Currency Exchange Rate Prediction using Machine Learning (USD/JPY Focus)

A REPORT SUBMITTED TO THE UNIVERSITY OF MANCHESTER FOR THE DEGREE OF BACHELOR OF SCIENCE
IN THE FACULTY OF SCIENCE AND ENGINEERING

2024

Neel Vashisht
Supervised by Xiao-Jun Zeng

Table of Contents

Table of Figures

Abstract

This dissertation investigates the application of statistical and machine learning models for the prediction of USD/JPY exchange rates. The study compares the effectiveness of ARIMA, LSTM, and XGBoost models, exploring the impact of dataset size and the incorporation of external market factors. Offline ARIMA models exhibited poor performance, however online ARIMA demonstrated superior adaptability, with the best results achieved using a 1-year dataset. LSTM models exhibited a clear correlation between larger datasets and increased accuracy, with the 10-year dataset yielding the lowest errors. Furthermore, the incorporation of stock market data (Nikkei 225 and S&P 500) demonstrably enhanced the performance of both LSTM and XGBoost regression models. Despite promising insights, the XGBoost classification models exhibited limited practical use due to low accuracy scores. This research highlights the importance of model flexibility, dataset size, and the potential value of considering inter-market relationships within FX forecasting.

1 Introduction

1.1 Motivation

The field of financial markets, notably the foreign exchange (FX) market, holds substantial interest due to its scale and influence on global economies. The FX market's daily trading volume surpasses USD 7.5 trillion (Jones and John, 2022), and reflects the expansion of international trade and financial investment activities. This project specifically focuses on predicting exchange rates within this market, as understanding these dynamics is critical for various stakeholders such as traders, investors, and policymakers.

Of the major FX currencies that are traded, the currencies of the G10 are the most significant. These include USD (US Dollar), EUR (Euro), JPY (Japanese Yen), GBP (British Pound Sterling) and CHF (Swiss Franc) which comprise the largest trading volumes. Out of these, the USD/JPY currency pair was selected as they are both considered to be 'safe haven currencies'. This is due to their strong liquidity, stable political systems and low interest rates, meaning at times of geopolitical conflict, investors will invest in this currency pair as they are more likely to maintain their value (Hayes, 2023). This reputation causes the USD/JPY currency pair to be the second most traded pair globally, with a market share of 13.2% (Investopedia, 2023).

Such an active currency pair has multiple factors that drive its rate, some of which include:

- Central Bank Interest Rate Policies: Decisions made by the U.S. Federal Reserve (Fed) and the Bank of Japan (BoJ) regarding interest rates markedly influence the USD/JPY exchange rate (Beine, Bos and Laurent, 2006). For example, a higher U.S. interest rate can strengthen the USD by attracting foreign investment.

- Economic Indicators: Key economic data in the U.S. and Japan, such as GDP growth, inflation, and employment figures, shape investor sentiment and drive fluctuations in the exchange rate.

- Trade Balance: The trade balance between the two countries plays a role in the strength of their currencies, and therefore their exchange rate (Kallianiotis, 2022). For example, Japanese trade surplus can increase demand

for the JPY, while a U.S. trade deficit may cause a downward pressure the USD.

1.2 Objectives

This project aims to develop and test different forecasting models for the USD/JPY currency pair. In particular the main objectives are:

- Build machine learning models to predict FX rates based on historical data
- Identify key factors that influence the exchange rate
- Use these factors in the models to improve accuracy

1.3 Outcomes

Three models were used in the project, with different implementations of each model compared. They were trained on datasets which were split into test and training datasets and evaluated using common error metrics. The core objectives were successfully achieved and through rigorous evaluation, several valuable insights were obtained.

Initial findings expose inherent limitations in the offline ARIMA model's ability to handle the dynamics of financial data. However, the online ARIMA displayed a clear performance advantage, underscoring the importance of incorporating continuous updates to adapt to ever-changing market conditions. This approach offers a promising avenue for further development.

The LSTM displayed a positive correlation between larger datasets and increased performance, confirming its capacity for handling complex patterns. The demonstrable improvement in the LSTM model when incorporating stock market data highlighted the benefit of exploring a wider range of factors for enhanced prediction accuracy.

The XGBoost classification implementation was limited in its output, with mediocre accuracy metrics. However, the regression results suggest a correlation between stock market data and foreign exchange markets with lower error metrics once stock market data was introduced. However, the performance gains were not as pronounced as those seen with the LSTM.

2 Background and Literature Review

2.1 What drives the Forex market

The foreign exchange (also known as forex) market is the world's largest financial market (Kallianiotis, 2013). It is a vast, decentralised network where currencies are traded electronically and facilitates everything from large transactions between nations, to individuals sending money to each other across borders. This is done by exchanging currencies, and it plays a critical role in enabling international trade and investment by determining the relative value of these different currencies. The forex market operates 24 hours a day, 5 days a week, with a daily turnover reaching $7.5 trillion, highlighting its immense size and liquidity.

Given the markets vastness and dynamic nature, even small fluctuations in exchange rates can translate into significant gains or losses. Modern computing power coupled with machine learning algorithms offers the potential to analyse immense datasets, identify patterns, and develop predictive models that could give traders an edge in this highly competitive environment. However, in order to accurately predict how currencies appreciate and depreciate in relation to each other, it is necessary to understand the main factors which drive these changes.

Central banks manage the monetary policy of a particular nation and therefore have the ability to significantly influence exchange rates. They can intervene to stabilise their own currency's value in a number of different ways, with varying degrees of success (Sarno and Taylor, 2001). An example of this can be seen in 2011 with the Swiss National Bank. In order to combat the Swiss franc's (CHF) excessive appreciation during the European debt crisis, the Swiss National Bank (SNB) imposed a temporary peg against the Euro (EUR) in September 2011, setting a minimum exchange rate of 1.20 CHF per EUR (Broby et al., 2016). In practice, this meant that the SNB was committed to buying an unlimited amount of Euros to prevent the Swiss franc from strengthening past this level. This aimed to protect Swiss exporters and counter deflationary pressures, however concerns about the interventions sustainability persisted. In January 2015, the SNB abruptly announced the discontinuation of the currency peg, which led to an immediate appreciation of 20% against the Euro (Delivorias, 2015). This example shows how the actions of a central bank can have a substantial effect on the forex market, and their actions should be considered closely when trying to predict future exchange rates.

Large institutions can momentarily sway exchange rates through sizable trades, especially during periods of lower liquidity. A pertinent example of this was the 1992 ERM (European Exchange Rate Mechanism) ERM crisis, also known as Black Wednesday. The ERM was a system designed to reduce fluctuations between the currencies of European Economic Community (EEC) member states. It aimed to provide economic stability and pave the way for the eventual adoption of a single currency, the Euro. To become and remain a member of the ERM, each nation had to establish a central rate for their currency against the European Currency Unit (ECU) and was only allowed to fluctuate by 2.5% around the ECU. Many investors and large institutions believed that Great British Pound (GBP) was overvalued compared to other EEC currencies and decided to bet against the pound. Their collective actions culminated in the 1992 ERM crisis, which occurred on September 16, 1992. On this day, George Soros and his hedge fund famously made a massive bet against the pound, taking a short position worth over £10 billion making him a profit of more than £1 billion. The UK's central bank (the Bank of England) attempted to defend the pound by buying large amounts of the currency using the UK's foreign reserves, but the pressure from Soros and other investors proved too much. Eventually, the British government was forced to exit the ERM, which caused a sharp devaluation of the pound. In contrast to the previous example of the SNB, this example shows how private companies were able to devaluate the GBP despite heavy intervention from the Bank of England.

2.2 Challenges of Exchange Rate Prediction

To accurately predict FX rates, it is crucial to grasp the nature of historical FX data. This data falls under the category of time series data, where observations are recorded at regular intervals over time. The sequential ordering of these observations is fundamental, as the temporal context can significantly influence the value of each data point. In the context of this project, historical USD/JPY exchange rates constitute time series data. Without their sequential order, any attempt to forecast future values would be inherently flawed.

Accurately predicting forex rates remains a significant challenge in the financial world for many reasons. Financial products such as government bonds or dividend-paying stocks provide a more predictable income stream due to the decreased risk associated with them. Exchange rates, however, are influenced by a complex interchange of factors, making them inherently difficult to forecast with certainty. Examples such as the European debt crisis's impact on the CHF and black Wednesday highlight the difficulty of anticipating major market shifts. However, even

beyond these specific events, the inherent volatility of exchange rates makes forecasting with certainty extremely difficult.

One key challenge is the non-linear nature of the relationships between the various factors influencing exchange rates. Interest rates, economic growth, inflation, and political stability are all interconnected, but their impact on exchange rates can be not only non-linear but potentially counterintuitive at times. According to the Uncovered Interest Rate Parity (UIP) theory, the relative change in foreign exchange rates over the same period will be equal to the difference in interest rates between two countries (Hayes, 2021). Theoretically this makes sense as higher interest rates would be expected to attract capital, strengthening the relevant currency in proportion.

However, research suggests that the effect of interest rate changes on exchange rates depends on whether the currency depreciation is expansionary or contractionary (Sanchez, 2008). Expansionary depreciation is when a weak currency leads to a boost in the nation's exports. In contractionary scenarios, a weak currency has an overall negative impact on the economy. This may cause central banks to raise interest rates to counteract the negative economic effects of depreciation, further demonstrating the non-linear dynamics at play. Investors might seek safe-haven currencies regardless of interest rate differentials.

Interest rates are just one of many financial tools and markets that affect FX rates. Another such market is the stock market. The relationship between stock market performance and exchange rates is complex and multifaceted. A thriving stock market often signals a healthy domestic economy, potentially attracting foreign investors seeking higher returns. This increased inflow of foreign capital can strengthen the demand for the domestic currency, leading to its appreciation. Conversely, a declining stock market may indicate economic difficulties, prompting investors to shift their capital to safer assets or currencies, resulting in a depreciation of the domestic currency. Ajayi and Mougoué (1996) found that rising stock prices can lead to a short-term decrease in the domestic currency's value, due to investors anticipating inflation in a strong stock market.

However, it's crucial to note that correlation does not imply causation. Various factors, including interest rates, inflation, and global economic events, also play significant roles in shaping exchange rates. Studies have found varying degrees of correlation between stock markets and forex rates, with the strength of the relationship often depending on specific currency pairs and prevailing economic conditions.

2.3 ARIMA

2.3.1 Introduction to ARIMA

Autoregressive Integrated Moving Average (ARIMA) models use statistical techniques and are used extensively to forecast time series data. Data is considered time series if it contains information which is collected at regular intervals over a specific period. This allows changes to be tracked and trends to be identified over time. This means the order in which the observations are recorded is vital. Forex historical data is a prime example of time series data, as it records exchange rates between currencies at specific intervals, such as hourly, daily, or monthly. ARIMA models offer a structured way to understand and forecast these patterns.

2.3.2 ARIMA Theory

ARIMA models combine three key components:

- Autoregressive (AR): This part of the model uses past values of the time series to predict future values. For instance, in exchange rate forecasting, an AR component might use the past three days' exchange rates to predict the fourth day's rate. The order ('p') of the AR component indicates how many lagged values of the series are used in the prediction. This is represented by the formula

$$Y_t = c + \phi_1 Y_{t-1} + \phi_2 Y_{t-2} + \cdots + \phi_p Y_{t-p} + \varepsilon_t$$

  Where:
  - Yt is the value of the time series at time t
  - c is a constant
  - $\phi 1, \phi 2, \ldots, \phi p$ are the coefficients of the autoregressive terms
  - $Yt-1, Yt-2, \ldots, Yt-p$ are the lagged values of the time series
  - εt is the error term at time t

- Integrated (I): A central requirement for applying ARIMA models is that the time series should be stationary. Stationary data exhibit a consistent mean and variance over time. The 'I' component addresses non-stationarity through differencing. Differencing involves calculating the differences between consecutive observations, which often leads to a more stationary time series

suitable for analysis with AR and MA components. The degree of differencing required is denoted by 'd'. The formula for first order differencing is

$$Z_t = Y_t - Y_{t-1}$$

Where Z_t is the differenced series.

- Moving Average (MA): This component improves predictions by incorporating information from past forecast errors. Essentially, the MA component attempts to smooth out random fluctuations by considering the error from previous forecasts. The order ('q') of the MA component indicates how many past errors are included in the model. This is denoted by the formula

$$Y_t = c + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2} + \cdots + \theta_q \varepsilon_{t-q}$$

Where:

- $\varepsilon t, \varepsilon t - 1, \varepsilon t - 2, \ldots, \varepsilon t - q$ are the error terms at time t and previous periods
- $\theta 1, \theta 2, \ldots, \theta q$ are the coefficients of the moving average terms

Together, these components create the ARIMA (p,d,q) model. Selecting the optimal values for p, d, and q is an essential part of ARIMA modelling. This frequently involves techniques like examining autocorrelation function (ACF) and partial autocorrelation function (PACF) plots. These plots help visualise patterns and correlations within the time series data, guiding the choice of appropriate ARIMA parameters. The flow diagram below visualises this process.

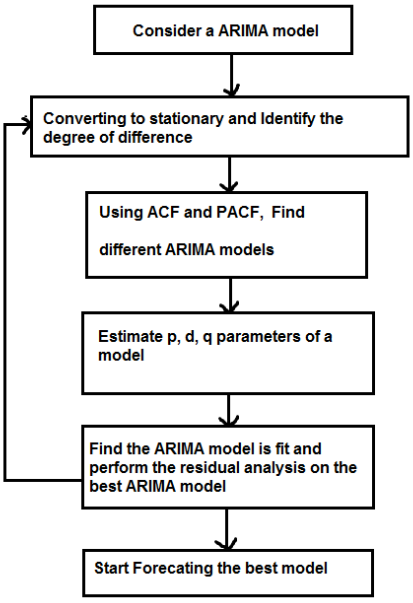

Figure 2.1 - Flow Diagram of ARIMA Model (Shakti et al., 2017)

The effectiveness of ARIMA models in the field of foreign exchange rate prediction is further supported by research conducted on the Vietnamese Dong (VND) against the US Dollar (USD) (Mong and Ngan, 2016). This study demonstrated the suitability of ARIMA in forecasting short-term exchange rates. By using an ARIMA model on data from 2013 to 2015, researchers accurately forecasted VND/USD exchange rates for the subsequent year. Errors were lowest when predictions were made for less than 15 days in the future. Another study found that ARIMA models outperformed traditional machine learning techniques such as Neural Networks and Fuzzy NEuron models when predicting Indian Rupee exchange rates against various currencies (Babu and Stock, 2015). Additionally, This highlights the potential variability in model performance under different market conditions and currency pairs.

