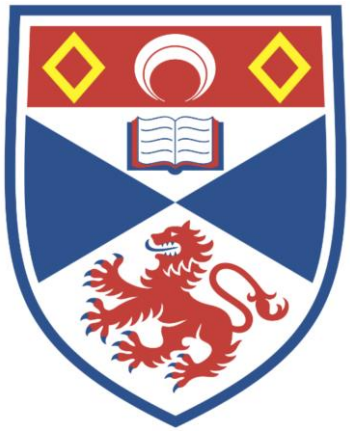
Advanced forecasting of stock prices with Long-Short Term Memory Neural Networks based on Attention mechanisms



Blair Robert Ian Welsh
School of Computer Science
St Andrews University

This thesis is submitted in partial fulfilment for the degree of
MSc Artificial Intelligence

16th August 2022

Abstract

The stock market is renowned for its intricacy and volatility, and investors have long relied on technology to assist them in making trading decisions. This paper investigates the application of LSTM neural networks with attention mechanisms and wavelet transformations for stock price forecasting, based on the impressive findings of a prior study. The objective of the first portion of this research duplicates the results of the authors by constructing the same WLSTM+Attention model and analysing its performance using the same stock market data. In addition, important information on the profitability of a trading strategy based on this model is missing from the author's work. Therefore, the second phase of this project develops an algorithmic investment strategy based on the model's predictions in order to generate profitability metrics to support the model's predictions. The results show that replication of the authors work was not possible, due to numerous proven flaws discovered in their work that effect the validity of their results, however, the evaluation of three separate datasets proved successful with accurate model predictions resulting in a coefficient of determination of 0.997 and trading strategy performance reaching an average profitability of 16.608%. Therefore, this study supports the application of LSTMs, attention mechanisms and wavelet transformations in financial forecasting and investment strategies and is highly competitive with existing research.

Declaration

I declare that the material submitted for assessment is my own work except where credit is explicitly given to others by citation or acknowledgement. This work was performed during the current academic year except where otherwise stated.

The main text of this project report is 13,931 words long, including project specification and plan.

In submitting this project report to the University of St Andrews, I give permission for it to be made available for use in accordance with the regulations of the University Library. I also give permission for the title and abstract to be published and for copies of the report to be made and supplied at cost to any bona fide library or research worker, and to be made available on the World Wide Web. I retain the copyright in this work.

16th August 2022

Blair Robert Ian Welsh

Contents

List of Figures

List of Tables

1. Introduction

1.1 About & Motivation

Can anyone predict the stock market? Stocks are the centrepiece of any investment portfolio and are arguably the most popular financial tool ever created for accumulating wealth. A stock is a type of security asset that grants the holder proportionate ownership in the issuing corporation and gives them the opportunity to vote in shareholder meetings, receive dividends when they are distributed, and sell their shares to other investors for profit on stock exchanges.

Developments in trading technology have opened up the markets and made it possible for practically everyone to buy stocks. In the last decade, there has been an explosive increase in the common person's interest in the stock market, further fuelled by the COIVD-19 pandemic. Downloads of E-trading applications such as Robinhood, Coinbase, and Webull skyrocketed from 98 million before the pandemic to 175 million during the first quarter of 2021, a 34 percent increase year-over-year [1]. Not only has the convenience of financial trading risen, but so too has information accessibility. The era of the stock market being a secret club of "those in the know" is slowly coming to an end as the internet now provides free access to resources dedicated to the reading of analytics and trends, discussion forums full of support groups and communities, and financial gurus offering experienced trading advice through social media. All of which has culminated in hordes of investors with starry-eyed dreams of making their fortunes in the buying and selling of shares.

But every investment comes with some degree of risk. Analysis of trading-platform data conducted by the E-trading application Etoro revealed that 80 percent of day traders are unprofitable over the course of a year, with around 75 percent of traders leaving the market within just two years [1]. Even in 1999, the National American Securities Administration Association (NASAA) estimated that 70 percent of traders will lose virtually all their money while fewer than 12 percent make a profit on their short-term transactions [2], so why is this the case? In short, the forecasting of financial markets is an extremely complicated endeavour. Financial data is noisy, non-stationary, irregular, and fluctuates wildly, and these arbitrary variations can drastically affect any attempt to predict the current stock market price for a particular day. Market prices are determined by the laws of supply and demand, and confidence in future investments has a substantial impact in whether markets rise or fall. Investors will be more inclined to acquire shares if they believe the stock will perform well in the future, boosting demand and therefore price, and will look to sell their shares if they believe that the stock will depreciate in value, thereby increasing supply and decreasing the price. The primary problem here lies in that there are an enormous number of factors that can affect investor confidence, with inflation, government policy, technological advancements, regulation, and consumer confidence being but a few of the variables that contribute to this unpredictability. Thus, while many enter the financial market with high expectations of profitable investments, only a select few have the will or good fortune to realise those expectations.

To ensure success when investing on the stock market, one must then be able to predict which stocks will appreciate in value and which will decline. A paper by Hendrik Bessembinder, an Arizona State University finance professor, analysed the performance of 26,000 publicly traded stocks since 1926 [3]. His research revealed that only 1000 of the stocks analysed account for all trading gains since 1926, and of those 1000 stocks, only 86, or less than 0.3%, were responsible for half of those profits. With the potential financial reward for successfully navigating the stock market being so substantial, a growing number of investments have been made in technology that can accurately anticipate the future value of stocks and determine which are part of this small number that will offer profitable returns. Advancements in Artificial

Intelligence and Deep Learning are among the most inventive and effective approaches for achieving this goal.

Deep learning is a subfield of Machine Learning that mimics the cognitive abilities and structure of the human brain in order to identify correlations and patterns in processed data. By employing numerous hidden layers as opposed to traditional Neural Networks, which only use one, Deep Neural Networks can deal with high dimensional datasets and automatically learn features to achieve strong accuracy in their forecasts. Deep Learning, which was first introduced in the 1960s, was initially met with scepticism due to its lack of large datasets and high processing power requirements. However, technological advancements have eliminated these issues and Deep Learning is now one of the most popular research fields in Artificial Intelligence and Machine Learning. With successful applications in challenging subfields such as Natural Language Processing, Computer Vision, and Anomaly Detection tasks, Deep Learning has been at the forefront of technology designed to predict stock market values and fluctuations during the last few decades (see Section 2.2.1).

One paper which applies Deep Learning techniques for Stock market prediction was published by Jiayu Qiu, Bin Wang, and Changjun Zhou in 2020 and demonstrates the potential of Long-Short Term Memory (LSTM) Neural Networks to accurately predict the opening price of stock market data by employing an attention-based mechanism and a wavelet transformation, yielding very impressive predictions shown in Figure 1 [4]. However, the authors do not provide any code that supports their publication nor do they report crucial information regarding network structure and composition. Therefore, the primary objective of this paper is to replicate their results by implementing an LSTM with attention mechanism and evaluating its performance on the same (alongside alternative) stock market data. In addition, the authors never incorporate their model predictions into any sort of trading strategy, hence important information on the profitability of a trading strategy based on this model is also missing. Consequently, when the results of Qui et al. have been successfully replicated, the second phase of this project will involve the development of an algorithmic investment strategy based on the model's predictions to generate profitability metrics and gain a deeper understanding of the potential application of Deep Learning and specifically LSTMs for financial forecasting and investment strategies.

