, 11 2012.

[55] A. Oliveira and A. Seijas-Macias, "An approach to distribution of the product of two normal variables," *Discussiones Mathematicae Probability and Statistics*, vol. 32, 2012.

[56] W. Feller, *An introduction to probability theory and its applications*. Wiley, 2010.

[57] M. L. de Prado, "The 10 reasons most machine learning funds fail," *SSRN Electronic Journal*, 2018.

[58] T. Sugiarto, "Application of principal component analysis (pca) to reduce multicollinearity exchange rate currency of some countries in asia period 2004-2014," *International Journal of Educational Methodology*, vol. 3, 12 2017.

[59] M. Kuhn and K. Johnson, *Applied predictive modeling*. Springer, 2016.

[60] I. Syarif, A. Prugel-Bennett, and G. Wills, "Svm parameter optimization using grid search and genetic algorithm to improve classification performance," *TELKOMNIKA (Telecommunication Computing Electronics and Control)*, vol. 14, 12 2016.

A List of Twitter Profiles

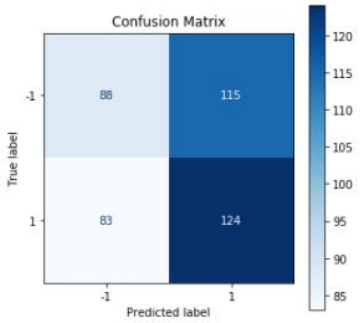
The full list of twitter profiles utilised for scraping tweets can be seen below:

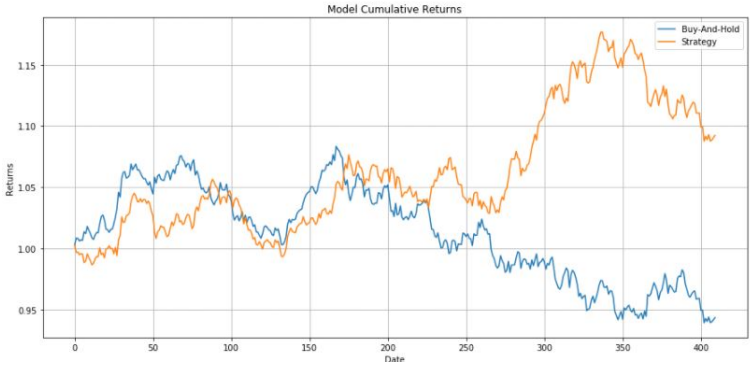
['HotForex_Broker','ForexLive', 'FXstreetNews', 'DailyFXTeam', 'Sean_lee_forex', 'ForexTweets', 'FOREXcom', 'forexcrunch', 'LiveSquawk', 'FT', 'zerohedge', 'ecb', 'bankofengland', 'federalreserve']

B Model Confusion Matrices & Cumulative Return Plots

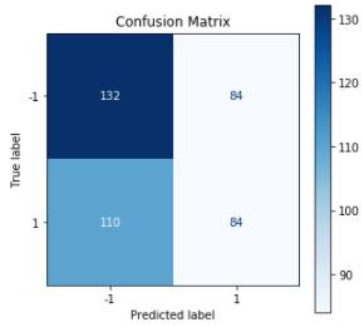
Decision Tree Classifier:

Raw Data:



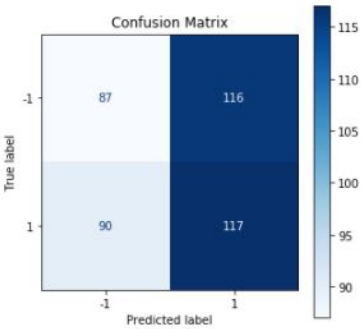


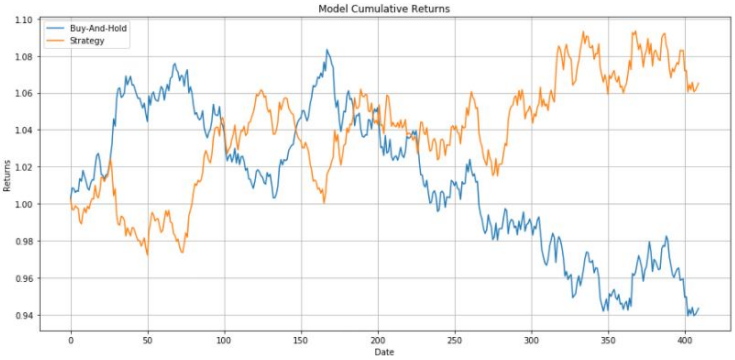
Smoothed:



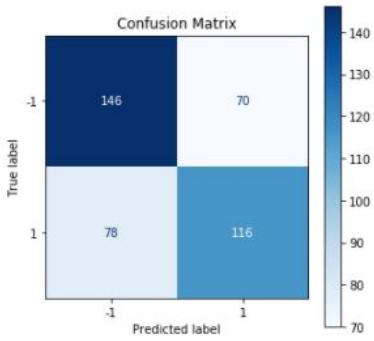
Random Forest Classifier:

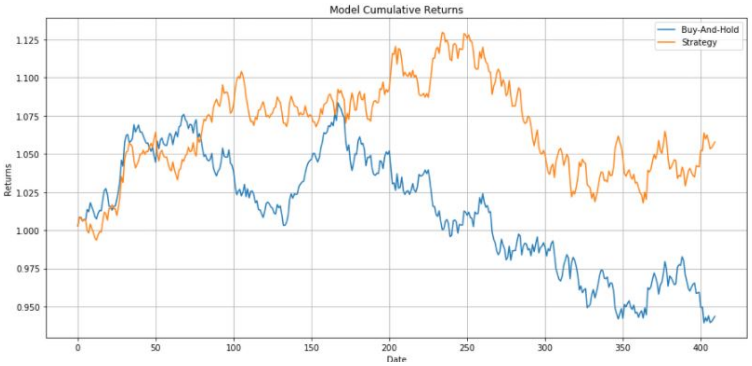
Raw Data:





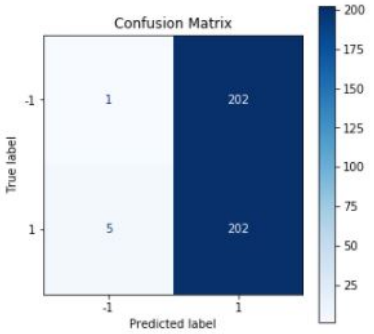
Smoothed:



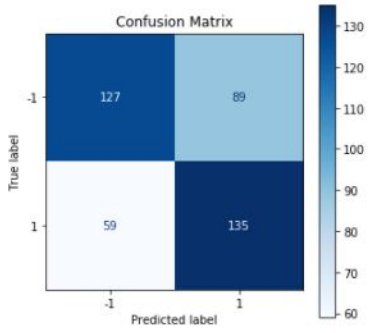


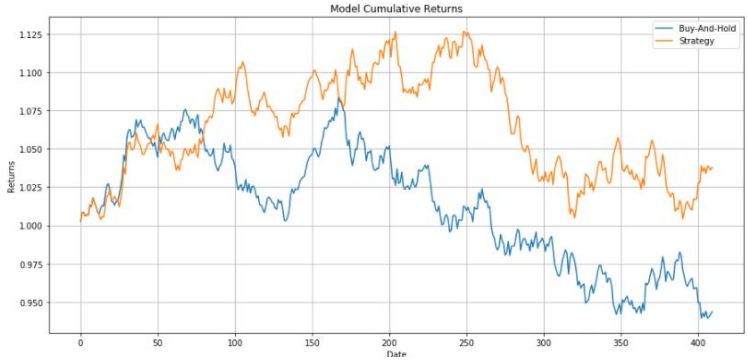
Support Vector Machine:

Raw Data:



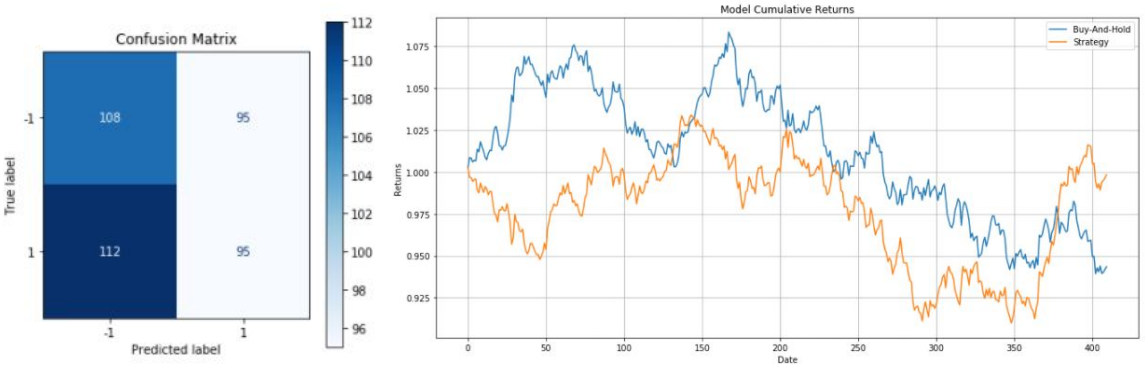
Smoothed:



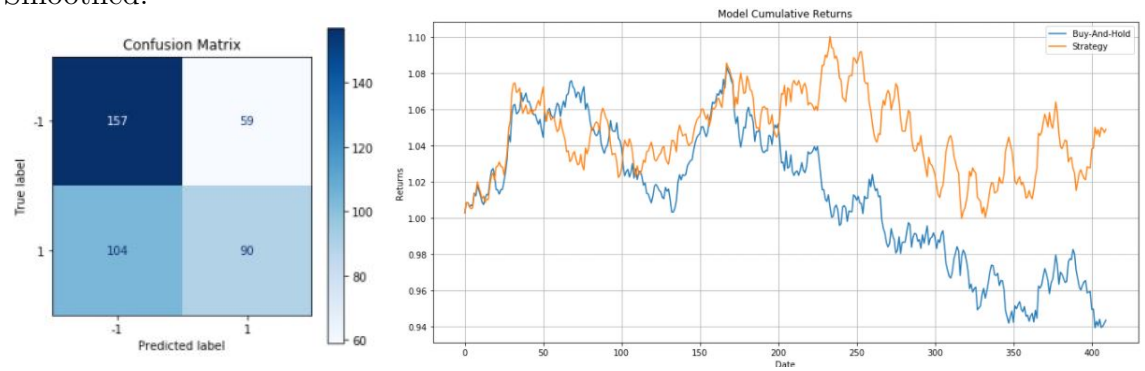


KNN Classifier:

Raw Data:

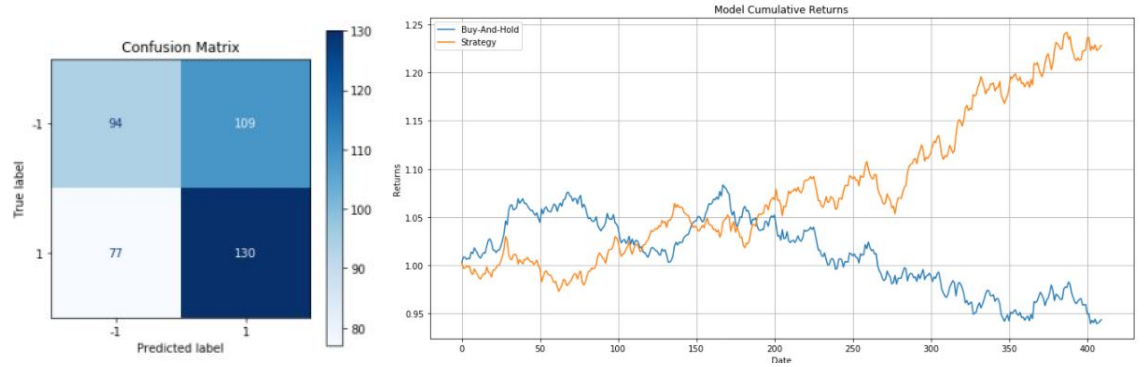


Smoothed:

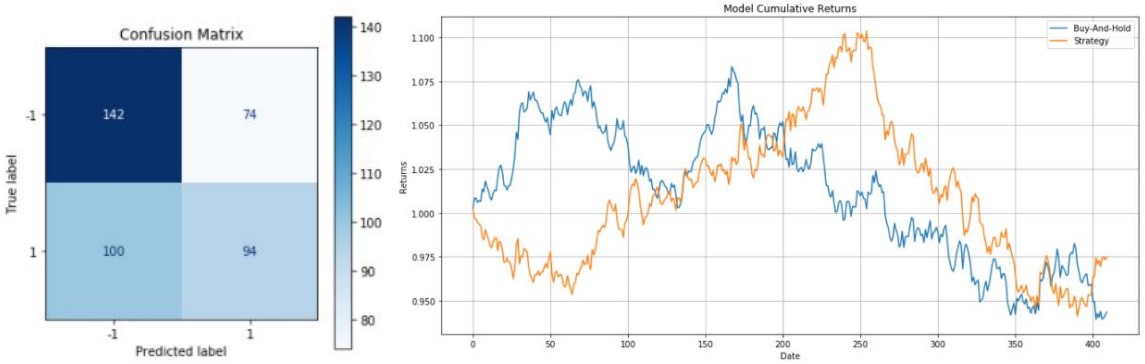


AdaBoost Model:

Raw Data:

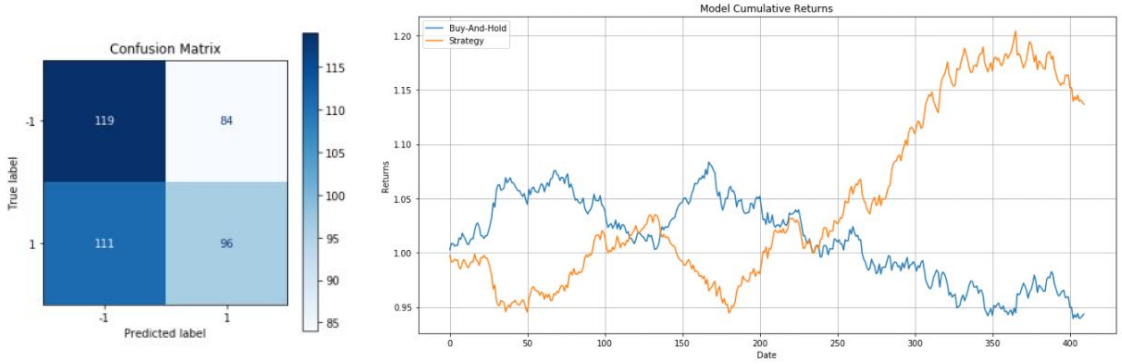


Smoothed:

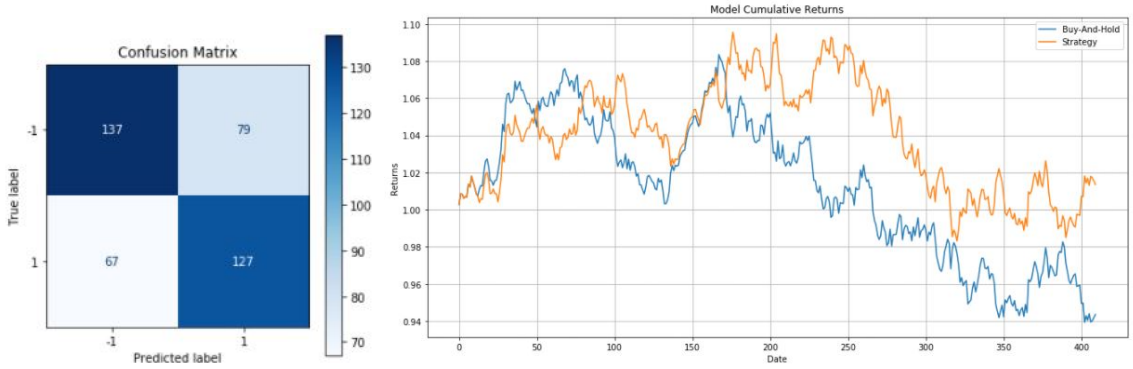


Logistic Regression Model:

Raw Data:

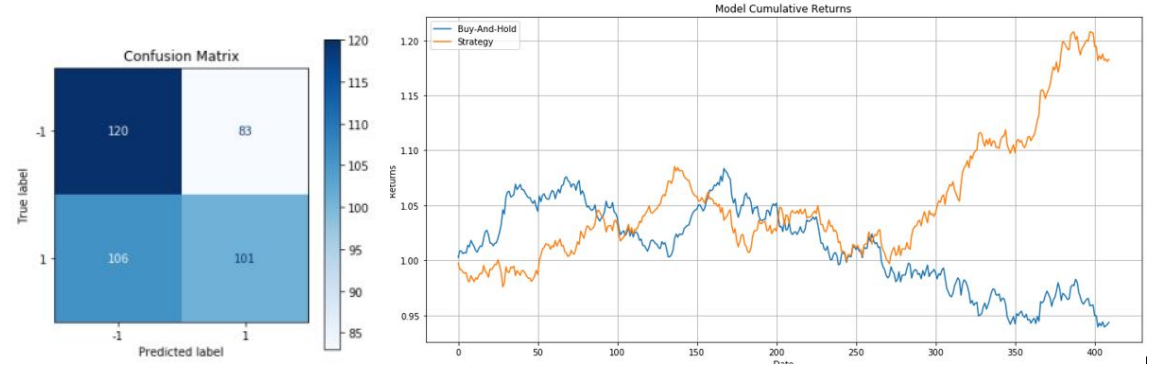


Smoothed:

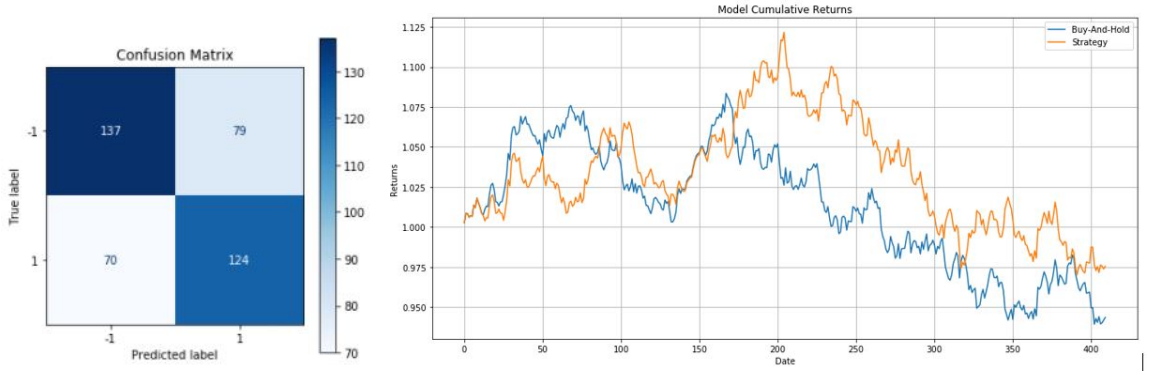


Voting Classifier:

Raw Data:


Confusion Matrix
120
83
106
101
True label
Predicted label
Model Cumulative Returns
Buy-And-Hold
Strategy
Returns

Smoothed:


Confusion Matrix
137
79
70
124
True label
Predicted label
Model Cumulative Returns
Buy-And-Hold
Strategy
Returns
Date