2.3.3 Offline vs. Online ARIMA

ARIMA models offer flexibility in how they are implemented, with two primary approaches: offline and online. Offline ARIMA involves training the model on a fixed historical dataset. Once trained, the model's parameters remain static and are used to generate predictions. However, it does not incorporate new data as it becomes available. Offline ARIMA is suitable for scenarios where the time series exhibits stability, and the patterns learned from historical data are expected to persist in the future. For instance, in an industrial setting where sensor data from a film-blowing machine exhibits consistent behaviour, offline ARIMA could be used for real-time data imputation (Fernandes et al., 2024). The paper mentions that the film-blowing machine process is used as an example because it demonstrates "consistent behaviour." This implies that the sensor data from the machine likely follows a relatively stable pattern over time.

On the other hand, online ARIMA models continuously adapt to incoming data. This is often achieved using a rolling window approach, where the model is retrained with the most recent observations. This adaptability is advantageous when the time series experiences shift in trends, seasonality, or overall behaviour. However, online ARIMA models can potentially be less stable than their offline counterparts, as updates based on limited recent data might introduce more variability into the predictions. In the context of stock price prediction, online ARIMA models could be particularly useful for handling the dynamic and ever-changing nature of financial markets. By continuously updating the model with the latest stock price data, online ARIMA can capture sudden market shifts, news events, or other factors that may impact stock prices in real-time (Ariyo, Adewumi and Ayo, 2014). This adaptability allows for more responsive and up-to-date predictions, which can be crucial for investors and financial analysts looking to make informed decisions in a rapidly

changing market environment. In the context of FX prediction, online ARIMA is particularly beneficial due to the dynamic nature of currency markets. Factors like interest rate decisions, economic data releases, and geopolitical events can cause rapid fluctuations in exchange rates. By continuously updating with the latest data, online ARIMA has the potential to better capture these shifts, leading to more accurate and timely predictions.

2.3.4 Advantages and Limitations of ARIMA

ARIMA models have the potential to accurately forecast FX rates using historical data. Their strengths lie in their ability to handle both trends and seasonal components, characteristics which are often observed in FX markets (Box et al., 2016). By modelling the relationships between past values and error terms, ARIMA models can reveal the underlying patterns and dynamics governing FX rate behaviour (Hyndman and Athanasopoulos, 2016). ARIMA models are also well established, having been invented in the 1970s (Sato, 2013). This combined with a robust mathematical foundation means there are many case studies showing ARIMA models effectiveness in forecasting time series data (Pankratz, 1983). Empirical evidence shows how a hybrid neural network and ARIMA model can been used to increase accuracy of time series forecasts (Zhang, 2003). This is in contrast to the two approaches being compared, and instead combining statistical and machine learning techniques. One example of this shows ARIMA models combined with machine learning techniques used to make stock price predictions (Pai and Lin, 2005). As the stock market has many of the same characteristics of the FX market, this can be used as a proof of concept for the use of ARIMA models in this project.

However, it is essential to be aware of ARIMA's limitations when considering its suitability for FX rate predictions. One such limitation is the model's reliance on the assumption of linearity. If the currency exchange rates exhibit highly non-linear patterns, the accuracy of ARIMA forecasts might diminish (Hyndman & Athanasopoulos, 2018).

A fundamental requirement of ARIMA models is the need for input data to be stationary. A stationary time series exhibits a consistent mean and variance over time. This means that the statistical properties of the series do not change with time, allowing the ARIMA model to make reliable predictions. This is important to note as FX historical data is non-stationary, as it can exhibit trends or changing volatility. However, if a time series is non-stationary, differencing is often applied to achieve stationarity (Box et al., 2016). Differencing involves subtracting a previous value of the time series data point from the current value, and in doing so trends and

seasonality are removed from the data. The resulting series will exhibit a constant mean and variance over time, making it stationary and suitable for ARIMA modelling (Solo, 1984). For example, first-order differencing involves calculating the differences between consecutive observations, transforming trends into more predictable patterns. Higher orders of differencing may be needed to obtain stationarity.

Finally, traditional ARIMA models are inherently offline. They might not readily adapt to drastic shifts in FX rate patterns, potentially necessitating re-training or the integration of adaptive techniques to improve their responsiveness (Pankratz, 1983). The dynamic nature of foreign exchange markets, where prices fluctuate rapidly in response to news events, economic indicators, and trader sentiment, necessitates models that can adapt quickly. Online ARIMA models, with their ability to update themselves with new data, are particularly valuable in this context as they can learn from recent changes in FX rates. This will capture any shifts in trends, seasonality, or volatility, leading to more accurate and timely forecasts. When testing ARIMA models for this project, it would be useful to compare the results of an offline and online ARIMA model.

2.4 LSTM

Long Short-Term Memory Networks (LSTMs) are a subclass of Recurrent Neural Networks (RNNs). RNNs are a neural network architecture designed to process time series data where interdependencies exist among the inputs (Grossberg, 2013). A distinctive feature of RNNs is their internal state mechanism, often referred to as "memory" (Grossberg, 2013). This allows for information to be processed based not only on the immediate input, but also on insights derived from preceding elements within the sequence. In practice, an RNN uses a loop that allows information to persist (Olah, 2015). Information from the previous step is passed to the next step for processing.

While RNNs are theoretically good at handling long-term dependencies, they often encounter the "vanishing gradient problem" (Hochreiter, 1998). In practice, as sequences get longer, the information from earlier inputs can become lost. LSTMs were specifically developed to mitigate the vanishing gradient problem (Hochreiter and Schmidhuber, 1997). Financial time series data, such as historical FX data, exhibit complex temporal dependencies where historical market behaviour can influence future trends. Given this characteristic, LSTMs were deemed a suitable

approach for this project due to their capacity to selectively retain, modify, and transmit information across numerous time steps.

2.4.1 LSTM Architecture

This section explains the LSTMs underlying architecture, which helps in better understanding how and when predictions are made and updated within the loop. The formulas in this section were obtained from Olah (2015).

Cell State (c): The LSTM's heart is the cell state, denoted by c, which acts as a long-term memory, carrying vital information across time steps. It's initially set to zero at the beginning $(c_0 = 0^*)$ equation

LSTMs regulate the flow of information into and out of this cell state using three specialised gate structures. Below is a description of each gate structure and it has been indicated when the cell state is updated:

1. Forget Gate (f): The forget gate f_t, determines what information from the previous cell state should be discarded. It receives the previous hidden state h_{t-1} and the current input x_t, passes them through a sigmoid function (σ), and produces an output between 0 and 1 for each value in the cell state. A '1' signifies "retain this information" while a '0' means "discard it."

$$f_t = \sigma(W_f * [h_{t-1}, x_t] + b_f)$$

   Where:
   - W_f represents the forget gate weight matrix
   - b_f signifies the forget gate bias vector
   - $[h_{t-1}, x_t]$ denotes the concatenation of the previous hidden state and current input.

   A value close to 1 in f_t indicates retaining that specific information, while a value near 0 suggests discarding it.

2. Input Gate (**i**): The input gate i_t controls which new information should be stored in the cell state. It uses a sigmoid function to determine update values and a $\tanh$ function to create a vector of possible new values. It has two parts:

- Candidate Values ($\mathbf{C_t^{\sim}}$): This vector creates potential new values for the cell state based on the previous hidden state and current input. It uses a tanh activation function (φ) to achieve this.

$$C_t^{\sim} = \phi(W_i \cdot [h_{t-1}, x_t] + b_i)$$

  Where:
  - W_i represents the input gate weight matrix
  - b_i signifies the input gate bias vector.

- Gate Values ($\mathbf{i_t}$): This vector determines the significance of these candidate values. It utilises a sigmoid function similar to the forget gate.

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

Finally, the element-wise product of the candidate values and gate values determines the information actually added to the cell state.

$$\text{Update}(c_t) = i_t \cdot C_t^{\sim}$$

3. Cell State Update ($\mathbf{c_t}$):

The cell state is then updated by incorporating the forget gate's output and the new information from the input gate

$$c_t = f_t \cdot c_{t-1} + \text{Update}(c_t)$$

Finally, the element-wise product of the potential output and the gate values forms the actual hidden state output

$$h_t = o_t \cdot \widehat{h_t}$$

4. Output Gate ($\mathbf{o}$): The output gate o_t modulates the information to be passed from the cell state to the hidden state, regulates the information flowing from the cell state to the current hidden state (h_t). It has two steps:

- Output Activation: Similar to the candidate value creation, a hidden layer with a tanh function generates a vector of potential outputs.

$$\widehat{h_t} = \phi(W_o \cdot [h_{t-1}, x_t] + b_o)$$

- Gate Values ($\mathbf{o_t}$): A sigmoid function determines the relevance of these potential outputs.

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$

- This hidden state (h_t) serves as the output for the current time step and becomes the previous hidden state (h_{t-1}) for the next time step. The process iterates through the entire sequence, allowing the LSTM to learn and utilise long-term dependencies within the data.

This process can be visualised by the diagram below:

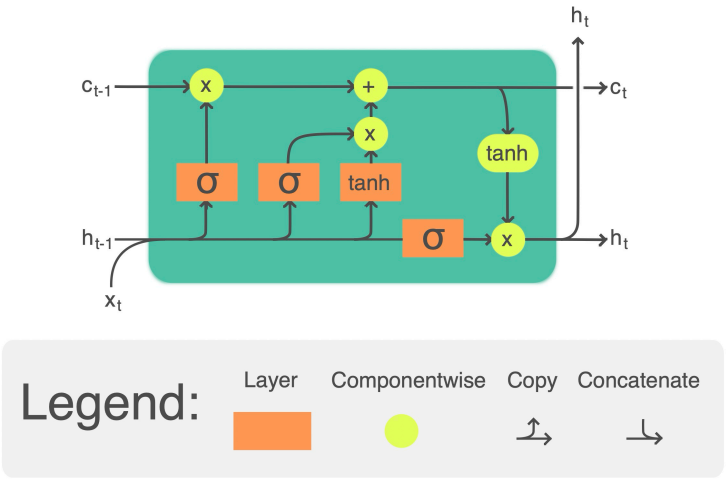

Figure 2.2 - Diagram of LSTM Neural Network (https://en.wikipedia.org/wiki/Long_short-term_memory#/media/File:LSTM_Cell.svg)

2.4.2 Advantages and Limitations of LSTM

LSTMs offer specific advantages for analysing sequences such as FX rates. Their unique "memory cells" facilitate the retention of past information, allowing the model to discern long-term dependencies in FX market dynamics (Hochreiter & Schmidhuber, 1997). Another key advantage is that LSTMs can handle multivariate data. This means multiple time series can be inputted into the model to produce an output. As discussed in (reference forex background here), research has shown there is a link between stock market performance and FX rates. This multivariate attribute of the LSTM could be used to train the model not only on historical FX rates, but also historical stock market data for the same time period.

The sigmoid and tanh activation functions within LSTMs play a crucial role in their ability to effectively model the complex and often non-linear patterns inherent in FX

data (Olah, 2015). The sigmoid function enables the LSTM's gates to regulate information flow, thereby facilitating decisions regarding what data to retain, forget, or update within the network's memory (Farzad, Mashayekhi and Hassanpour, 2017). Meanwhile, the tanh function acts to rescale values between -1 and 1, introducing non-linearity and aiding in the identification of complex trends beyond simplistic linear relationships (Farzad, Mashayekhi and Hassanpour, 2017). This architectural advantage promotes adaptability in volatile environments, as it allows the LSTM to only propagate information that is important to the prediction. Ray et al. (2022) used this advantage by combining an LSTM with an ARIMA model to predict volatile agricultural price series. This design also mitigates the models susceptibility to short-term fluctuations that can prove detrimental to the performance of other time-series modelling approaches. This is evidenced by research showing LSTMs performing better than other time series forecasting models. Sirisha et al. (2022) compared ARIMA, SARIMA and LSTM models to each other when predicting profits from time series data. It was found the LSTM performed best, with an accuracy of 97.01% with ARIMA performing the worst at 93.84% accuracy. Earlier research showed an LSTM model reduced error rates up to 87% compared to an ARIMA model when forecasting time series data (Siami-Namini et al., 2018).

However, understanding LSTM's limitations is equally important for optimal deployment. LSTMs can be prone to overfitting, particularly with smaller datasets. This means the model may "memorise" noise or specific patterns within the training data, hindering its ability to generalise effectively to unseen real-world data (Srivastava et al., 2014). They are computationally expensive, particularly compared to models like ARIMA (Siami-Namini et al., 2018). This increased demand for resources can prolong training times and necessitate special hardware. Vladimir Rybalkin et al. (2020) conducted research on this and produced empirical evidence showing significant speedups (up to 84x) and energy efficiency improvements (up to 1238x) when using hardware architecture specifically designed for RNNs compared to traditional GPU implementations. However, this type of specialised hardware is out of the scope of this project and runtimes will therefore have to be taken into consideration when training the LSTM model.

2.5 XGBoost

XGBoost (Extreme Gradient Boosting) is a machine learning algorithm and comes under the subset of gradient boosting decision trees. The distinguishing factor of XGBoost lies in its speed, scalability, and performance as compared to other gradient boosting algorithms (Chen and Guestrin, 2016). It achieves this through a series of meticulously crafted optimisations. To understand these optimisations, it is

imperative to understand what decision trees and gradient boosting are, and how they interlink.

2.5.1 Decision Trees

Decision trees are learning models that resemble a flowchart. They operate by recursively partitioning the data space into subsets based on features (attributes) that best distinguish between the target variable's classes or values. This process continues until a stopping criterion, such as reaching a certain depth or achieving sufficient purity within a subset, is met. The resulting tree-like structure allows easy interpretation of the results, allowing each step of the decision-making process employed by the model to be understood.