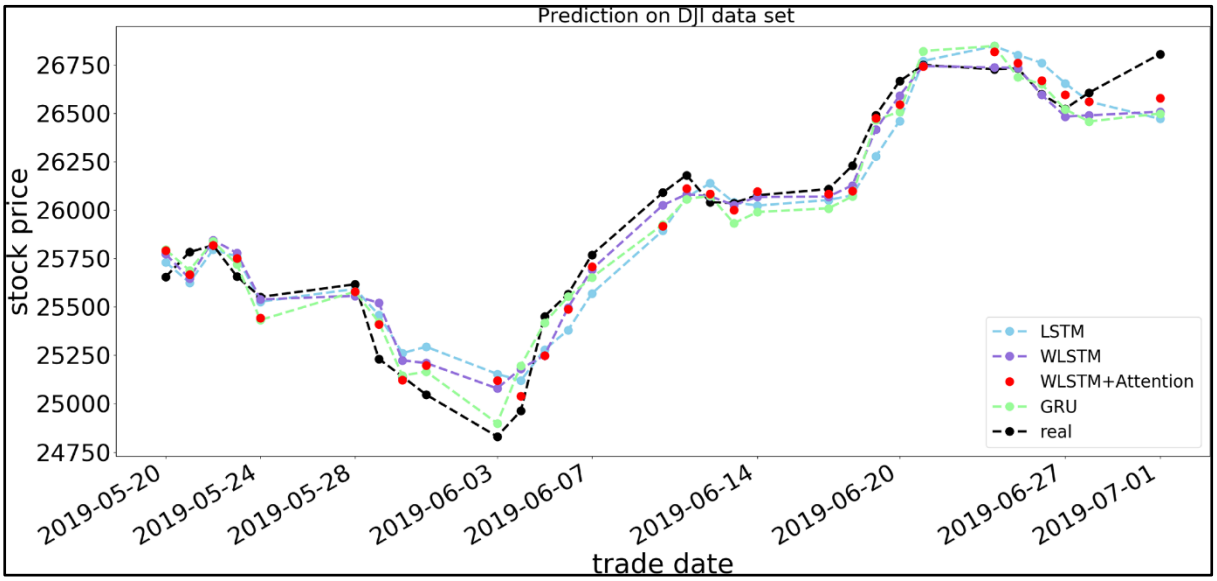


Figure 1 - Qui et al.'s forecast results of four models for DJIA opening price. Taken from [4].

1.2 Aims and Objectives

The aims of this project can be split into primary and secondary objectives. The primary objective involves replication of the results found by Qui et al. using the same financial datasets as the authors and potentially alternative data. The secondary aim involves implementation of an algorithmic investment strategy based on the model predictions to generate profitability metrics. All objectives of this project are described in Table 1.1 below:

#	DESCRIPTION
	PRIMARY OBJECTIVES
P1	Implement a LSTM with wavelet transformation and attention mechanism to predict the future price of a stock based upon the historical market data.
P2	Evaluate the performance of the network using the same stock market data used as Qui et al.
P3	Evaluate the performance of the network using different stock market data.
	SECONDARY OBJECTIVES
S1	Evaluate the application of a trading strategy based on the model's predictions in order to provide a real-world illustration of the model's performance.

Table 1.1 - List of Project Objectives

1.3 Structure

The remainder of this text is organised as follows. The background material for the project is presented in Chapter 2, with a focus on Recurrent Neural Networks, Long-Short Term Memory Networks, Attention Mechanisms, Wavelet Transformations, and Trading Strategies. The context survey that follows describes the preceding research that has been performed in each of these topics. The system's prerequisites, design, and implementation are then covered in depth in Chapter 3. Prerequisites explains the development environment and project dependencies, while Design and Implementation describes the overall structure of the system and how it was built with examples of code and novel algorithms used. The chapter is finalised with a brief discussion of the project evaluation metrics. The datasets utilised for this project are then explored in Chapter 4, before the model's results and trading strategy performance are assessed and discussed. A Critical Appraisal that compares the work to the original aims and similar work in the public domain concludes the chapter, then the final chapter, Chapter 5, then summarises the project and discusses future directions the work could be taken in.

2. Context Survey

This section will examine the project's context, background literature, and any recent work with similar goals. Section 2.1 describes the background knowledge about important concepts and theory for the project, including RNNs, LSTMs, Attention, Wavelet transformations, and Trading Strategies. This section is then finalised by discussing the previous research with similar aims relating to the project in Section 2.2.

2.1 Background

2.1.1 Recurrent Neural Networks (RNNs)

Traditional Neural Networks (also known as Feed-Forward Neural Networks) view their inputs as independent of each other, and as a result they can struggle in situations where the output of the network relies on the values of the previous inputs, such as time series data or data that involves sequences [5].

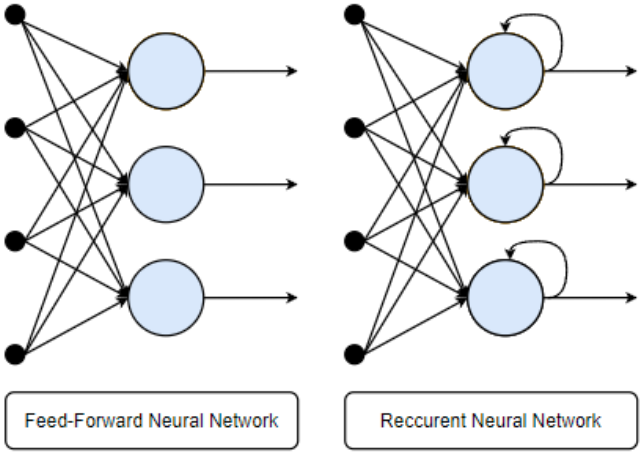


Figure 2- Comparison between Feed Forward NN and Recurrent NN [6]

In order to address this issue, Recurrent Neural Networks (RNNs) were created with the concept of 'memory', based on work performed by Rumelhart, et al. in 1986 [7]. This 'memory' is referred to as the hidden state and is a representation of the previous inputs to the network. The hidden state is used in conjunction with the current input to calculate an output for the RNN based on all previous inputs to the network. The hidden state is then updated to include the new input before being fed back into the original layer via a feedback loop to be used for the next input in the sequence, as illustrated in Figure 2 [6] and Figure 3 [8]. Due to the RNNs ability to model time series or sequential data, they have been successfully applied in a variety of domains including Machine Translation [9], Speech Recognition [10] and Anomaly Detection [11].

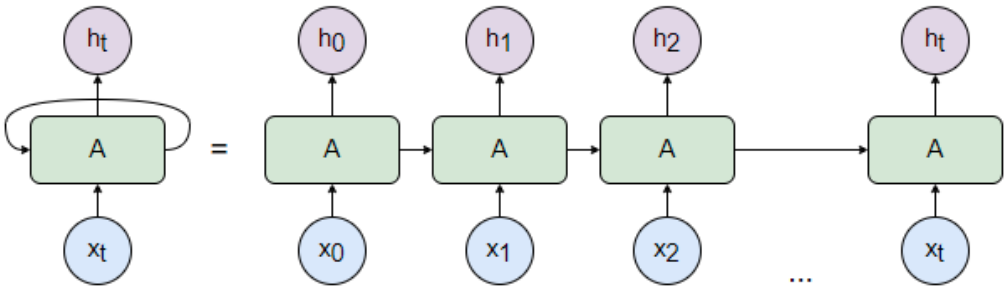


Figure 3- An example of a RNN with a feedback loop and its unrolled version [8]

However, during their training process RNNs can suffer from two issues known as vanishing gradients and exploding gradients [12], which are also prevalent in other neural network architectures. During the specific backpropagation process for RNNs called 'backpropagation through time' (BPTT) when the gradient is too small it continues to shrink increasing smaller, constantly decreasing the weight parameters of the network until they eventually become negligible at which point the network ceases learning. Likewise, the gradients can also 'explode', growing exponentially larger until they can no longer be represented by the network's allocated memory. These issues lead into the problem of short-term memory, where as an RNN processes more inputs it has trouble retaining information from previous inputs. This means that if the previous state influencing the current prediction is not in the recent past the network may not be able to accurately predict the current state, resulting in the network possessing a 'short-term memory' [13].

2.1.2 Long-Short Term Memory Networks (LSTMs)

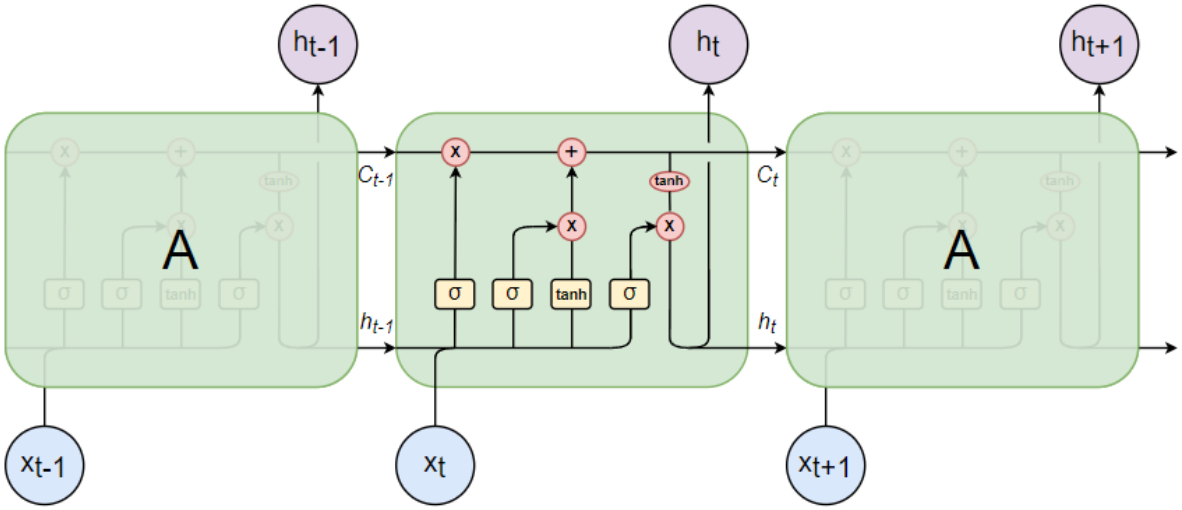


Figure 4 – Structure of the LSTM cell [14]

To mitigate the RNN issue of short-term memory, Sepp Hochreiter and Jurgen Schmidhuber proposed the Long-short Term Memory (LSTM) model in 1997 [15]. This model essentially functions just like RNNs but is capable of learning long-term dependencies by replacing the hidden layer of neurons with a unique memory cell shown in Figure 4 [14]. These cells have three gates that control which information to add or remove from the hidden state: an input gate, an output gate, and a forget gate. By utilising these gates, the LSTM is able to address the issue of long-term dependencies by updating the hidden state with every input, adding new relevant information and forgetting information deemed irrelevant. Because of this, the LSTM model has yielded excellent results on a wide range of applications including speech recognition [16] and text classification [17].

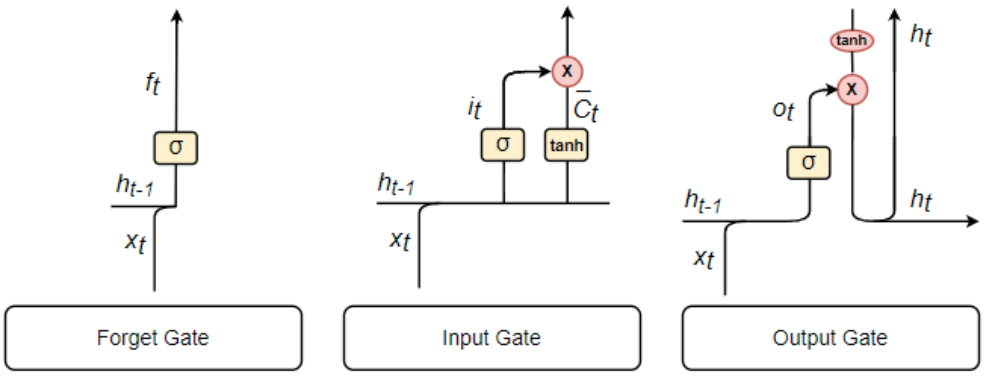


Figure 5- The three gates inside an LSTM cell: Forget, Input, and Output.

The first step in an LSTM is to decide which information to discard from the previous cell state that is not needed for the present unit, this is made by the forget gate shown in Figure 5. The forget gate uses the output of the previous cell h_{t-1} and current input x_t combined into a long vector $[h_{t-1}, x_t]$ as input for a particular time step. These inputs are multiplied by a weight matrix with a bias added, before being passed through a sigmoid (activation) function to output a number between 0 and 1 for each value in the cell state C_{t-1}. A value of 1 forces the network to completely keep the value whilst 0 forces it to completely discard the value. The equation for the forget gate is shown as in Eq.(1) below.

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

The next step is the input gate which determines how much of the current input x_t to add to the cell state C_t and prevents insignificant information from entering the cell's memory, shown in Figure 5. There are two main operations in the input gate. The first uses a tanh layer to create a vector $\tilde{C}_t$ of the new information to add to the state, as in Eq.(2). The second uses a sigmoid layer to decide which values of the cell state to update i_t, as in Eq.(3). The outputs of these two operations are then combined to create an update to the state. To find the new state C_t, the network multiplies the old state by the output of the forget gate f_t, forgetting the information deemed unnecessary earlier. It is then updated by adding the new information, scaled by how much it decided to update each state value $i_t * C_{t-1}$, as shown in Eq.(4).

$$\tilde{C}_t = \tanh(W_c \cdot [h_{t-1}, x_t] + b_c) \tag{2}$$

$$\mathrm{i_t} = (W_t \cdot [h_{t-1}, x_t] + b_i) \tag{3}$$

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{4}$$

The final step is to calculate the output of the cell h_t using the output gate, again shown in Figure 5. This output will be based on the updated cell state C_t, filtered through a tanh layer to push the values to be between -1 and 1. This filtered cell state will be multiplied by our previous output and current input ran through a sigmoid layer to determine which parts of the cell state to output, as in Eq.(5) and Eq.(6).

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{5}$$

$$h_t = o_t * \tanh(C_t) \tag{6}$$

2.1.3 Attention Mechanism

While the addition of memory cells has helped to alleviate the long-term dependency issues that plague RNNs, a problem can still occur within LSTMs due to the fixed length of the hidden state vectors. Long and/or complex input sequences are compressed and forced to be represented with the same dimensionality as shorter or simpler sequences, and the network may lose information and struggle to retain knowledge from prior inputs for these more challenging sequences as a result. As Bengio et al. explained in one of the first papers describing attention for natural language processing tasks: "*A potential issue with this encoder–decoder approach is that a neural network needs to be able to compress all the necessary information of a source sentence into a fixed-length vector. This may make it difficult for the neural network to cope with long sentences, especially those that are longer than the sentences in the training corpus*" [18].

To address this issue, Bengio et al. proposed the concept of Attention, a form of biomimicry that aims to replicate the cognitive ability of the human brain to selectively concentrate on certain information in its surroundings within Neural Networks. To implement this, networks are given the ability to devote more focus to, and so enhance a portion of, their input data, while simultaneously diminishing and disregarding other parts that are deemed to be less important. The implementation of Attention mechanisms has yielded excellent results in serialised data problems such as machine translation [19], video captioning [20] and flood prediction [21].

In the original research paper on which this project is based, Qiu et al. [4] implements a soft attention-based mechanism, assigning a weight to every value in the input data representing how much focus the network should assign to each value. Each weight is non-negative, and the sum of all the weights should be equal to one. By multiplying each value vector by its respective weight the network is able to calculate the output of the attention mechanism. This is in contrast to a hard attention mechanism, which only focuses on a single element in the input data chosen by some form of random or maximal sampling and necessitates extensive training to achieve effective results.