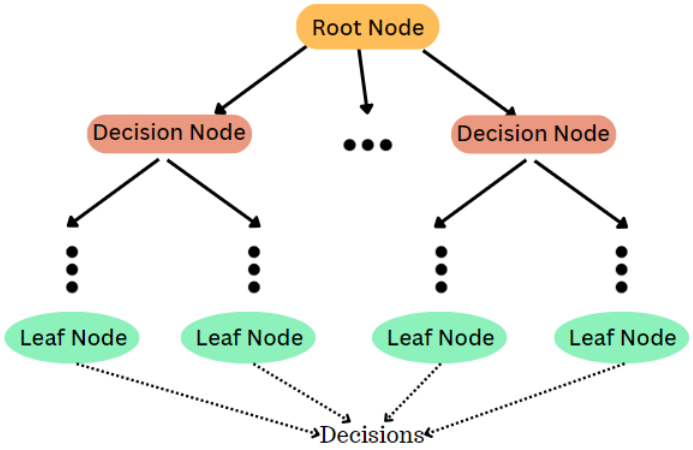

Figure 2.3 - Diagram of a Decision Tree (Nidhi, 2023)

2.5.2 Gradient Boosting

Gradient boosting is an iterative machine learning technique that involves building an ensemble of weak learners, usually decision trees (Natekin and Knoll, 2013). The fundamental principle lies in sequentially training new models, and each model aims to correct the errors of its predecessors. Starting at the root node, data is split based on features following a series of yes/no questions. This process continues until a leaf node is reached, representing the final prediction. This process gradually minimises a loss function, leading to improved overall predictions.

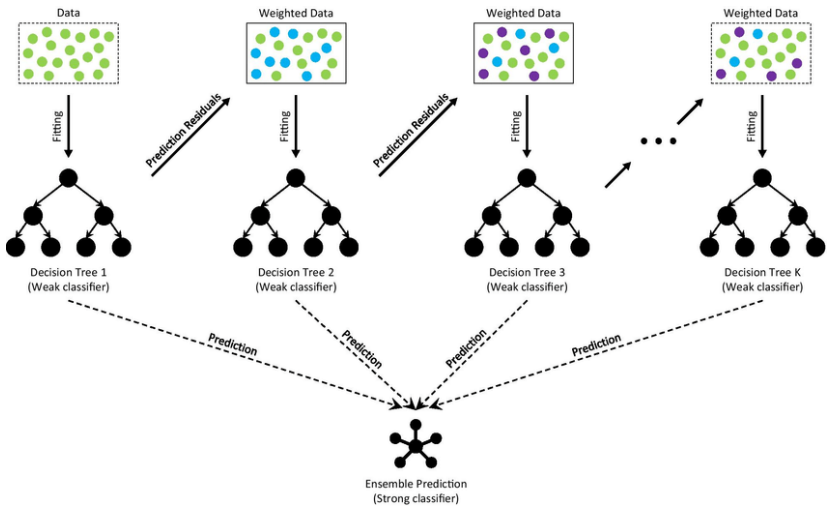

Figure 2.4 - Flowchart of the gradient boosting decision tree (Jia and Xue, 2022)

2.5.3 XGBoost: A Gradient Boosting implementation

XGBoost was developed as a highly optimised implementation of gradient boosting decision trees with the aim of being highly scalable and computationally efficient (Chen and Guestrin, 2016). Feature importance scoring to identify the most influential features, and efficient parallel processing techniques to expedite training on large datasets both help with the speed and scalability of XGBoost.

The foundation of XGBoost's ensemble structure lies in decision trees. Gradient boosting carefully coordinates the iterative construction of these trees, successively enhancing predictive accuracy. XGBoost, as a sophisticated evolution of gradient boosting, incorporates further optimisations to further increase its efficiency and effectiveness beyond the standard implementation.

2.5.4 Key Innovations and Optimisations in XGBoost

One major optimisation is the use of a regularised objective function to prevent overfitting. The XGBoost objective function guides the model's learning process by determining how well a given ensemble of decision trees fits the training data. It guides the model towards better predictions by combining a loss term and a regularisation term.

General Formula:

$$\text{Objective}(T) = \sum l(y_i, \hat{y}_i) + \sum \Omega(f)$$

Components:

$\mathbf{l(y_i, \hat{y}_i)}$: Loss Function

This term quantifies the discrepancy between the true values (y_i) and the predicted values ($\hat{y}_i$) produced by the model. XGBoost supports various loss functions suitable for different problem types. For example, for a regression problem Mean Absolute Error (MAE) could be used, yet for a classification problem logistic loss could be employed.

$\mathbf{\Omega(f)}$: Regularisation Term

This term penalises the complexity of the model (f), which consists of the ensemble of decision trees. Regularisation helps to prevent overfitting by favouring simpler models that are likely to generalise better to unseen data. XGBoost employs regularisation techniques such as L1 and L2 regularisation. These essentially work by penalising the sum of the absolute values of the weights, and the sum of the squares of the weights to prevent overfitting.

2.5.5 Second-Order Approximation

XGBoost distinguishes itself by using a second-order Taylor approximation of the loss function when building new decision trees during the boosting process. This approximation provides more precise information about the direction and magnitude of change needed to improve the model's predictions.

For example, XGBoost could be used for a regression task (as would be the case with FX forecasting) with squared error as the loss function and L2 regularisation. The objective function would therefore be as follows:

$$\text{Objective}(T) = \sum (y_i - \hat{y}_i)^2 + \lambda \sum |w|^2$$

The first term measures the sum of squared differences between true values and predicted values. The second term ($\lambda \sum |w|^2$) is the L2 regularisation, with λ controlling its strength. This term penalises large weights (w) within the decision trees, pushing the model towards simplicity.

2.5.6 Advantages and Limitations

XGBoost offers several advantages for FX forecasting. One key advantage is its ability to handle complex relationships within data, making it suitable for capturing intricate patterns in financial data (Pin & Zhang, 2018). XGBoost has been shown to outperform other models such as random forest, Bayesian, and k-Nearest Neighbours in terms of speed and prediction accuracy, making it a valuable tool for accurate and efficient forecasting (Pin & Zhang, 2018). Additionally, the XGBoost model can be easily extended to include more features, further enhancing its capability to capture complex patterns and improve forecast accuracy (Wang et al., 2023).

XGBoost is also known for its high prediction accuracy, as demonstrated in various studies. For instance, in sales prediction for chain stores, the XGBoost linear regression model has shown high accuracy and provided scientifically based predictions, indicating its reliability for forecasting sales trends (Li, 2023). In the context of stock index futures price forecasting, an XGBoost-based multivariate deep learning framework has been proposed to enhance predictive accuracy, showcasing the algorithm's effectiveness in financial markets (Wang et al., 2022).

Furthermore, XGBoost has been successfully applied in predicting intra-daily market trends in the futures market, highlighting its ability to produce satisfactory forecasting performance in financial scenarios (Zhu et al., 2022). The futures market is an exchange where a particular commodity is agreed to be bought or sold at a particular price at a particular point in the future. With currencies being one of the main commodities on the futures market, this forecasting ability can be easily translated to the use case of this project.

Research focusing on the prediction of foreign exchange prediction (specifically USD/IDR) reinforces XGBoost's potential for currency forecasting (Islam, Sholahuddin and Abdullah, 2021). Their work leveraged historical exchange rate data and the XGBoost algorithm within a KDD framework. Despite the volatility of the USD/IDR exchange rate, the study achieved promising accuracy with RMSE and MAPE values within acceptable ranges. The positive outcomes of this research, alongside broader applications of XGBoost, open avenues for investigating its performance in the field of currency rate predictions.

Despite its advantages, XGBoost also has limitations that need to be considered. One limitation is the need for careful hyperparameter tuning to achieve optimal performance. The accuracy of the XGBoost model heavily depends on hyperparameters such as the number of gradient boosted trees, maximum tree depth,

and boosting learning rate, among others (Ngoc et al., 2023). This tuning process can be time-consuming and requires expertise to ensure the model's effectiveness.

Another notable disadvantage is the sensitivity of XGBoost to noisy data and outliers, which can lead to overfitting and reduced generalisation performance (Wang et al., 2023). Machine learning algorithms like XGBoost rely heavily on the quality of training and testing data; inconsistencies between different databases may hinder the generalisability of the XGBoost prediction model, impacting its effectiveness in financial applications (Wang et al., 2023).

In conclusion, while XGBoost offers significant advantages for financial forecasting, including high accuracy and the ability to capture complex patterns, it is essential to consider its limitations. Sensitivity to noisy data, the need for hyperparameter tuning, and the risk of overfitting are important factors to be mindful of when utilizing XGBoost in financial applications. By addressing these limitations through careful data preprocessing, model tuning, and regularisation techniques, the effectiveness of XGBoost in financial forecasting can be optimised.

3 Methodology and Implementation

In the interest of data accessibility and ease of integration, `Yahoo! Finance' (Yahoo Finance, 2023) was chosen as the source for historical foreign exchange (FX) data within this project. Yahoo! Finance offers a publicly available repository of FX and stock market data online. Yahoo! Finance discontinued their own API in 2017, and the yfinance Library (Aroussi, 2023) was created to scrape financial data from Yahoo! Finance for use in Python programs. This format allowed for seamless integration with the Python programming language libraries utilised for data analysis thereby streamlining the data acquisition process.

```
# Specify start and end dates
start_date = "2023-01-29"
end_date = "2024-01-29"

# Download USD-JPY data
currency_data = yf.download("USDJPY=X", start=start_date, end=end_date)["Adj Close"]
```

Figure 3.1 - Code downloading FX data

The data was downloaded using the "USDJPY=X" ticker and provided daily closing data for the currency pair for a year. However, the start and end dates were easily adjustable which allows for many experiments to be run over different lengths of data. This was essential as it allowed the same models to be tested over different time periods with varying amounts of data. The models accuracy can then be tested and compared to see how far in advance the optimal prediction can be made.

Throughout this project across all models, an 80/20 train-test split was used. This decision was founded on the need to strike a balance between providing the machine learning models with a substantial dataset for learning, while simultaneously retaining an adequate portion of data for robust evaluation. The 80% allocation to the training set ensures that the models are exposed to a diverse range of patterns and examples, facilitating effective generalisation. The remaining 20% dedicated to the test set offers a sufficiently sized, unseen dataset on which to measure model performance in a manner that simulates real-world deployment. While there is no universally perfect ratio, the 80/20 split is a widely acknowledged standard across the field of machine learning, offering a suitable compromise between these two critical requirements.

3.1 ARIMA

3.1.1 Data Pre-processing

Missing Values

For both ARIMA models (online and offline), the raw data underwent a two-stage pre-processing and cleaning process to ensure its suitability for machine learning analysis. The first stage addressed potential missing values.

There are two main techniques for handling missing data, imputation and interpolation. Imputation replaces missing values with substituted values, often using information from existing data points, such as the mean or median. Interpolation, on the other hand, estimates missing values by fitting a curve or a function through existing data points. In time series data, values tend to exhibit autocorrelation, meaning there is often a strong correlation between adjacent time points. In the case of financial data, FX rates generally exhibit trends and seasonality. Forward filling imputation leverages this characteristic by carrying the previously observed value forward, assuming a degree of continuity which is often realistic in FX markets (Mara et al., 2022).

```
# Specify start and end dates
start_date = "2000-01-29"
end_date = "2024-01-29"

# Download USD-JPY data
currency_data = yf.download("USDJPY=X", start=start_date, end=end_date, progress=False)["Adj Close"]

# Check for missing values
print(currency_data.isnull().sum())
```

```
0
```

Figure 3.2 - checking for missing values in raw data

Upon inspection no missing values were found in the dataset from January 2000 to January 2024, so this step can be bypassed entirely. This was expected as Yahoo! Finance is a reputable source for FX data, however it is good academic practice to check for completeness of data.

Winsorization

Having addressed potential missing values, the second stage of data preparation focused on outlier management through a technique called Winsorization. Through Winsorization extreme values were replaced with values closer to the central tendency of the data distribution, typically by setting them to a predetermined

percentile. In this case the 5th and 95th percentiles were chosen as this created a symmetrical adjustment, as it treated high and low outliers equally.

Compared to trimming, the approach of Winsorisation reduced the influence of outliers without completely removing them (Lusk, Halperin and Heilig, 2011). This was especially relevant when looking at financial data, as it is likely spikes and dips will occur in the market.

```
Lower threshold: 130.23254852294923
Upper threshold: 149.99444885253905
```

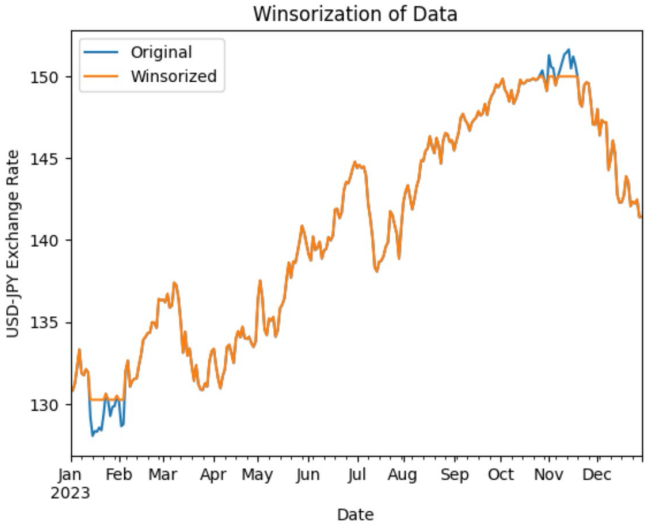

Figure 3.3 - graph showing winzorized data

3.1.2 Stationarity (ADF) Testing

ARIMA models require input data to be stationary, to ensure the statistical properties (such as mean and variance) are constant over time. This allows the model to effectively recognise trends and seasonality within the data, therefore allowing predictions to be made on future data. To statistically test the stationarity of the data, the Augmented Dickey-Fuller (ADF) test was performed.

```
# Pass the winsorized_df to the adfuller test
adf_result = adfuller(winsorized_df)

# Print the test statistic and p-value
print("ADF Statistic:", adf_result[0])
print("p-value:", adf_result[1])
```

```
ADF Statistic: -1.4425731182215498
p-value: 0.5617045563421605
```

Figure 3.4 - ADF Test to check stationarity

In this test, the null hypothesis is that the data is non-stationary (it has a unit root), and the alternative hypothesis is that the data is stationary. Since the p-value is

greater than the significance levels of 0.05, we fail to reject the null hypothesis. Therefore, we do not have enough evidence to conclude that the data is stationary. This was to be expected as historical exchange rate data is usually non-stationary (Wu, 1996), however there are techniques which can be used to address non-stationarity in the data.