The inputs to the implemented attention mechanism are the output of the LSTM hidden layer; This is represented by $x = [x_1, x_2, \dots, x_N]$. Firstly, the attention distribution of the input data needs to be calculated. To do this, the degree of matching of each element in the input information is found as shown in equation (7). The input x is multiplied by w_a (the weight matrix of the attention mechanism) indicating information that should be emphasised, and value b (the deviation of the attention mechanism) is added. After that, the vector is then passed through a tanh layer to limit the values between -1 and 1. The attention distribution is then calculated by inputting the results into a softmax layer to obtain the final attention weights $[\alpha_1, \alpha_2, \dots, \alpha_N]$, as shown in equation (8). To obtain the final output, the attention weight vector is weighted and averaged with the input data.

$$e_t = \tanh(w_a[x_1, x_2, \dots, x_T] + b) \quad (7)$$

$$a_t = \frac{\exp(e_t)}{\sum_{k=1}^{T} \exp(e_t)} \quad (8)$$

2.1.4 Wavelet Transformation

Due to its complex and volatile nature stock market prices are a good example of a noisy data signal, that is, a signal that exhibits unwanted and random fluctuations that interfere with the main useful signal [22]. These arbitrary fluctuations can drastically affect any attempt to predict the current stock market price for a given day, and as a result research has been applied to find ways to denoise the signal and extract the useful data [23]. Such work has posed challenging, as traditional denoising algorithms struggle with financial data due to the main useful signal being nonstationary with a random overlap of noise. An area that has proved effective is wavelet transformations, which are considered more suitable for financial data as it can perform both time and frequency domain analysis. Research by Shi et al. [24] and Sun & Mein [25] has shown that the effectiveness of wavelet transformations to successfully denoise financial datasets and separate the useful signal from the noise, and as a result it is common to perform a wavelet transformation on any economic data during pre-processing to smooth out the signal and enhance the accuracy of any predictions made. Common wavelet basis functions for denoising financial data include coif N, sym N, db N, and Haar. An example of the effect of a wavelet transformation for denoising financial is shown below in Figure 6,

note the smoother denoised orange line with less variation than the jagged and noisy blue line.

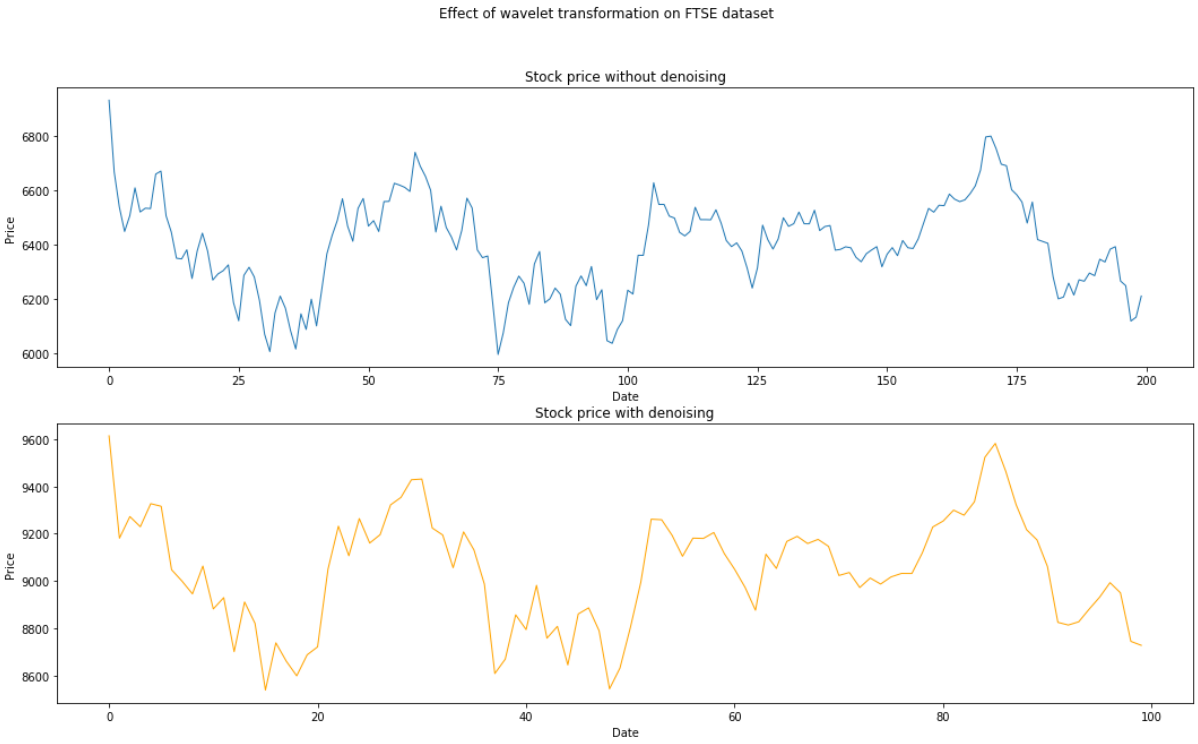


Figure 6 – Visualisation of the denoising effect of a wavelet transformation applied on financial data

2.1.5 Trading Strategies

In finance, a trading strategy is a fixed plan designed to achieve a profit on stock markets by developing a precise and well-defined set of trading rules based on fundamental and technical indicators. A trading strategy will direct a trader into long or short positions with the goal of producing a profit based on certain asset measurements. Common trading methods include Day Trading, which involves buying and selling inside the same day [26], Scalping, which involves making hundreds of transactions each day to gain a small quick profit from each [27], and Social Trading, which utilises the trading behaviour of others to determine a trading strategy [28].

2.2 Previous Research

2.2.1 Machine Learning for Stock Market Prediction

Since Machine Learning algorithms can take many different forms they have been proposed in various ways depending on the financial market being analysed or viewpoint of the authors. In 2022, Illa et al. compared the use of a Random Forest (RF) model and Support Vector Machine (SVM) to predict stock prices to find profitable trading strategies using data from the Dow Jones Industrial Average from 2008 to 2016 [29]. The algorithms “exhibited notable execution in predicting the stock markets”, resulting in accuracies of 81.6 and 87.4 percent respectively, with the RF model outperforming its counterpart on the provided dataset when taking into account other evaluation metrics. In the same year, Rath et al. compared the uses of SVMs, RF, Decision Trees (DT) and Neural Networks (NN) to predict the Bombay Stock Exchange using a dataset of 8 features across 5 years, implemented in Python [30]. Their results agreed with those of Illa et. al, as RFs performed the best with an accuracy of 0.91, compared to 0.85 and 0.91 for the NN and SVM-based models, while the simpler DT algorithm

performed poorly with an accuracy of only 0.67. This is no doubt due to the complex nature of the stock market (discussed previously in Section 2.1.4), which more simple ML algorithms such as DTs can have difficulty capturing the complexities of. Another example of this is shown in work by Ghani et al., where the authors used Linear Regression (LR), 3-month average, and Exponential Smoothing to predict the stock prices of Amazon alongside others [31]. The best performance of these three coming from the Exponential Smoothing method, which only resulted in R^2 and MSE values of 0.6938 and 363.83 respectively. In June 2017 Krauss et al. compared the use of Gradient-Boosted Trees (GBTs), RFs and Deep Neural Networks (DNNs) to forecast the S&P 500 stock market in R and Java, finding that RFs outperform the other chosen methods [32]. Then, by combing the predictions of all the aforementioned methods using ensemble techniques the authors were able to outperform each individual model, noting the potential of these ensemble methods in future research. The authors did however state that with careful hyperparameter optimisation the DNN may still yield advantageous results. Following on from this, Zhong and Enke developed their own DNN to classify the future trends of assets belonging the SPDR S&P 500 ETF based on a 60-feature dataset [33] in 2019. When comparing their results to a regular Artificial Neural Network (ANN), the DNN provided greater returns and lower risks in a given trading simulation. The authors also noted that despite the number of layers in the DNN increasing issues regarding overfitting still remained under control unlike in the ANN, highlighting the use of DNNs over the simpler ANN. In October of 2020, Wu et al. used ten Taiwanese and American stocks to compare the use of a Convolutional Neural Network (CNN) to predict trends in the movement of financial asset prices [34]. Despite CNNs commonly being used to only analyse visual imagery, their results found that CNNs have great potential to be used for stock market prediction, and the authors compared their use with SVM and NN-based approaches finding that CNNs outperformed them both across all analysed stocks with an average accuracy of 0.826 compared to 0.595 and 0.648 respectively.

2.2.2 LSTMs for Stock Market Prediction

When compared with previous ML algorithms, recent research has found an advantage in using LSTM models for stock market prediction due to their ability to learn long-term dependencies. In 2018 Fischer and Krauss applied an LSTM to for financial market prediction of the S&P 500 from 1992 until 2015, benchmarking them against DNN, RF and LR models implemented in Python's Keras, TensorFlow and Scikit-Learn libraries [35]. Their work found that LSTMs "constitute an advancement in this domain", outperforming the other models "by a clear margin", giving higher returns when applied with a given trading strategy. Nabipour et al. furthered this in 2020 by comparing the use of LSTMs on a ten-feature dataset of the Tehran stock exchange across 10 years [36]. Their results were compared against the eight other common ML models, namely DT, Bagging, RF, Adaboost, Gradient Boost, XGBoost, ANN, and RNN models. The author's work found that the LSTM was the most powerful of all the implemented models, consistently achieving the lowest error measurements (MSE, rMSE, MAPE, and MAE) when predicting stock values 1, 2, 5, 10, 15, 20 and 30 days ahead. The authors did however note the issue of large runtimes involved in training the LSTM model compared to the simpler models. In 2020 Vignesh collected the open, close, low, and high prices of assets from Microsoft and Yahoo from 2011 to 2015 to predict the future value of each company's stock by using an SVM and LSTM, implemented using Python's Scikit Learn library [37]. Their work found a mean accuracy of 0.652 and 0.668 for the SVM and LSTM respectively, with the author noting the work "proves LSTM work better compared to back propagation and SVM algorithms". In 2019, Lai et al. used an LSTM to predict stock exchange values of four Taiwanese companies from 2009 to 2018, achieving MSE of 1.9 and RMSE of 1.3 [38]. Nikou et al. improved on this in the same year, used their LSTM to predict the closing price of EWU stocks from Jan 2015 to Jun 2018, achieving RMSE of 0.306543 and MAE of 0.21035 [39]. This was further improved by Rana et al. who's LSTM to predict the closing stock

price of two Spanish companies achieved an RMSE of 0.0151 [40]. In the same year, Fazelia and Houghten used their own model to predict the opening price of the S&P 500 across 5 years to apply to their own trading strategy, achieving a return on investment (ROI) of 6.67% [41].

More recently, work has been applied on the combination of LSTMs with another ML model using hybrid networks in order to increase prediction accuracy. For example, in 2020 Long et al. used an LSTM in combination with a CNN to predict information about the Chinese stock market. The CNN was used to extract trading features from the dataset before being fed into the LSTM, achieving accuracy of 0.7310 [42]. Likewise, Li et al. extracts investor sentiment from forum posts using Naïve Bayes before feeding the data (alongside stock information) into an LSTM to predict the Chinese stock market, again achieving a strong accuracy of 0.8786 [43]. Chung and Shin propose a hybrid Genetic Algorithm-LSTM model that to predict the price of the Korean Stock Price Index [44], with the model giving better results for MSE, MAE and MAPE compared to the given benchmark, meanwhile Zhang et al. use an LSTM with Generative Adversarial Network (GAN) to predict the S&P 500 price, with the model outperforming a basic LSTM across all performance metrics [45].

2.2.3 Attention & Wavelet Transformation for Stock Market Prediction

At the time of its writing, when Qui et al. analysed recent research for their paper they found the LSTM model based on the attention mechanism was common in speech and image recognition but was rarely used in finance, however, this has changed in the years since that paper was published. In 2019 Xu and Keselj experimented with the use of attention-based LSTMs to predict future US stock market movement based on stock price, technical indicators, and finance tweet sentiments [46]. By comparing the attention-based model with a regular LSTM the authors were able to prove the impact of the attention mechanism, noticing “a moderate improvement over the traditional LSTM model”. Li et al. also used an attention-based LSTM to predict the closing price of the CSI-300 stock market, and by adding the attention module the authors managed to decrease the MSE of the model from 0.1050×10^{-3} to 0.996×10^{-3} [47]. Gu et. al did likewise for to predict agricultural commodity prices in the South Korean market, finding that their results achieved a 1.41% to 4.26% lower MAPE than that of benchmark models [48].

The use of wavelet transformations however has been in application for some time, such as in 2016 when Pesio and Honchar used the Keras framework to implement different NN architectures to predict the S&P500 stock market indices, namely the Multi-Layer Perception (MLP), Convolutional Neural Network (CNN) and LSTM models [49]. Their results found that CNNs outperformed the other proposed models with MSE of 0.2491 and accuracy of 0.536, but by applying a Haar wavelet transformation to the data during pre-processing the authors managed to improve these results to 0.24 and 0.55 respectively, highlighting the advantage of wavelet transformations on financial datasets. Skehin et al. did similar, applying Haar and Daubechies wavelet transformations to predict the closing price of large multinational companies such as Amazon, Facebook and Google using different LSTM topologies [50], while Li et al. showed that using wavelet transformations with LSTMs to predict East Asian Stock indexes provides greater performance when compared to an original LSTM model, with MAPEs reducing across all market implementations [51].

2.2.4 Trading Strategies using Machine Learning

While the use of Machine Learning techniques such as LSTMs to anticipate stock prices has grown increasingly prevalent, integrating model predictions with an algorithmic investing strategy to make trading choices is less widespread, however in recent years there has been a growing number of studies that have found success in combining the two. Jothimani et al.

used an LSTM to forecast the NIFTY 50 and Bombay Stock Exchange Sensitive Index values for the Indian equity market in 2015 [52]. After undergoing a wavelet transformation, the data was fed into an ANN and SVR model to create the final forecasts. The authors then utilised a rule-based trading strategy to evaluate the return on investment based on the anticipated values and found that their model outperformed the return of investment (ROI) of standard Buy-and-hold strategies. Fischer and Krauss experimented with the use of LSTMs for financial market prediction for the S&P 500 stock market, benchmarking their models against deep network, random forest and logistic regression models [53]. The authors then synthesised their findings into a simple rule-based trading strategy for selecting 'winning' and 'losing' stocks, discovering that their approach of purchasing short-term extreme losers and selling short-term extreme winners yielded daily returns of 0.23%. Touzani & Douzi implemented an LSTM and GRU alongside a trading strategy to predict the short-term and medium-term closing price of the Moroccan Stock Market [54]. The authors used the forecasted values to estimate an expected return then held stocks whose return value outperformed a certain threshold value. The authors manged to achieve promising findings, with an annualised return of 27.13% over the testing period. Michanków et al. used LSTM networks to forecast the values of the BTC and S&P 500 indices from 2013 to the end of 2020 and used these forecasts to generate buy and sell investment signals to implement in an investment strategy [55], while Troiano et al. utilised three different trading strategies based on their LSTM forecasted values for the DJIA closing price [56], both finding promising results.

3. Methodology

This chapter focuses on the design of proposed system, the software used to build it and how this design is subsequently implemented into a fully functional system. Firstly, the development environment and project dependencies are discussed in Section 3.1. The overall architecture of the system and how it was built are then explained in Section 3.2, along with samples of the code and novel algorithms employed. Finally, the metrics used to evaluate model performance is then discussed in the last section.

3.1 Prerequisites

3.1.1 Development Environment

The implementation of the project is entirely conducted in a Google Colab [57] notebook running Python 3 [58]. Colab is a web-based application that integrates live code, computational output, data visualisations, equations and explanatory text into a single document known as a notebook. Due to their flexibility and interactivity, notebook-based editors like Colab and Jupyter [59] have gained popularity with data scientists in recent years as code can be easily executed, modified and re-ran in sections while being combined with images, plots, and text to enhance explainability. For the aforementioned reasons, alongside Colab's ability to allow written software to be stored on the cloud and executed on powerful Google GPUs free of charge, the decision was made to implement this project using Colab.