3.1.3 Differencing

Differencing aims to transform the non-stationary data into a stationary format, making it more suitable for machine learning algorithms. It achieves this by calculating the difference between consecutive data points in the time series. This involves subtracting the previous value (t-1) from the current value (t) at each time step. Mathematically, the differenced value (y't) is represented as:

$$y't = yt - yt - 1$$

This removes the trend component from the data, focusing on the rate of change between exchange rates. The figures below illustrate pure historical data and differenced data during the same one-year time period.

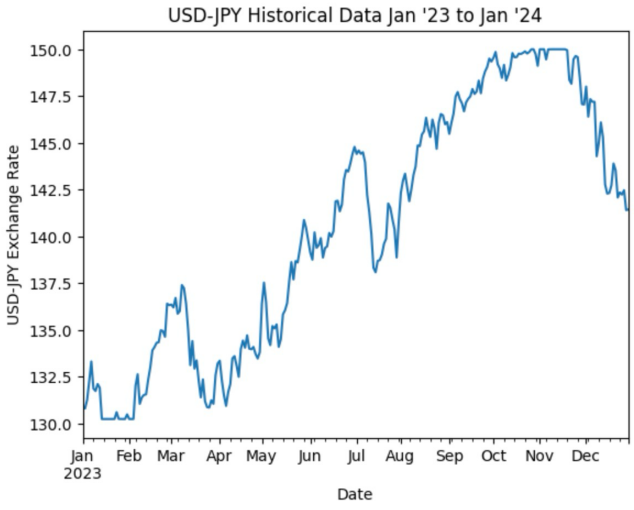

Figure 3.5 - Raw winzorized historical FX data

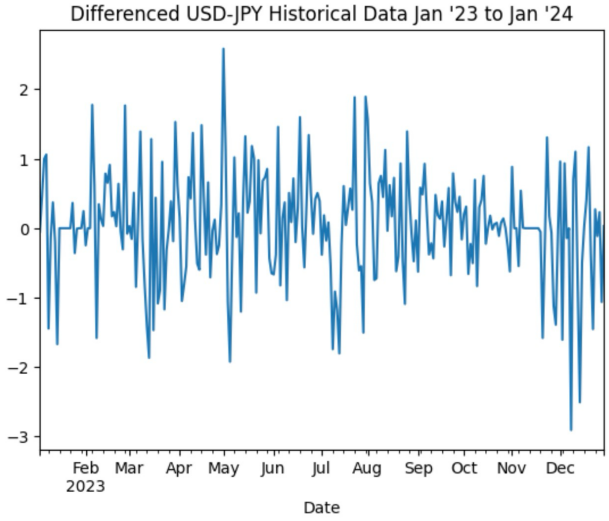

Figure 3.6- Differenced historical FX data

The ADF test can once again be performed to verify the stationarity of the differenced data.

```
# Apply one difference (d = 1)
differenced_df = winsorized_df.diff().dropna()  # Drop the first NaN value

# Check for stationarity again
adf_result_diff = adfuller(differenced_df)

print("ADF Statistic (differenced):", adf_result_diff[0])
print("p-value (differenced):", adf_result_diff[1])
```

```
ADF Statistic (differenced): -12.41325329184544
p-value (differenced): 4.2964859226992446e-23
```

Figure 3.5 - ADF test performed on differenced data

As can be seen above, the ADF statistic after applying differencing is now significantly more negative and the p-value is much smaller than 0.05, indicating strong evidence against the presence of a unit root at the 5% significance level. This suggests that differencing has successfully made the data stationary. As differencing was only applied once, the D value is 1.

3.1.4 Building the model

As discussed in the background section, ARIMA models need p, d, and q values. The number of autoregressive terms (p) and the number of moving average terms (q) can be determined using ACF (Autocorrelation Function) and PACF (Partial Autocorrelation Function) plots.

The ACF plot depicts the correlation of a time series with itself at various lags (time differences). For this project, the USD-JPY exchange rate at day 'x' is compared with its value at different points in the past (x-1 day ago, x-2 days ago, etc). The ACF plot shows how strong these correlations are at each lag. PACF isolates the impact of a specific lag on the current value, excluding the influence of earlier lags. This helps identify significant lags with direct predictive power. By visually analysing past data dependencies, optimal 'p' and 'q' values can be easily determined.

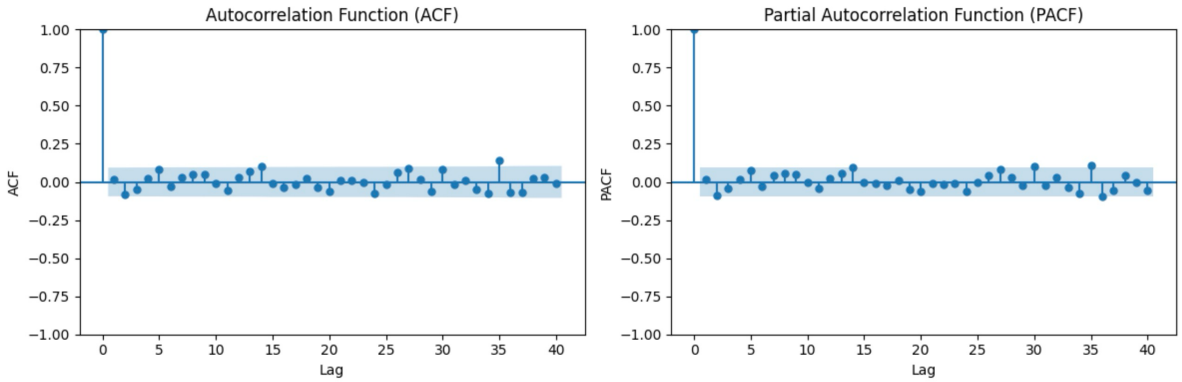

Figure 3.6 - ACF and PACF plots to determine ARIMA p and q values

On the ACF plot, there is a significant spike at lag 1, meaning the p value is 1. Also, the PACF plot significantly spikes at lag 1, indicating a potential moving average term (MA). This means the q value is 1. The remaining spikes are within the confidence interval bounds, highlighted. By the shaded blue region on the plots. These are values between -0.25 and 0.25 and are not statistically significant, therefore they are ignored. Based on the ACF and PACF plots, ARIMA (1,1,1) was used.

3.1.5 Model Fitting

Offline Model

The model was fitted to the training set of winsorized and differenced data. Once the models have been trained on the differenced data, their predictions were also given in a differenced form. To turn this into a useable forecast, the output was inverted to produce a level of the original data. This process reverses the differencing applied earlier to achieve stationarity, and the resulting forecast can then be interpreted in the context of the original time series.

```
inverted_forecast = pd.Series(forecast, index=test_data.index)
for i in range(len(inverted_forecast)):
    inverted_forecast[i] += differenced_df[train_size + i - 1]  # Backshift to original scale
```

Figure 3.7 - Code inverting differenced forecast

To assess the goodness of fit and check for potential issues, the output of model_fit.summary() was reviewed. This provided important statistics such as the p-values of the model coefficients and indicators of model adequacy like the AIC (Akaike Information Criterion), which were later used to compare the online and offline models.

```
# Print model summary for diagnostics
print(model_fit.summary())
```

```
                               SARIMAX Results
==============================================================================
Dep. Variable:                  Close   No. Observations:                  347
Model:                 ARIMA(1, 1, 1)   Log Likelihood                -486.543
Date:                Tue, 30 Apr 2024   AIC                            979.086
Time:                        00:47:21   BIC                            990.626
Sample:                    05-03-2022   HQIC                           983.681
                         - 08-30-2023
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1         -0.7885      0.256     -3.078      0.002      -1.291      -0.286
ma.L1          0.8345      0.230      3.622      0.000       0.383       1.286
sigma2         0.9748      0.049     19.766      0.000       0.878       1.071
===================================================================================
Ljung-Box (L1) (Q):                   0.11   Jarque-Bera (JB):               183.09
Prob(Q):                              0.74   Prob(JB):                         0.00
Heteroskedasticity (H):               0.84   Skew:                            -0.70
Prob(H) (two-sided):                  0.34   Kurtosis:                         6.28
===================================================================================
```

Figure 3.8 - model_fit.summary() output

Online Model

In contrast to the offline approach, the online ARIMA model adapted to evolving data patterns by fitting models within a rolling window. The parameters window_size and step_size controlled the amount of historical data used for each model fit and the frequency of forecast updates, respectively.

In this case, a window size of 75 and step size of 1 were selected. The window size provides a sufficient amount of historical data for the model to learn from while allowing for reasonably frequent forecast updates. The choice of window_size and step_size involves a trade-off, as a larger window captures more historical data but might be less responsive to recent changes. A smaller window is more responsive but may not capture long-term trends as effectively. The chosen values (75 and 1) were chosen through experimentation with different values.

A loop iterated through the data starting from the end of the training set. Within each iteration, a train_window was extracted, and an ARIMA model was fitted to the training window data. The order (p, d, q) used for the online model could differ from the offline for each window. This flexibility allows the online model to better adapt to local trends or non-stationarity that might emerge within specific windows.

A forecast was then generated for the next step_size time steps using the fitted model. To ensure robustness, a try-except block was employed. This block attempted to fit an ARIMA model to the current data window. If the fitting process failed, the except block handled the error. In these failure scenarios, a placeholder value (NaN) would be inserted into the forecast results, serving as an indicator that a reliable forecast could not be generated for that specific window. While this placeholder mechanism was not triggered in this specific case, it remains a crucial safeguard against unexpected data issues that could compromise the forecasting process.

Finally, after iterating through all windows, the individual forecasts were concatenated to form a single series for evaluation purposes. As with the offline model, this forecast had to be inverted to return to the original data scale.

```
window_size = 75
step_size = 1

online_forecasts = []
online_rmses = []

# Process data in a rolling window
for i in range(train_size, len(winsorized_df) - step_size + 1, step_size):
    train_window = winsorized_df[i - window_size : i]
    test_window = winsorized_df[i : i + step_size]

    try:
        model = ARIMA(train_window, order=(15, 1, 1))  # Retrain model with each window
        model_fit = model.fit()

        forecast = model_fit.forecast(steps=step_size)[0]  # Forecast one step ahead
        online_forecast = model_fit.forecast(steps=step_size)  # For plotting

        inverted_forecast = pd.Series(forecast, index=test_window.index)
        for j in range(len(inverted_forecast)):
            inverted_forecast[j] += differenced_df[i + j - 1]  # Backshift to original scale

        # Combine forecasts
        online_forecasts.append(inverted_forecast)

        # Calculate and store metric
        online_rmse = mean_squared_error(test_window, inverted_forecast, squared=False)
        online_rmses.append(online_rmse)

    except:
        print("ARIMA fitting failed for window", i)
        online_forecasts.append(pd.Series(np.nan, index=test_window.index))
        online_rmses.append(np.nan)

# Combine online forecasts into a single Series
online_forecasts = pd.concat(online_forecasts)
```

Figure 3.9 - Code fitting online ARIMA to window

3.1.6 Critical analysis

It should be noted that ARIMA models require more data pre-processing than the other models compared in this project. For example, one key assumption of the ARIMA model (stationarity) necessitated the use of stationary testing and differencing. The forecast then also had to be inverted such that it was not in a differenced form. If the data given to the model was not stationary, false trends could have been predicted in the forecast with less accurate results overall.

It was also found that the online ARIMA approach was more computationally expensive than the offline approach due to repeated model fitting. This was evidenced by significantly longer runtimes for the online model. This increased computational cost arises because the online method re-estimates model parameters within each rolling window, while the offline model fits parameters only once to the entire training set. However, the online approach proved better suited for capturing evolving trends and seasonality in non-stationary time series data (as will be expanded on in the results section). This adaptability stems from the online ARIMA's ability to update its model to reflect changing dynamics in the data, which is particularly valuable in scenarios like currency exchange rates where market conditions can shift over time.

3.2 LSTM

As discussed in the literature review, there is research that suggests there are links between stock market and FX markets. With the USD/JPY currency pair, it was

decided to test if there was any statistically significant links between the US stock market (S&P 500), Japanese stock market (Nikkei 225) and the FX data for USD/JPY. It was therefore decided to use historical data from the S&P 500 (US stock exchange) and Nikkei 225 (Japanese stock exchange) alongside FX data to train one LSTM neural network. An identical LSTM was also trained only on USD/JPY historical data, allowing for comparison between the two approaches.

3.2.1 Data Pre-processing

In order to train the LSTM on all three data sources together, the data was concatenated into a single pandas DataFrame aligned by date. As with the ARIMA models, missing data points were addressed using forward filling imputation to maintain continuity, followed by the removal of any remaining rows with null values.

Different financial markets operate on different scales which can cause issues during training. Currency pairs such as USD/JPY tend to move in smaller increments compared to stock indices such as the Nikkei 225 (top 225 companies on the Tokyo Stock Exchange) and S&P 500 (500 largest companies listed on US stock exchanges). This is illustrated by the graph below.

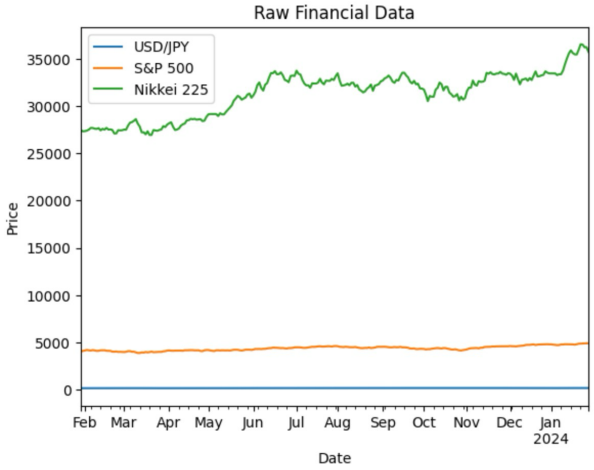

Figure 3.10 - Raw FX and Stock Market Data

To address this, data normalisation techniques were employed. In this case, MinMaxScaler. normalisation was used, scaling the data from all sources to a common range between 0 and 1, ensuring that no single market overpowers the others during the training process. This will allow the LSTM to learn the relationships between the variables more effectively.