3.1.2 Project Dependencies

Regarding project dependencies, data preparation for the project relies on the Python libraries NumPy [60] and Pandas [61]. NumPy is a library that facilitates the usage of large multi-dimensional arrays, whereas Pandas provides data manipulation and analysis techniques not included in a standard python installation. Due to their speed, effectiveness, and ease of use both libraries have become ubiquitous in data science projects, with a large percentage of ML libraries requiring both as a pre-requisite. Available functionality from both libraries will be utilised to load, analyse and edit the financial datasets, presenting them in standardised formats able to be analysed by the vast majority of ML libraries. Visualisation for these datasets is provided by Plotly [62], Seaborn [63], and MatPlotLib [64], extensions of NumPy that provides interactive and customisable figures, plots, and charts with the added benefit of enabling these plots to be embedded in notebook-based editors such as Colab. Financial datasets will be loaded into the notebook using the popular library yfinance [65] to allow users to run the code without needing to download the relevant .csv files. For data pre-processing, the Scikit-Learn data analysis library is imported to utilise its evaluation metrics and feature scaling methods which provide normalisation and standardisation upon NumPy datasets, and the open-source PyWavelets [66] software is also utilised to provide the wavelet transformations used to de-noise the volatile financial data. The deep learning LSTM networks are developed with Keras [67] on top of Google TensorFlow, one of the largest and most powerful libraries for large-scale customisable Neural Networks, and the attention layer is implemented using Philippe Remy's 'Attention' library [68].

3.2 Design & Implementation

As seen in Figure 7, a very simple flowchart of the project, the proposed system can be divided into two distinct parts. The first component entails predicting stock market prices using an LSTM network with Attention-based mechanisms, similar to the work of Qui et al. whose findings this study intends to reproduce. In the next section (Section 3.2.1), further information about the design of this Prediction Model is presented. This section can be further subdivided

into subsections, with Data Pre-processing involving the wavelet transformation, scaling, and restructuring of the datasets (Section 3.2.1.1), Network Establishment involving the creation and training of the LSTM+Attention Model (Section 3.2.1.2), and Model Predictions involving the model's final stock price forecasts (Section 3.2.1.3). The second part of this project addresses the incorporation of the model's projections into an algorithmic investing strategy, which will create buy or sell signals and permit the development of profitability indicators based on the WLSTM+Attention model's forecasts. The design of this Trading model is detailed later in Section 3.2.2.

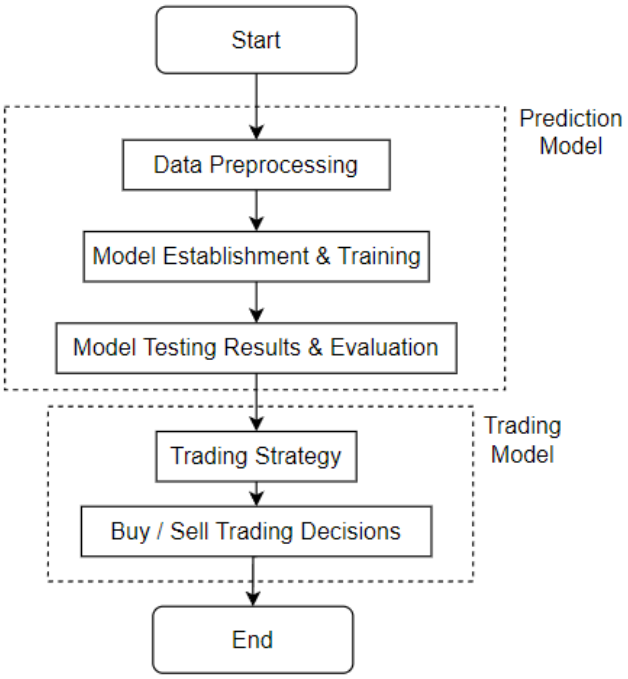


Figure 7 - Simple Overview Diagram of the Project

3.2.1 Prediction Model

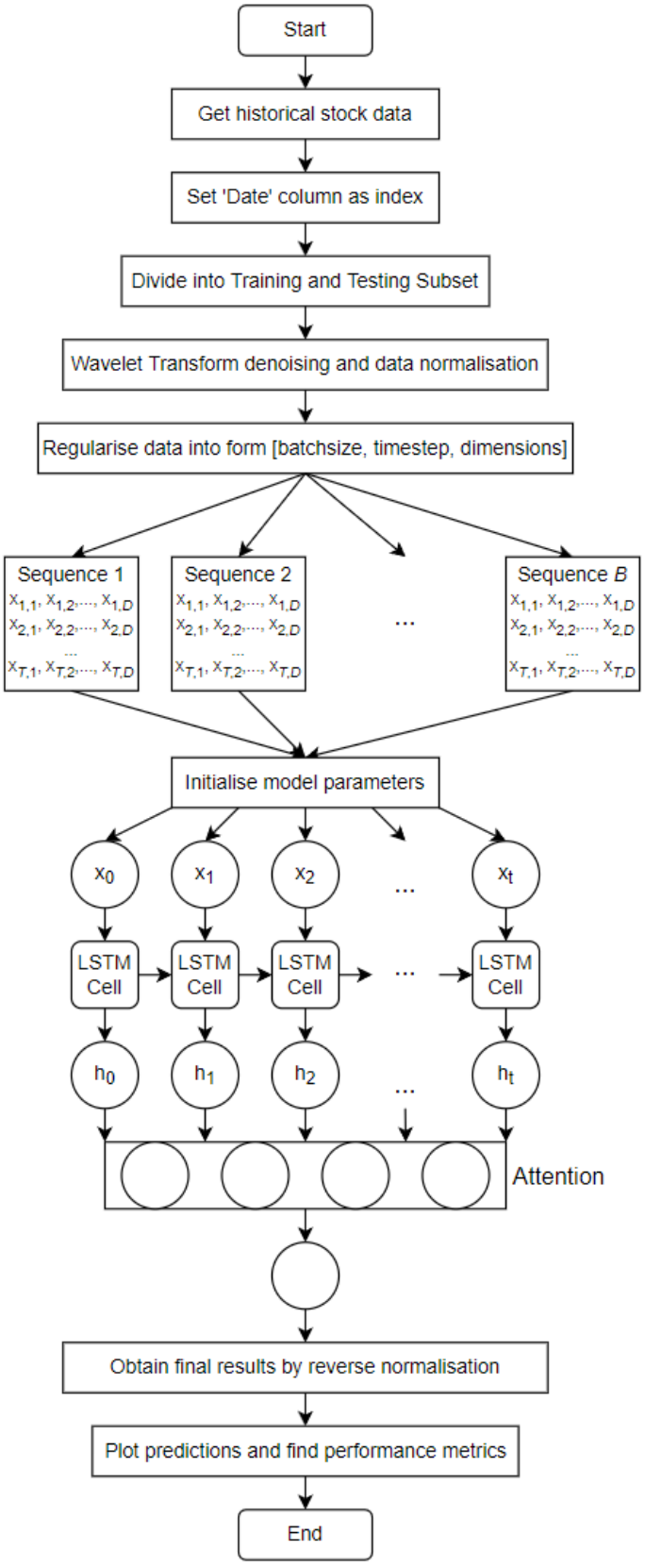


Figure 8 - Flowchart of LSTM-Attention Prediction Model

3.2.1.1 Data Pre-processing

The necessary information to forecast stock prices will be accessed from Yahoo! Finance [69]. Yahoo! Finance is a media site that delivers financial news, statistics and opinions and enables users to view and download the historical stock indices data of every publicly listed company and major stock index in the world. The historical data is available for download in the widely used .csv format and consists of import financial metrics from across the stock's lifespan, including opening and closing prices, high and low prices, adjusted close prices and daily volume of shares traded. In Section 4.1, further information about the accessible financial data and certain instances will be provided. The data will be imported from Yahoo! Finance using yfinance and read into a NumPy dataframe prior to pre-processing.

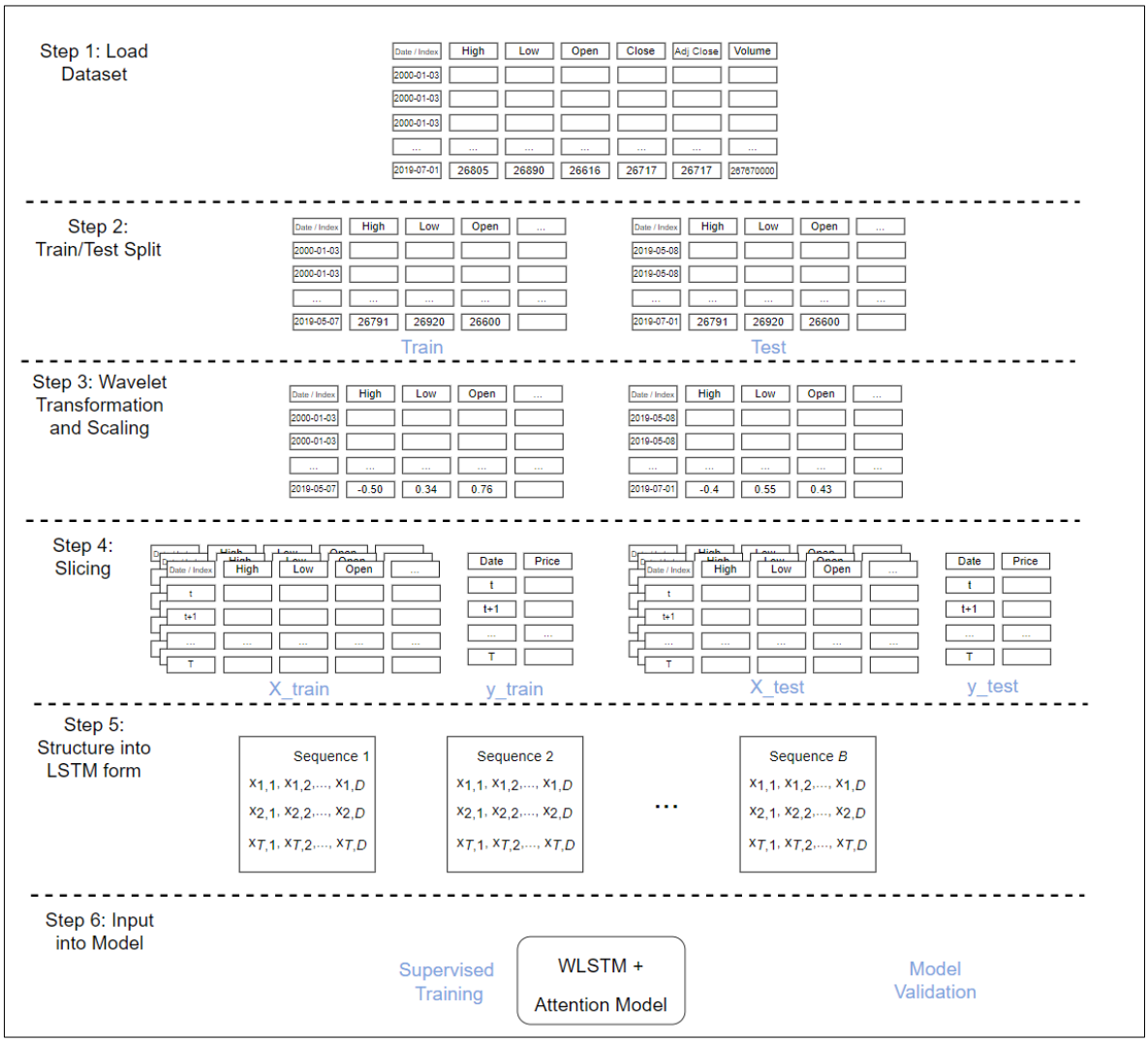


Figure 9 - Stages in the Data Pre-processing section

Yahoo Finance's financial data sometimes contains a column for each sample's designated date. This is an unnecessary feature for this predictive modelling problem; thus, this column will be utilised as the dataframe's index, which is also beneficial as MatPlotLib will display the dates of each sample along the X axis when charting the dataframe. Commonly, in order to evaluate the performance of machine learning algorithms, the dataset is divided into two subgroups. The first subset, known as the training dataset, is utilised to fit the model. The second subset, known as the testing dataset, is used to evaluate this fit by comparing the model's projected values for a particular sample with the true values. The imported dataset will be divided into these two sections in order to quantify and compare model performance, and information on the specific performance metrics used is discussed later in Section 3.3.

The example code below demonstrates how the train test spit is implemented in the system. The label column is created previously and represents the actual value the model should predict for the next day (so is the next day's closing price). When splitting a sample based on the date the two sets will share a sample, so in order to avoid data leakage the last sample in the training dataset is dropped.

```
[505] def train_test_split(df):

        x = (df.loc[:, : 'Volume'])
        y = (df.loc[:, 'Label': ])

        x_train = x[:split_index]
        x_test = x[split_index:]

        y_train = y[:split_index]
        y_test = y[split_index:]

        # drop last row of train sets to avoid sample repetition
        x_train.drop(x_train.tail(1).index,inplace=True)
        y_train.drop(y_train.tail(1).index,inplace=True)

        return x_train, y_train, x_test, y_test

      x_train, y_train, x_test, y_test = train_test_split(df)
```

As discussed in Section 2.1.4, financial data is complicated and volatile resulting in random oscillations that interfere with the primary relevant signal. To combat this, a wavelet transformation will first be applied to the training and testing set using PyWavelets in order to reduce this noise and improve the accuracy of any forecasts made. The example code below demonstrates how a wavelet transformation with the haar function is applied to a given dataframe in Python, and Figure 10 shows the smoothing effect of the wavelet transformation on a given dataset. The wavelet transformation is only applied on the training dataset, and as a consequence decreases the number of samples in the dataset by a factor of two.

```
# Apply wavelet transformation on given dataframe
def wavelet(data):
  df_wav = pd.DataFrame()
  for column_name, column_data in data.items():
    (cA, cD) = pywt.dwt(column_data, 'haar')
    df_wav[column_name] = cA
  return df_wav

x_train_wav = wavelet(x_train)

# Display Datasets
print('X_train shape before wavelet transformation == {}.'.format(x_train.shape))
print('X_train shape after wavelet transformation == {}.'.format(x_train_wav.shape))
```

```
X_train shape before wavelet transformation == (4873, 6).
X_train shape after wavelet transformation == (2437, 6).
```