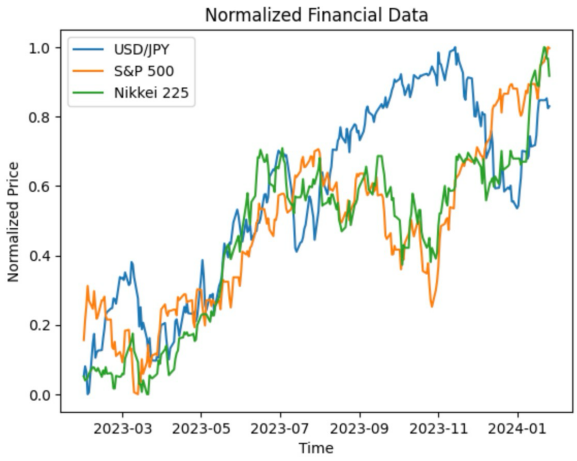

Figure 3.11 - Normalized FX and Stock Market Data

3.2.2 Implementation and Considerations

Several open-source libraries offer robust implementations of LSTM architectures with TensorFlow Keras being employed for this particular implementation. TensorFlow is a powerful computational engine used for deep learning, offering flexibility and granular control. However, it can be challenging to learn. Keras is a high-level API built on top of TensorFlow and provides a user-friendly interface for building and training complex neural networks. This pairing allows researchers to leverage TensorFlow's computational muscle for intricate models while benefiting from Keras' ease of use, making it a popular and well-regarded deep learning toolkit.

3.2.3 Hyperparameter Optimisation

One key consideration when building LSTM models is hyperparameter optimisation or tuning. This involves systematically refining the configuration settings that govern the model's learning process. This optimisation is crucial as it directly influences an LSTM's capability to capture the complex, non-linear patterns inherent in FX markets. Hyperparameter values that are too extreme in either direction can lead to issues relating to accuracy and use of computational resources. A well-optimised LSTM model will be better able to model subtle relationships and dependencies within historical FX data, leading to enhanced forecasting accuracy and robustness.

Hyperparameter tuning can quickly become computationally overwhelming, particularly for complex models like LSTMs. The vast number of potential combinations of hyperparameters creates a vast search space. Evaluating each combination requires training and testing the model, which can be exceptionally time-consuming and computationally expensive. With the limited time and computational resources available, it was necessary to prioritise a focused

hyperparameter search. Therefore, the following three hyperparameters were chosen for tuning:

1. Batch Size

The batch size dictates the number of data samples the model processes before updating its internal weights. Smaller batch sizes lead to more frequent weight updates, which can be beneficial as the model adapts to new information quickly. However, frequent updates can also introduce noise and instability into the training process, potentially leading to poorer convergence. Conversely, larger batch sizes provide smoother weight updates based on a broader sample of the data which can improve stability and accelerate learning. However, excessively large batches can become computationally expensive and may lead the model to get stuck in local optima, hindering its ability to find the best solution.

2. Epochs

An epoch represents one complete iteration of the training dataset where the model is exposed to all available training samples. The number of epochs directly influences how extensively the model trains on the data. Choosing too few epochs risks underfitting, where the model hasn't learned enough from the data to capture the underlying relationships and patterns. This manifests as poor performance on unseen data. Conversely, training for too many epochs can lead to overfitting. In this scenario, the model memorises specific details and noise present in the training data, hindering its ability to generalise to unseen examples. Selecting the optimal number of epochs requires careful monitoring of the model's performance on a validation set to identify the point where learning plateaus and overfitting begins.

3. LSTM Units

LSTM units are the core computational elements within an LSTM layer. They act as memory cells, crucial for capturing long-term dependencies within sequential data. The number of LSTM units directly determines the model's capacity to represent complex patterns in the data. A limited number of units can hinder the model's ability to learn subtle relationships and dependencies within the data, leading to underfitting. Conversely, an excessive number of units can significantly increase the model's complexity. This can lead to overfitting as the model captures not only the underlying patterns but also noise and irrelevant specifics from the training data. Additionally, a larger number of units translates to higher computational cost, requiring more resources to train the model. Finding the optimal number of LSTM

units requires balancing the model's capacity to learn complex patterns with its ability to generalise and computational efficiency.

Batch Size	Epochs	Units	RMSE	MAE	MSE	MAPE	R-squared
1	50	30	1.051	0.872	1.103	3.213	0.861
1	50	40	1.014	0.824	1.020	2.982	0.874
1	50	60	1.073	0.898	1.145	3.344	0.857
1	100	30	1.032	0.856	1.061	3.126	0.867
1	100	40	0.971	0.802	0.957	2.892	0.878
1	100	60	0.994	0.825	0.982	3.018	0.874
16	50	30	1.082	0.904	1.166	3.395	0.853
16	50	40	1.043	0.869	1.082	3.183	0.859
16	50	60	1.066	0.876	1.124	3.266	0.857
16	100	30	1.028	0.844	1.043	3.083	0.869
16	100	40	1.003	0.826	1.007	3.036	0.872
16	100	60	1.035	0.852	1.061	3.164	0.863

Figure 3.12 - Hyperparameter Tuning Results

When analysing the results above, the smallest RMSE, MAE, MSE and MAPE values were chosen, and the highest R-squared values. This allowed the most optimal hyperparameters to be chosen based on the following observations:

- Batch size: A batch size of 1 consistently yielded superior performance compared to a batch size of 16, likely stemming from the sequential nature of FX data. Smaller batches force more frequent weight updates, allowing the LSTM to better capture subtle, time-dependent patterns.

- Epochs: Increasing epochs from 50 to 100 for a batch size of 1 resulted in performance gains. This highlights the model's ability to extract meaningful patterns with longer exposure to the data. However, overly long training could lead to overfitting, an aspect warranting careful monitoring in future experiments.

- LSTM Units: A value of 40 LSTM units appears optimal, demonstrating the importance of balancing model capacity as too few units may hinder the model's ability to learn complex FX dynamics, while too many can lead to overfitting and increased computational cost. It is noteworthy that the difference in performance between 40 and 60 LSTM units is relatively small for most metrics, especially with a batch size of 16. This suggests that in this

specific case, the model might not be overly sensitive to the number of LSTM units within a certain range.

Whilst these results show the hyperparameter tuning for training on 1 year of data, the same hyperparameter tuning was performed for each experiment.

3.2.4 Overfitting

Overfitting is a common pitfall in machine learning, particularly when dealing with complex datasets. An overfit model becomes overly specialised to the nuances of the training data, capturing not only the underlying signal but also random noise. This leads to exceptional performance on the training set but poor generalisation to unseen data, undermining its usefulness in real-world forecasting.

Financial time series, including FX markets, often exhibit noise and complex, non-linear patterns. Powerful models like LSTMs, while capable of learning these patterns, are also susceptible to overfitting. Signs of overfitting in this project were signalled as a significant divergence between performance on training versus validation or testing sets.

Dropout: A Regularisation Technique

Dropout is a powerful regularisation technique designed to counteract overfitting. Introduced by Srivastava et al. (2014), dropout randomly deactivates a proportion of nEurons (often 20-50%) within specific layers of the neural network. This has several profound effects:

- Ensemble-like Behaviour: Each time a training sample is presented, the model effectively trains a slightly different sub-network due to random dropout. This can be viewed as implicitly training an ensemble of numerous smaller networks, leading to more robust representations.

- Preventing Co-adaptation: By randomly removing nEurons, dropout forces the network to learn more independent representations. This reduces the risk of nEurons becoming overly reliant on each other, which can lead to memorising specific, potentially noisy, aspects of the training data.

- Improved Generalisation: As the network can no longer fully rely on specific units, it's encouraged to spread important information across the

network. This leads to a less specialised model that is better equipped to handle new, unseen data.

Implementing Dropout in the LSTM Model

To combat potential overfitting in the LSTM model, dropout layers were strategically added between the main LSTM layer and the output layer. The dropout rate (percentage of nEurons to deactivate) was itself considered a hyperparameter during the tuning process and therefore experimented with the find the optimal rate. A range of 0 (no dropout) to 0.5 (half of all nEurons were deactivated) was explored.

Dropout Rate	RMSE (Run 1)	RMSE (Run 2)	RMSE (Run 3)	Average RMSE
0.0	1.05	1.01	0.99	1.02
0.1	1.00	0.98	1.03	1.00
0.2	0.97	0.96	0.98	0.97
0.3	1.03	0.99	1.01	1.01
0.4	1.10	1.08	1.05	1.08
0.5	1.20	1.18	1.15	1.18

Figure 3.13 - Dropout Rate Optimisation

As can be seen from the results, a dropout rate of 0.2 provided the lowest RMSE which is what was used for the experiment with 1 year of data. It is noteworthy that there is a small difference between dropout values 0 to 0.4, however a significant jump at 0.5 suggests too many nEurons being deactivated would have negatively impacted the models accuracy. Whilst time taken for training was not measured, it was noticed that a higher dropout rate corresponded with a quicker training time. This is because with dropout, a smaller number of nEurons are involved in each forward pass through the network during training, thus reducing the models overall complexity. As with the hyperparameter tuning, this was repeated for each experiment as the optimum dropout rate is dependent on the characteristics of the dataset.

3.3 XGBoost

The XGBoost model was implemented in two different ways for two distinct FX rate prediction tasks. First, an XGBoost regression model was trained to forecast specific USD/JPY exchange rate values, providing a similar output to the previous two models in that a numerical prediction is given. Secondly, an XGBoost classification model was developed to predict the direction of change (up or down) in the exchange

rate. This approach offers insights into market trends without focusing on precise numerical values. A classification approach can provide a distinct perspective on the forces influencing FX markets. It can complement purely numerical predictions by highlighting trend patterns, informing decision-making strategies, and offering insights into broader market sentiment.

3.3.1 Feature Engineering

Many of the core data acquisition and pre-processing steps remained the same for the XGBoost regression model as they were for the LSTM and ARIMA models. Whilst data preprocessing was straightforward in this case, feature engineering was used to further enhance the model. Feature engineering builds upon data preprocessing by creating new features from the existing data or transforming existing features in a way that improves the performance of a machine learning model.

Sliding window

In XGBoost, the create_dataset function is a crucial part of feature engineering for time series forecasting tasks, and is often implemented with a sliding window technique. It transforms raw time series data, such as historical exchange rates, into a structured format suitable for training the model. The following steps are used in the function:

- Sliding/lookback Window: Iterates over the time series, taking a "window" of the past n_prev values at each step. These past values become the features (X) for a single data point. This is similar to the online ARIMA model where multiple ARIMA models were fit within a rolling window.

- Target Variable: The exchange rate immediately following the window becomes the target variable (y) that the model aims to predict.

- Date Indexing: The dates associated with each window are preserved as indices. This maintains the temporal order of the data, which is essential for understanding trends and seasonality within the exchange rates.

By using the create_dataset function, the time-series data was further prepared for XGBoost. This enabled it to learn the intricate patterns between past exchange rates and future values. In order to determine the best lookback window, a range lookback

window sizes were experimented with and the MSE of the corresponding predictions were compared.

Lookback window size	MSE
3	1.89
5	3.04
7	2.99
10	3.08
14	3.58
21	4.32
30	3.59

Figure 3.14 - Lookback window optimisation

Lookback windows representing significant durations within the financial market were experimented with. A 3-day period was included to capture short-term trends relevant to highly volatile currency pairs. Additionally, 5-day and 7-day windows were used to represent a standard workweek and a full calendar week. Larger windows of 10 and 14 days signified two workweeks and two calendar weeks, respectively. Finally, 21-day and 30-day windows were chosen to reflect three weeks and a complete calendar month. While the 3-day lookback window yielded the lowest MSE, the potential for overfitting was higher with such a small window and the currency pair in question (USD/JPY) is not relatively volatile. Ultimately, a 7-day lookback window was selected for the study as this had the second lowest MSE and a lower chance of overfitting to the training data.

Calculate Percentage Changes

New features (usd_jpy_change, sp500_change, n225_change) were created to represent the 1-day percentage change for each financial instrument. These features indicated whether the value increased or decreased, offering insights into market fluctuations. This information was then used to create a threshold that aids in modelling the likelihood of a future FX rate being higher or lower than previous data in the classification model.

Number of decision trees

The n_estimators value controls the number of decision trees within the XGBoost model. A higher n_estimators value can improve model performance but might also lead to overfitting. Conversely, too small a value of n_estimators would be too

simplistic to capture complex patterns in the data. It would struggle to learn the inherent relationships between the features and the target variable, leading to poor performance on both training and unseen data.

n_estimators	Accuracy
50	0.443
100	0.439
200	0.423
500	0.403
1000	0.408

Figure 3.15 - Number of Decision Trees optimisation

It was noted that although 50 decision trees provided the highest accuracy, there was a very small difference between 50 and 100 decision trees compared to higher numbers of decision trees. In order to mitigate the risk of underfitting to the data, it was decided to use 100 as the value for n_estimators. While a slightly higher risk of overfitting might exist with 100 trees, this choice aims to strike a balance between model complexity and performance. Since the accuracy drop-off with 100 trees is marginal compared to 50 trees, the additional decision trees might be able to capture subtle patterns in the data that the simpler model could potentially miss.

4 Evaluation

This section compares results from experiments undertaken on each of the three models. For each model, the data was trained on multiple datasets consisting of financial data going back different lengths of time from 1st January 2024. These lengths were 1 month, 3 months, 6 months, 1 year, 3 years, 5 years and 10 years.

4.1 Statistical Techniques

A crucial aspect of this project was assessing the effectiveness of the trained models. In order to do this, certain metrics and statistical techniques were needed to evaluate and compare each models performance. This section explores the statistical metrics used to evaluate the performance of each prediction model.

4.1.1 Mean Absolute Error (MAE)

The MAE calculates the average of the absolute differences between predicted ($\hat{y}_i$)

and actual (y_i) values. Therefore, the closer the MAE for a model is to 0, the better. the MAE is less sensitive to outliers compared to MSE/RMSE, which can be advantageous for currency exchange rate prediction as unexpected market fluctuations might occur.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

4.1.2 Mean Squared Error (MSE)

MSE measures the average squared difference between the predicted values and the actual values. It penalises larger prediction errors more heavily compared to MAE and can therefore be beneficial for capturing significant deviations from the actual exchange rates. Similarly to MAE, the closer the MSE is to 0 the better the model is performing.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

4.1.3 Root Mean Squared Error (RMSE)

RMSE is derived from the MSE by taking its square root. It provides a measure of the average error in the same units as the original data, allowing easier interpretation compared to MSE. Like MSE, RMSE emphasizes larger errors. The goal is to minimise RMSE, with a value of 0 indicating perfect accuracy.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

4.1.4 Mean Absolute Percentage Error (MAPE)

MAPE is relative as it expresses the absolute error as a percentage of the actual value. It is particularly useful when dealing with time series data where exchange rates can fluctuate at different scales, and it allows for fairer comparisons across varying data ranges. The more the MAPE approaches 0, the better the performance of the model.

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \times 100$$

4.1.5 R^2 (coefficient of determination)

R^2 represents the proportion of variance in the actual data (y_i) that can be explained by the model's predictions ($\hat{y}_i$). It ranges from 0 (no explanatory power) to 1 (perfect fit). For

example, an R^2 score of 0.7 means 70% of the output can be correctly predicted by the model. Therefore, as the R^2 score approaches 1 the model improves.

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

4.1.6 Akaike Information Criterion (AIC)

AIC is a metric used for model selection, particularly for time series models. It balances how well the model fits the data (measured by the likelihood function) with a penalty term for model complexity (number of parameters). The goal is to minimise the AIC score as lower values indicate a more favourable trade-off between these factors.

$$AIC = 2k - 2\ln(L)$$

It is important to note that the AIC is not suitable for comparison between models such as comparing an LSTM to an XGBoost. Instead, it is used to compare different variants of the same model. An example of this is using it to compare an online and offline ARIMA model trained on the same dataset, which is explored in the results section.

4.1.7 Software Implementations

The Python library scikit-learn (sklearn) was employed for various functionalities including data pre-processing, model training, and evaluation metrics. Specific functionalities used include:

- sklearn.preprocessing for scaling and normalisation of data discussed in the previous section
- sklearn.metrics for calculating the MSE, RMSE, MAE and R^2 scores

In each of the following subsections, the tables provided show error metrics for the regression experiments and accuracy metrics for the classification experiments. Cells coloured in green highlight the best performing dataset for a particular metric, whilst those coloured in red highlight the worst.

4.2 ARIMA

4.2.1 Offline ARIMA

Offline ARIMA	RMSE	MSE	MAE	MAPE	R2
1 Month	0.387	0.150	0.263	0.186	0.274
3 Months	3.325	11.055	3.095	2.171	-13.440
6 Months	5.144	26.458	4.420	3.081	-2.956
1 Year	3.735	13.950	2.422	1.683	-0.506
3 Years	7.192	51.726	6.273	4.269	-3.162
5 Years	9.147	83.668	7.000	5.100	-4.500
10 Years	3.885	15.093	7.800	5.800	-6.200

Figure 4.1 - Offline ARIMA Error Results

One advantage of offline training is that it can potentially capture long-term trends and seasonality within the data, especially when using a substantial dataset. However, the data reveals an unexpected set of results: the 1-month dataset yields the best performance across all metrics (RMSE, MSE, MAE, MAPE) with values of 0.387, 0.150, 0.263, and 0.186, respectively. More substantial datasets such as those with 5- and 10-year data performed the worst, with 5 years reaching a Mean Squared Error (MSE) of 83.668. This is counterintuitive, as it was expected for the model to benefit from more historical data for capturing complex financial market behaviour.

One possible explanation for this observation is that financial markets exhibit random fluctuations. While ARIMA models are adept at capturing trends and seasonality, they might struggle to predict these random variations. Using a smaller dataset (1 month) might mitigate the influence of randomness on the model, leading to a better fit for the in-sample data (the data used for training).

The R-squared (R^2) values signify how much of the output can be correctly predicted by the model. A value of 1 would signify a perfect fit. However, all the R^2 values here are negative, indicating a poor fit between the model and the data. This further strengthens the argument that the offline ARIMA model might not be effectively capturing the underlying structure of the currency exchange data, especially for longer timeframes.

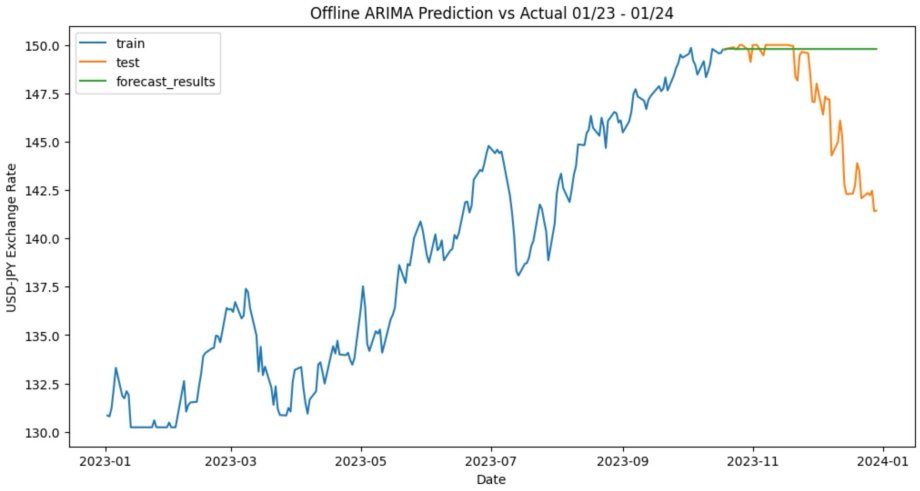

Figure 4.2 - Offline ARIMA prediction on 1 year dataset

A graphical visualisation of the model trained on a 1-year dataset further proves this, with the forecast being the green horizontal line. This is a classic sign of a model that has failed to learn any meaningful patterns from the data, as the model essentially predicts the average exchange rate across the training period for all future time steps, indicating a lack of ability to generalise or capture trends.

4.2.2 Online ARIMA

Online ARIMA	RMSE	MSE	MAE	MAPE	R2
1 Month	1311.741	1720665.090	588.078	415.803	-8353518.704
3 Months	1.992	3.969	1.746	1.222	-4.184
6 Months	0.509	0.259	0.427	0.296	0.961
1 Year	0.366	0.134	0.278	0.190	0.986
3 Years	0.398	0.158	0.271	0.188	0.987
5 Years	0.524	0.274	0.341	0.248	0.993
10 Years	0.429	0.184	0.285	0.213	0.998

Figure 4.3 - Online ARIMA Error Results

For the 1-month dataset, the RMSE is 1311.741, which is considerably high. This indicates a substantial difference between the predicted and actual exchange rates. As the training window expands to 3 months, the RMSE reduces significantly to 1.992, demonstrating a substantial improvement in the model's accuracy.

This pattern continues with further increase in dataset size up to the 1-year dataset. This produced an RMSE of 0.366, however after this point the error metrics began to rise again. This could be a sign of the model overfitting to the data, however the results are still much better than the offline model. By continuously incorporating new data points, the online ARIMA model can adapt to the dynamic nature of the FX market and improve its predictive accuracy.

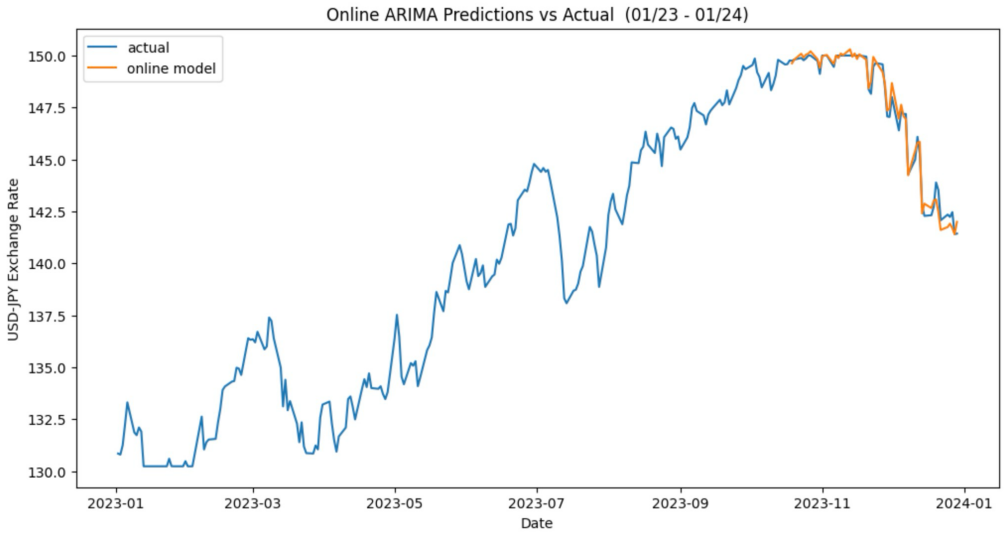

Figure 4.4 - Online ARIMA prediction on 1 year dataset

This is further exemplified by this graph, showing the online ARIMA models prediction compared to actual values on the 1-year dataset.

4.3 LSTM

The following section explores the difference between the LSTM with and without stock market data.

4.3.1 Without stock market data

LSTM	RMSE	MSE	MAE	MAPE	R2
1 Month	1.615	2.608	1.284	0.872	0.384
3 Months	1.302	1.695	1.038	0.705	0.579
6 Months	1.124	1.263	0.89	0.613	0.703
1 Year	0.85	0.723	0.595	0.398	0.848
3 Years	0.864	0.746	0.502	0.339	0.784
5 Years	0.872	0.760	0.535	0.342	0.715
10 Years	0.869	0.755	0.692	0.446	0.736

Figure 4.5 - LSTM trained on FX data error results

The model seems to benefit significantly from more data initially. Compared to a 1-month dataset (RMSE: 1.615, MSE: 2.608), training on 3 months (RMSE: 1.302, MSE: 1.695) and 6 months (RMSE: 1.124, MSE: 1.263) of data leads to a substantial reduction in errors. This indicates the LSTM's capability to learn complex patterns and sequential relationships within the currency exchange data.

However, the trend reverses for longer datasets. There's a slight increase in errors (RMSE: 0.872, MSE: 0.760) for the 10-year dataset compared to the 6-month

dataset. Overall, this LSTM performs best with the 1-year data set and also fairly well with the 3 year data set.

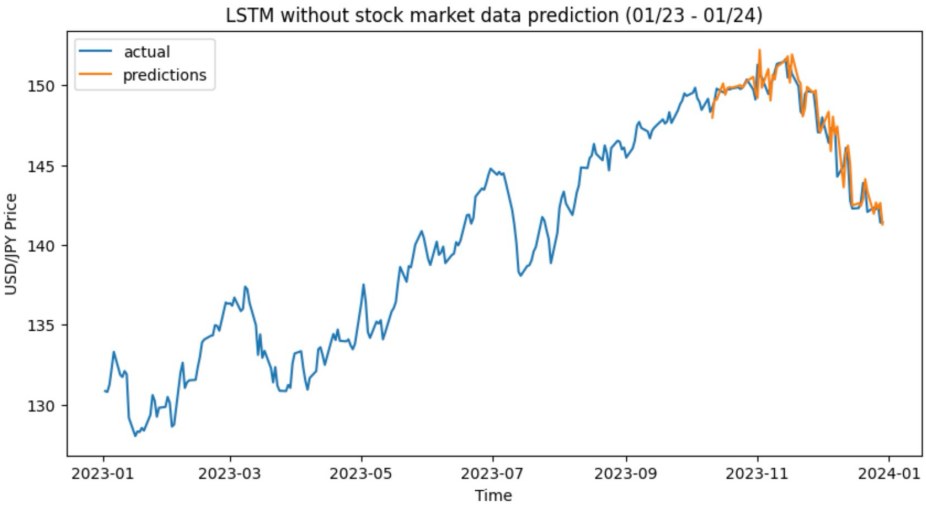

Figure 4.6 - LSTM trained on FX 1-year dataset prediction

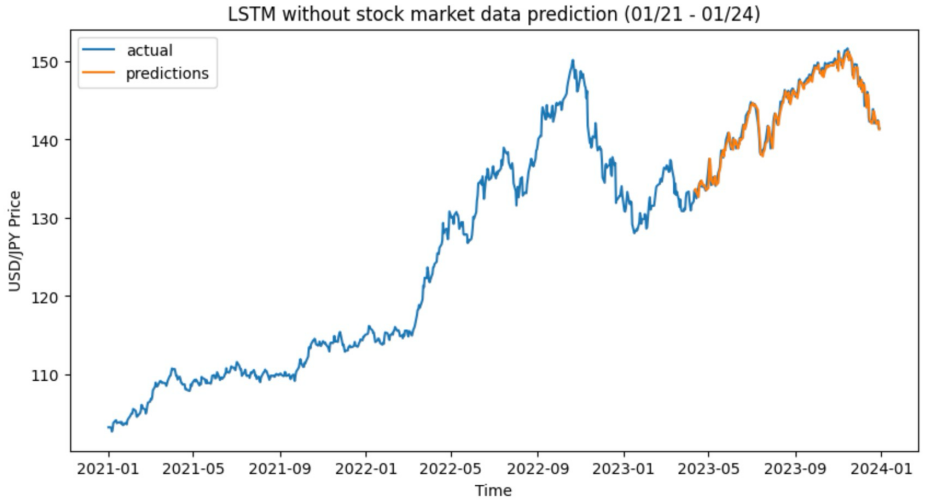

Figure 4.7 - LSTM trained on 3-year FX dataset prediction

4.3.2 With stock market data

LSTM	RMSE	MSE	MAE	MAPE	R2
1 Month	1.452	2.103	1.125	0.785	0.603
3 Months	1.196	1.43	0.926	0.634	0.582
6 Months	0.983	0.966	0.751	0.49	0.782
1 Year	0.621	0.386	0.462	0.475	0.823
3 Years	0.926	0.857	0.693	0.285	0.801
5 Years	0.575	0.331	0.409	0.212	0.852
10 Years	0.505	0.255	0.346	0.248	0.874

Figure 4.8 - LSTM trained on FX and Stock Market data Error Results

For a 1-month dataset, the RMSE is 1.452, indicating a moderate difference between predicted and actual exchange rates. As the training window expands to 3 months, the RMSE reduces to 1.196, suggesting some improvement in accuracy. This pattern continues with a further increase in dataset size, achieving its best performance with a 10-year training window, where the RMSE is 0.505. The MAPE metric shows a similar trend, decreasing as the dataset size increases. It reached its lowest value of 0.212 for the 5-year training window. It is also worth noting the steady increase of R^2 values with the increase of dataset size, reaching its peak at the 10-year dataset (0.874).

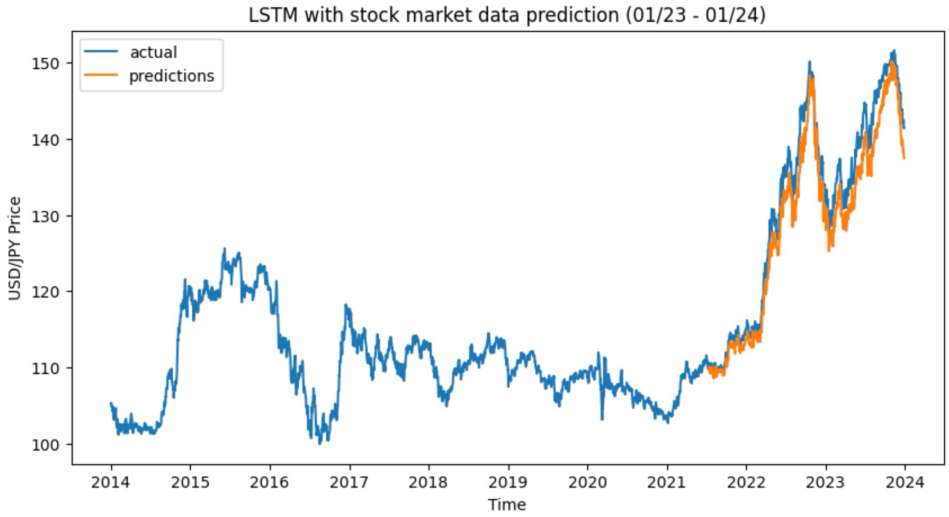

Figure 4.9 - LSTM with FX and Stock Market trained on 10 year dataset prediction

4.3.3 Impact of Additional Features on LSTM Performance

By comparing the two sets of results, the effect of incorporating additional features (in this case Nikkei 225 and S&P 500 data) on the performance of an LSTM can be examined. Some of the key observations across different error metrics were as follows:

- RMSE (Root Mean Squared Error): In all but the 1-month forecast, the LSTM with additional features exhibited lower RMSE values compared to the model trained only on USD/JPY. This indicates that for datasets of 3 months or more, the model that considers Nikkei 225 and S&P 500 data yields more accurate predictions.

- MSE (Mean Squared Error): The pattern for MSE largely mirrors that of RMSE. Lower MSE values for most datasets were observed with the LSTM trained on stock market data. This suggests a generally better fit for the model that incorporates additional features.

- MAE (Mean Absolute Error): The difference in MAE between the two models is less pronounced than for RMSE and MSE. There are instances where the USD/JPY-only model has a slightly lower MAE, particularly for shorter forecasts (1 and 3 months). However, the LSTM with stock market data still shows a competitive performance in terms of MAE.

- MAPE (Mean Absolute Percentage Error): Similar to MAE, the MAPE values are comparable across both models. The model trained on stock market data has a slight edge for longer forecasts (3 years, 5 years, and 10 years), indicating a marginally better ability to capture the proportional change in exchange rates.

The analysis of the two LSTM models suggests that incorporating relevant additional features (Nikkei 225 and S&P 500 data) can generally enhance the model's ability to predict USD/JPY exchange rates. The LSTM with a broader feature set exhibits lower error metrics for most forecast datasets, indicating a better fit and more accurate predictions. This highlights the potential benefits of considering intermarket relationships and a wider range of factors when modelling currency exchange rates using LSTMs.

4.4 XGBoost

For XGBoost, both classification and regression models were built and trained on USD/JPY historical data. Identical models were then built which were also trained on stock market data, the results of which are discussed below.

4.4.1 Regression without stock market data

Regressor XGBoost	RMSE	MSE	MAE	MAPE	R2
1 Month	2.137	4.568	2.084	1.471	-17.758
3 Months	2.339	5.471	2.232	1.569	-10.324
6 Months	1.712	2.932	1.373	2.115	0.514
1 Year	1.728	2.987	1.421	2.275	0.721
3 Years	1.553	2.410	1.278	2.839	0.832
5 Years	1.461	2.135	1.162	5.414	0.954
10 Years	15.432	238.144	13.149	10.571	-1.414

Figure 4.10 - XGBoost Regression FX Error Results

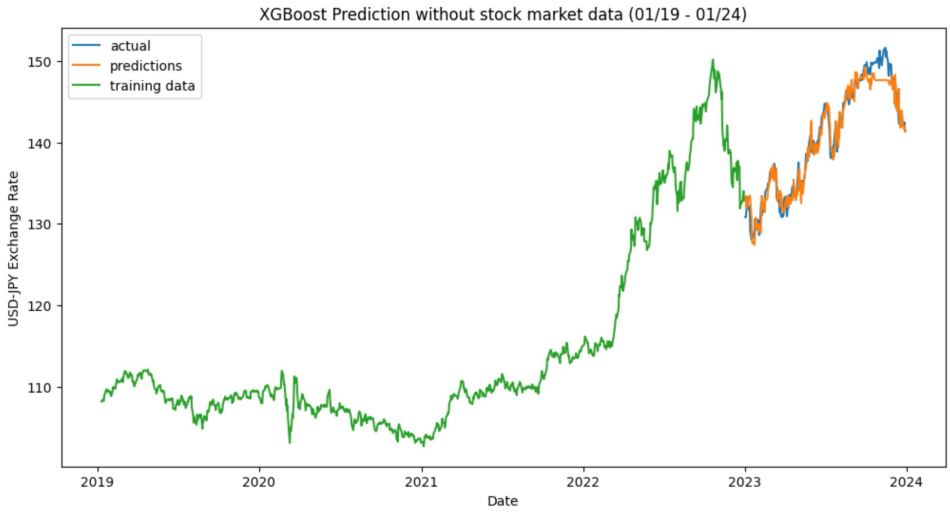

Figure 4.11 - XGBoost regression trained on 5-year FX dataset prediction

This model was the most accurate on the 5-year dataset, with both MSE and MAE values steadily decreasing with larger datasets. Interestingly, this trend reversed when training on the 10-year dataset as the model performed the worst with a considerably higher MSE (238.144).

4.4.2 Regression with stock market data

Regressor XGBoost	RMSE	MSE	MAE	MAPE	R2
1 Month	2.137	4.568	2.084	1.471	-17.758
3 Months	2.339	5.471	2.232	1.569	-10.324
6 Months	1.712	2.932	1.373	2.115	0.514
1 Year	1.755	3.080	1.452	2.259	0.712
3 Years	1.427	2.036	1.173	2.843	0.858
5 Years	1.475	2.176	1.195	5.395	0.953
10 Years	15.387	236.766	13.101	10.541	-1.400

Figure 4.12 - XGBoost regression trained on FX and Stock Market Error Results

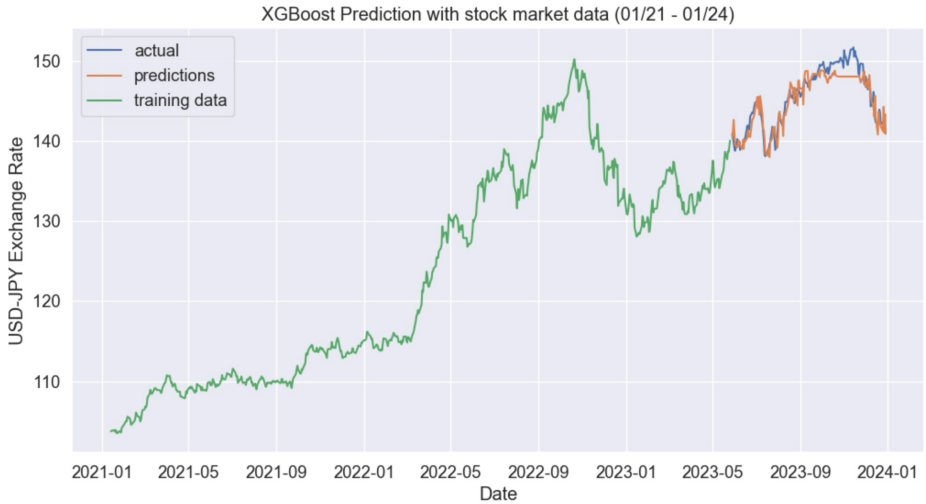

Figure 4.13 - XGBoost regression trained on 5-year FX and Stock Market dataset prediction

When introducing the stock market data, the model performs better on 3 years data instead of 5, with the RMSE, MSE and MAE all being lowest when the model is trained on the 3 year dataset. It is worth noting that there is very little difference between the values of this model and the previous one (some metrics even have the same value). However, there is a slight overall decrease in the numerical value of many of the metrics, further suggesting that there is a link between stock market performance and FX rates.

4.4.3 Classification without stock market data

Classifier XGBoost	Accuracy	Precision	Recall	F1 Score	Specificity
1 Month	NULL	NULL	NULL	NULL	NULL
3 Months	0.143	0.083	0.167	0.111	0.167
6 Months	0.500	0.456	0.450	0.451	0.450
1 Year	0.435	0.402	0.404	0.400	0.404
3 Years	0.373	0.361	0.361	0.360	0.361
5 Years	0.420	0.389	0.398	0.389	0.398
10 Years	0.373	0.367	0.371	0.367	0.371

Figure 4.14 - XGBoost trained on FX dataset classification accuracy results

The 1-month dataset was unable to provide any metrics for this classification model. This could have been due to the FX markets inherent volatility. With both short- and long-term trends, as well as seasonal fluctuations, a single month of data would likely be unable to encompass enough of these patterns for the model to learn from, which limits its ability to make accurate predictions beyond the very short term. Outside of this, it can be seen that the model performed best at 6 months with an accuracy of 0.5.

4.4.4 Classification with stock market data

Classifier XGBoost	Accuracy	Precision	Recall	F1 Score	Specificity
1 Month	NULL	NULL	NULL	NULL	NULL
3 Months	0.1429	0.3333	0.1111	0.1667	0.1111
6 Months	0.3500	0.3182	0.2685	0.2778	0.2685
1 Year	0.3842	0.3831	0.3873	0.3842	0.3873
3 Years	0.3200	0.3127	0.3020	0.3030	0.3020
5 Years	0.3898	0.3569	0.3582	0.3572	0.3582
10 Years	0.4447	0.4466	0.4430	0.4391	0.4430

Figure 4.15 - XGBoost trained on FX and Stock Market dataset classification accuracy results

Once again, the model was unable to successfully train on one month of data. However, with the introduction of stock market data, the model now performs best on the 10-year dataset instead of 6 months when trained on historical USD/JPY

data alone. This allows FX rates to be forecasted much further into the future, and further supports the hypothesis that larger datasets with more features lead to a better model.

However, it is also important to note that for both these classification models the highest accuracy is only 0.5. In a real-world scenario, this would not be very helpful as for any actionable FX trades to be made using these models, the accuracy would need to be significantly higher than this.

5 Conclusion

This dissertation delved into the application of machine learning and statistical models for the prediction of exchange rates within the USD/JPY currency pair. The central aims were to develop and evaluate various forecasting models, identify crucial factors influencing the exchange rate, and explore the potential of incorporating these factors for enhanced predictive accuracy. Three core models were examined – ARIMA, LSTM, and XGBoost – with variations implemented for each. The models were rigorously trained and evaluated using differing historical datasets, and their performance was assessed using pertinent error metrics.

5.1 Summary of Key Findings

The analysis revealed a set of insightful findings that underscore the significance of model selection, dataset characteristics, and feature engineering strategies in the context of ML-based FX forecasting.

5.1.1 Offline vs. Online ARIMA

The offline ARIMA model, achieved its optimal performance with the smallest dataset (1 month). Error metrics worsened significantly when using larger datasets (5-10 years). This counterintuitive trend suggests that while ARIMA models are suited for modelling trends and seasonality, they might struggle to adapt to the inherent randomness and non-stationarity of financial markets.

In contrast, the online ARIMA model's performance peaked with a 1-year dataset, demonstrating its superior adaptability to evolving market conditions. The online ARIMA's ability to continuously incorporate new data points proved essential in capturing the dynamic nature of the USD/JPY exchange rate.

5.1.2 LSTM

LSTM models exhibited a marked positive correlation between dataset size and performance. Models trained on larger datasets (3 years, 5 years, 10 years) consistently showed lower error metrics and higher R^2 values in regression tasks. This finding reinforces the LSTM's strength in learning complex patterns and long-term dependencies within vast financial datasets.

5.1.3 Impact of Stock Market Data

The inclusion of external market indicators (Nikkei 225 and S&P 500) demonstrably improved the predictive power of both LSTM and XGBoost models in regression tasks. Across most dataset sizes, the models trained with additional stock market data yielded lower error metrics. This underscores the interconnectedness of global financial markets and highlights the value of exploring related asset classes for enhanced FX forecasting.

5.1.4 Limitations of XGBoost Classification

While XGBoost has proven effective in various predictive tasks, its classification implementation in this project yielded limited practical applicability. The model's accuracy metrics remained consistently low, suggesting the need for either more extensive feature engineering or the exploration of alternative classification model architectures better suited to the highly volatile, non-linear patterns within the FX market.

5.2 Critical Reflection

The findings of this research prompt critical reflection on the strengths, limitations, and potential refinements for both model selection and experimental design. Much of the time spent on this project initially was research, both into factors that affect the FX market and into the models chosen for prediction. Whilst this project did provide a varied and interesting set of results, more time could have been spent on implementing and comparing more models and less time on theory. For example, a new model by Meta (formerly Facebook) was introduced last year for the prediction of time series data (Meta, 2023). A small experiment was run however not much time was spent on it, to allow focus on the more legacy models in this report. Whilst writing the report, however, it was realised that it would have been good to have a newer model to compare the existing data. This could have revealed how legacy models compare to latest advancements.

However, this did not hinder the project but was more a possible enhancement. The aims initially set were completed, and the process of completing this project has led to a deeper understanding of not just machine learning, but also financial markets and statistical analysis. Coding the project and analysing the outputs was enjoyable,

however the research completed allowed for a more in-depth theoretical understanding of each models implementation and underlying architecture. This is something that could not have been obtained by coding alone, as many libraries were used to implement the various models.

5.3 Ideas for Further Work

Building upon the outcomes and reflections, several promising avenues for future research emerge:

- Sentiment analysis – Another feature which could be added to the XGBoost and LSTM implementations in particular is analysis of news (both geopolitical and financial). This could potentially further improve prediction accuracy, and research indicates that market participants (such as traders) monitor such news to inform their own trading decisions (Mishev et al., 2020).

- Graphical User Interface – Should sentiment analysis further improve accuracy, resulting predictions could be used to make profit with the FX market. In order to make quick decisions, it would be better to have a GUI with which a user could easily change different parameters and currencies. Currently those without a background in computer science would find it difficult to manipulate the code written in this project, so such a development would help in making this work more accessible.

6 Reference list

Ajayi, R.A. and Mougouė, M. (1996). ON THE DYNAMIC RELATION BETWEEN STOCK PRICES AND EXCHANGE RATES. *Journal of Financial Research*, 19(2), pp.193–207. doi:https://doi.org/10.1111/j.1475-6803.1996.tb00593.x.

Ariyo, A.A., Adewumi, A.O. and Ayo, C.K. (2014). Stock Price Prediction Using the ARIMA Model. *2014 UKSim-AMSS 16th International Conference on Computer Modelling and Simulation*. [online] doi:https://doi.org/10.1109/uksim.2014.67.

Aroussi, R. (2023). *yfinance: Yahoo! Finance market data downloader*. [online] PyPI. Available at: https://pypi.org/project/yfinance/.

Babu, R. and Stock, J. (2015). Exchange Rate Forecasting using ARIMA, Neural Network and Fuzzy Neuron. [online] 4, p.3. Available at: https://web.archive.org/web/20190227153004id_/http://pdfs.semanticscholar.org/82c1/3bca3812e909261a968d06c85a862f77d803.pdf.

Beine, M., Bos, C.S. and Laurent, S. (2006). The Impact of Central Bank FX Interventions on Currency Components. *Journal of Financial Econometrics*, 5(1), pp.154–183. doi:https://doi.org/10.1093/jjfinec/nbl008.

Box, G.E.P., Jenkins, G.M., Reinsel , G.C. and Ljung, G.M. (2016). Time Series Analysis: Forecasting and Control, 5th Edition, by George E. P. Box, Gwilym M. Jenkins, Gregory C. Reinsel and Greta M. Ljung, 2015. Published by John Wiley and Sons Inc., Hoboken, New Jersey, pp. 712. ISBN: 978-1-118-67502-1. *Journal of Time Series Analysis*, 37(5), pp.709–711. doi:https://doi.org/10.1111/jtsa.12194.

Broby, D., Faessler, R., Josavac, M. and Dehut, C. (2016). The Impact of the (2011) Devaluation of the Swiss Franc on Eurozone Equity Benchmark Diversification. *International Journal of Economics and Financial Issues*, [online] 6(3), pp.1270–1286. Available at: https://dergipark.org.tr/en/pub/ijefi/issue/32012/353828.

Chen, T. and Guestrin, C. (2016). XGBoost: a Scalable Tree Boosting System. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '16*, [online] pp.785–794. doi:https://doi.org/10.1145/2939672.2939785.

Delivorias, A. (2015). *At a glance*. [online] Available at: https://www.europarl.europa.eu/EPRS/EPRS-AaG-545730-Swiss-decision-exchange-rate-ceiling-FINAL.pdf.

Deng, S., Zhu, Y., Huang, X., Duan, S. and Fu, Z. (2022). High-Frequency Direction Forecasting of the Futures Market Using a Machine-Learning-Based Method. *Future Internet*, 14(6), pp.180–180. doi:https://doi.org/10.3390/fi14060180.

Dian Jia and Ruixiang Xue, (2022). Research on Earnings Management of Growth Enterprise Market in China Stock Market: Comparative Analysis Based on the BPNN, GBDT, and MLR Models, *Computational Intelligence and Neuroscience*, vol. 2022, Article ID 6064536, 9 pages, 2022. https://doi.org/10.1155/2022/6064536

Farzad, A., Mashayekhi, H. and Hassanpour, H. (2017). A comparative performance analysis of different activation functions in LSTM networks for classification. *Neural Computing and Applications*, 31(7), pp.2507–2521. doi:https://doi.org/10.1007/s00521-017-3210-6.

Fernandes, M., Lourenço, A., Marreiros, G. and Corchado, J. (2024). Imputation of Industrial Streaming Data Using Arima Models and Kalman Filter. [online] doi:https://doi.org/10.2139/ssrn.4734890.

Grossberg, S. (2013). Recurrent neural networks. *Scholarpedia*, [online] 8(2), p.1888. doi:https://doi.org/10.4249/scholarpedia.1888.

Hayes, A. (2021). *Understanding Uncovered Interest Rate Parity – UIP*. [online] Investopedia. Available at: https://www.investopedia.com/terms/u/uncoveredinterestrateparity.asp#:~:text=Uncovered%20interest%20rate%20parity%20(UIP)%20theory%20states%20that%20the%20difference.

Hayes, A. (2023). *Safe Haven Currency: Definition, How It Works, How to Trade*. [online] Investopedia. Available at: https://www.investopedia.com/safe-haven-currency-7503736.

Hochreiter, S. (1998). The Vanishing Gradient Problem During Learning Recurrent Neural Nets and Problem Solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 06(02), pp.107–116. doi:https://doi.org/10.1142/s0218488598000094.

Hochreiter, S. and Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, 9(8), pp.1735–1780.

Hyndman, R.J. and Athanasopoulos, G. (2016). *Forecasting: Principles and Practice*. [online] Otexts.com. Available at: https://otexts.com/fpp2/.

Investopedia. (2023). *Top 6 Most Tradable Currency Pairs*. [online] Available at: https://www.investopedia.com/top-6-most-tradable-currency-pairs-4773389#citation-3.

Islam, S.F.N., Sholahuddin, A. and Abdullah, A.S. (2021). Extreme gradient boosting (XGBoost) method in making forecasting application and analysis of USD exchange rates against rupiah. *Journal of Physics: Conference Series*, 1722, p.012016. doi:https://doi.org/10.1088/1742-6596/1722/1/012016.

Jones, M. and John, A. (2022). Global FX trading hits record $7.5 trln a day - BIS survey. *Reuters*. [online] 27 Oct. Available at: https://www.reuters.com/markets/us/global-fx-trading-hits-record-75-trln-day-bis-survey-2022-10-27/.

Kallianiotis, I.N. (2022). Trade Balance and Exchange Rate: The J-Curve. *Journal of Applied Finance & Banking*, pp.41–64. doi:https://doi.org/10.47260/jafb/1223.

Kallianiotis, J.N. (2013). The Foreign Exchange Market. *Exchange Rates and International Financial Economics*, pp.51–82. doi:https://doi.org/10.1057/9781137318886_2.

Li, K. (2023). A Sales Prediction Method Based on XGBoost Algorithm Model. *BCP Business & Management*, 36, pp.367–371. doi:https://doi.org/10.54691/bcpbm.v36i.3487.

Li, P. and Zhang, J.-S. (2018). A New Hybrid Method for China's Energy Supply Security Forecasting Based on ARIMA and XGBoost. *Energies*, 11(7), p.1687. doi:https://doi.org/10.3390/en11071687.

Lusk, E., Halperin, M. and Heilig, F. (2011). A Note on Power Differentials in Data Preparation between Trimming and Winsorizing. *Business Management Dynamics*, [online] 1(2), pp.23–31. Available at: https://edwardlusk.com/contributions/164.pdf [Accessed 8 Apr. 2024].

Mara, S., Id, R., Leite, C. and Id, C. (2022). Learning and Nonlinear Models. *Journal of the Brazilian Society on Computational Intelligence (SBIC)*, [online] 20(1), pp.31–

46. Available at: https://sbic.org.br/lnlm/wp-content/uploads/2022/12/vol20-no1-art3.pdf [Accessed 8 Apr. 2024].

Meta (2023). *Prophet*. [online] Prophet. Available at: https://facebook.github.io/prophet/.

Mishev, K., Gjorgjevikj, A., Vodenska, I., Chitkushev, L.T. and Trajanov, D. (2020). Evaluation of Sentiment Analysis in Finance: From Lexicons to Transformers. *IEEE Access*, [online] 8, pp.131662–131682. doi:https://doi.org/10.1109/ACCESS.2020.3009626.

Mong, T. and Ngan, U. (2016). Forecasting Foreign Exchange Rate by using ARIMA Model: A Case of VND/USD Exchange Rate. *Research Journal of Finance and Accounting www.iiste.org ISSN*, [online] 7(12), pp.2222–2847. Available at: https://core.ac.uk/download/pdf/234631456.pdf.

Natekin, A. and Knoll, A. (2013). Gradient boosting machines, a tutorial. *Frontiers in Neurorobotics*, 7(21). doi:https://doi.org/10.3389/fnbot.2013.00021.

Ngoc Thanh Tran, Thi, T., Tuan Anh Nguyen and Minh Binh Lam (2023). A new grid search algorithm based on XGBoost model for load forecasting. *Bulletin of Electrical Engineering and Informatics*, 12(4), pp.1857–1866. doi:https://doi.org/10.11591/beei.v12i4.5016.

Nidhi (2023). *Decision Trees: A Powerful Tool in Machine Learning*. [online] Medium. Available at: https://medium.com/@nidhigh/decision-trees-a-powerful-tool-in-machine-learning-dd0724dad4b6.

Olah, C. (2015). *Understanding LSTM Networks -- colah's blog*. [online] Github.io. Available at: http://colah.github.io/posts/2015-08-Understanding-LSTMs/.

Pai, P.-F. and Lin, C.-S. (2005). A hybrid ARIMA and support vector machines model in stock price forecasting. *Omega*, 33(6), pp.497–505.

Pankratz, A. (1983). *Forecasting with Univariate Box-Jenkins Models. John Wiley & Sons, Inc. eBooks*. Wiley. doi:https://doi.org/10.1002/9780470316566.

Ray, S., Lama, A., Mishra, P., Biswas, T., Das, S.S. and Gurung, B. (2022). An Arima-Lstm Model for Predicting Volatile Agricultural Price Series with Random Forest Technique. *SSRN Electronic Journal*. doi:https://doi.org/10.2139/ssrn.4215052.

Sanchez, M. (2008). The link between interest rates and exchange rates: do contractionary depreciations make a difference? *International Economic Journal*, 22(1), pp.43–61. doi:https://doi.org/10.1080/10168730801898981.

Sarno, L. and Taylor, M.P. (2001). Official Intervention in the Foreign Exchange Market: Is It Effective and, If So, How Does It Work? *Journal of Economic Literature*, 39(3), pp.839–868. doi:https://doi.org/10.1257/jel.39.3.839.

Sato, R.C. (2013). Gerenciamento de doenças utilizando séries temporais com o modelo ARIMA. *Einstein (São Paulo)*, 11(1), pp.128–131. doi:https://doi.org/10.1590/s1679-45082013000100024.

Shakti, S.P., Hassan, M.K., Zhenning, Y., Caytiles, R.D. and N.Ch.S.N, I. (2017). Annual Automobile Sales Prediction Using ARIMA Model. *International Journal of Hybrid Information Technology*, 10(6), pp.13–22. doi:https://doi.org/10.14257/ijhit.2017.10.6.02.

Siami-Namini, S., Tavakoli, N. and Siami Namin, A. (2018). A Comparison of ARIMA and LSTM in Forecasting Time Series. *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*. doi:https://doi.org/10.1109/icmla.2018.00227.

Sirisha, U.M., Belavagi, M.C. and Attigeri, G. (2022). Profit Prediction Using ARIMA, SARIMA and LSTM Models in Time Series Forecasting: A Comparison. *IEEE Access*, 10, pp.124715–124727. doi:https://doi.org/10.1109/access.2022.3224938.

Solo, V. (1984). The Order of Differencing in ARIMA Models. *Journal of the American Statistical Association*, 79(388), pp.916–921. doi:https://doi.org/10.1080/01621459.1984.10477111.

Srivastava, N., Hinton, G., Krizhevsky, A. and Salakhutdinov, R. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, [online] 15, pp.1929–1958. Available at: https://jmlr.org/papers/volume15/srivastava14a/srivastava14a.pdf.

Vladimir Rybalkin, Chirag Sudarshan, Weis, C., Lappas, J., Wehn, N. and Cheng, L. (2020). Efficient Hardware Architectures for 1D- and MD-LSTM Networks. *Journal of Signal Processing Systems*, 92(11), pp.1219–1245. doi:https://doi.org/10.1007/s11265-020-01554-x.

Wang, H., Kin On Kwok and Riley, S. (2023). Forecasting influenza incidence as an ordinal variable using machine learning. *medRxiv (Cold Spring Harbor Laboratory)*. doi:https://doi.org/10.1101/2023.02.09.23285705.

Wang, J., Cheng, Q. and Dong, Y. (2022). An XGBoost-based multivariate deep learning framework for stock index futures price forecasting. *Kybernetes*. doi:https://doi.org/10.1108/k-12-2021-1289.

Wu, Y. (1996). Are Real Exchange Rates Nonstationary? Evidence from a Panel-Data Test. *Journal of Money, Credit and Banking*, 28(1), p.54. doi:https://doi.org/10.2307/2077966.

Yahoo Finance (2023). *Yahoo Finance – stock market live, quotes, business & finance news*. [online] uk.finance.yahoo.com. Available at: https://uk.finance.yahoo.com.

Zhang, G.Peter. (2003). Time series forecasting using a hybrid ARIMA and neural network model. *Neurocomputing*, [online] 50, pp.159–175. doi:https://doi.org/10.1016/s0925-2312(01)00702-0.