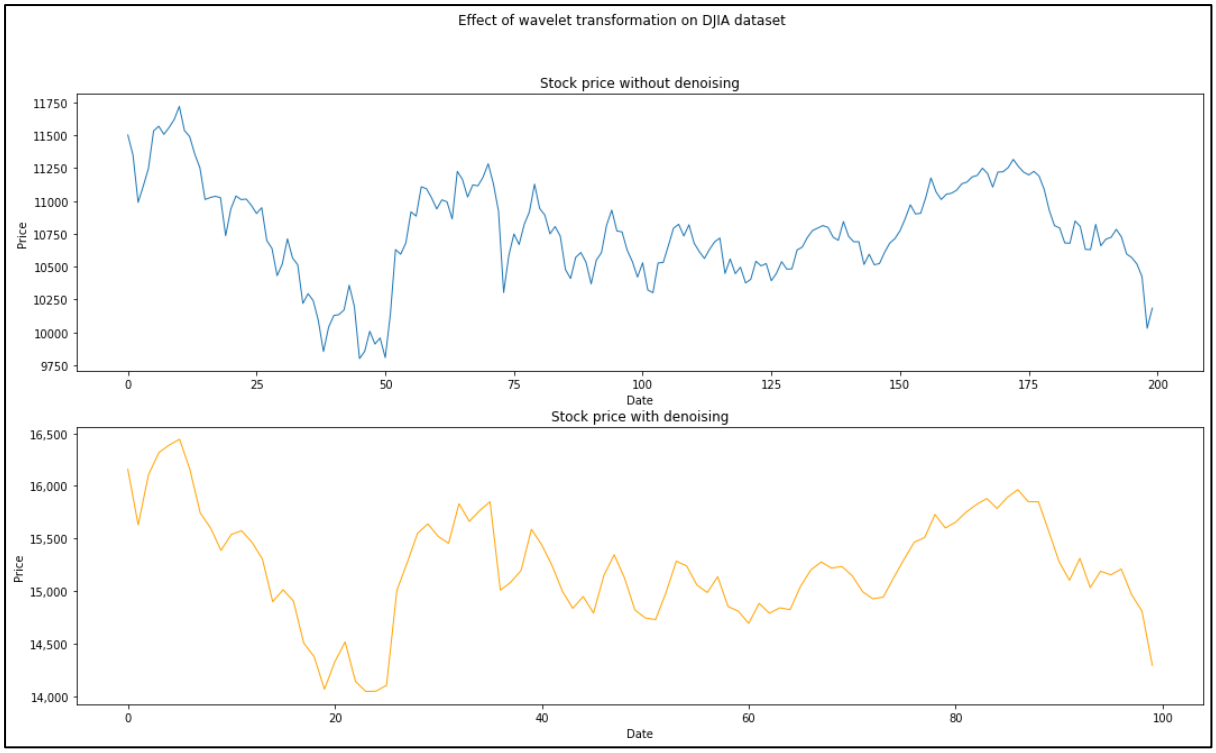


Figure 10 - Effect of Wavelet transformation on DJIA Dataset

Because the input data consists of two categories of data, i.e., stock price and volume, standardisation and normalisation techniques are also applied using Scikit-Learn's scaling methods. These feature scaling techniques are used to bring all metrics down to the same scale, ensuring that larger data features do not influence the model solely due to their magnitude and creating a smoother gradient descent during training, which aids backpropagation algorithms in reaching the cost function minima more quickly. Scikit-learn's MinMax scaler is used for data scaling as shown in the code below:

```
[17] # Load the scaler, and normalise the input training dataset
     scaler_X = MinMaxScaler(feature_range=(-1,1))
     X_train_norm = scaler_X.fit_transform(x_train_wav)

     # Normalise the output training dataset using a different scaler
     scaler_y = MinMaxScaler(feature_range=(-1,1))
     y_train_norm = scaler_y.fit_transform(y_train)
```

In order to prepare sequence data for input into any LSTM model, the data must be reformatted into a three-dimensional matrix with form [B, T, D]. B denotes the batch size, or the total number of sequences in the dataset. T stands for the time step, or the size of each sequence. D represents dimensions, or the number of features present at each timestep. Consider, for the sake of explanation, a 3D matrix with the following dimensions: (4824, 50, 6). This means that the LSTM will operate on 4824 distinct sequences, with each sequence consisting of 50 samples and each sample containing six features. Before feeding both the training and testing datasets into the LSTM-Attention model, they will both be reshaped into this 3D matrix structure. The implemented code for structuring a dataset in this way is shown below. As you can see, before the input training dataset is a regular 2D matrix, but now after the change has dimensions (3135, 150, 6), when using a window size/timestep of 150.

```
[20] X_train_S, y_train_S = [], []

     for i in range(PARAM_WINDOW_SIZE, X_train_norm.shape[0]):
         X_train_S.append(X_train_norm[i - PARAM_WINDOW_SIZE : i])
         y_train_S.append(y_train_norm[i])
     X_train_S = np.array(X_train_S)
     y_train_S = np.array(y_train_S)

     print('X_train shape == {}.'.format(X_train_norm.shape))
     print('y_train shape == {}.'.format(y_train_norm.shape))
     print('X_train_S shape == {}.'.format(X_train_S.shape))
     print('y_train_S shape == {}.'.format(y_train_S.shape))

     X_train shape == (3285, 6).
     y_train shape == (3286, 1).
     X_train_S shape == (3135, 150, 6).
     y_train_S shape == (3135, 1).
```

3.2.1.2 Model Establishment & Training

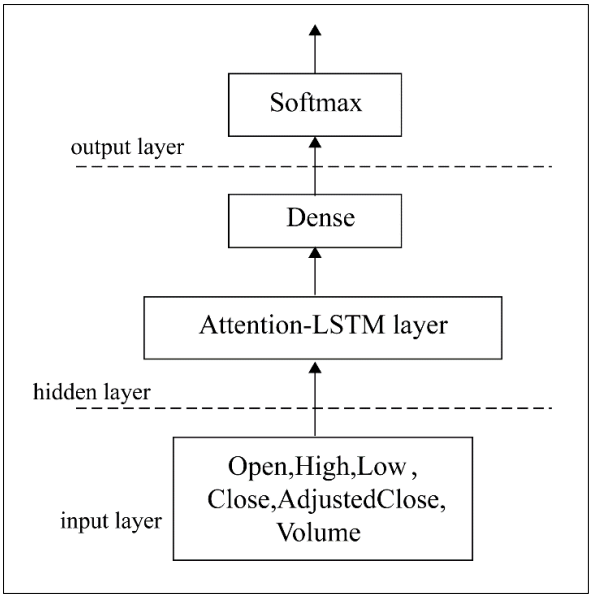


Figure 11 - LSTM Network structure used by Qui et al. Taken from [4]

As stated in Section 3.1.2, the proposed LSTM-Attention model is built using Keras on top of Google TensorFlow and Philippe Remy's 'Attention' package. As the first objective of this project is to try and replicate the work done by Qui et al., the model will initially consist of three layers: an input layer, a hidden layer, and an output layer. As shown in Figure 11 from Qui et al., the hidden layer consists of an LSTM and Attention layer followed by a dense layer and the output layer will consist of a single node representing the predicted output of the network. This model is then compiled, and while the authors do not specify which optimiser they have used, I selected the Adam optimiser based on studies by Doshi [70] and Gupta [71] that compared the performance of several Deep Learning Optimizers, with Adam achieving the best results in both experiments. The example code below demonstrates how a the WLSTM+Attention model could be implemented and complied in Python:

```
[514] from tensorflow.keras import Input
      from tensorflow.keras.layers import Dense, LSTM
      from tensorflow.keras.models import load_model, Model
      from attention import Attention
      from tensorflow.keras.optimizers import Adam

      # Input Layer
      model_input = Input(shape=(PARAM_WINDOW_SIZE, X_train_norm.shape[1]))

      # LSTM Layer with 10 Nodes
      x = LSTM(HIDDEN_LAYER_NODES, return_sequences=True)(model_input)

      # Attention Layer
      x = Attention(HIDDEN_LAYER_NODES)(x)

      # Output Layer
      x = Dense(1)(x)

      # Complie Model
      model = Model(model_input, x)
      model.compile(loss='mean_squared_error', optimizer='Adam')
```

The complied model will then be trained by fitting the model to the reshaped 3D training dataset, and in order to gain a full understanding of model performance a validation set will be utilised during training. The validation set is a subset of the training set that is used to evaluate the model after each epoch. This validation process indicates that the model training is progressing in the right direction and can help avoid overfitting, which occurs when the model becomes very good at classifying samples in the training set but is unable to generalise and make accurate classifications on unseen data. The history of the trained model will be stored to plot the decrease in training and validation loss as the number of epochs increases, which can provide a useful indication to the model's speed of convergence (based on its slope), whether it has already converged (if the loss has plateaued), or whether the model may be overfitting (an inflection in the validation line). The parameters to be used during this stage such as batch size and epochs will be addressed in greater detail during Section 4.1.1.1. An example of the plot produced at the end of this section is shown below in Figure 12. The smooth decrease in validation loss shows the model is not overfitting and is slowly converging as the number of epochs increase.

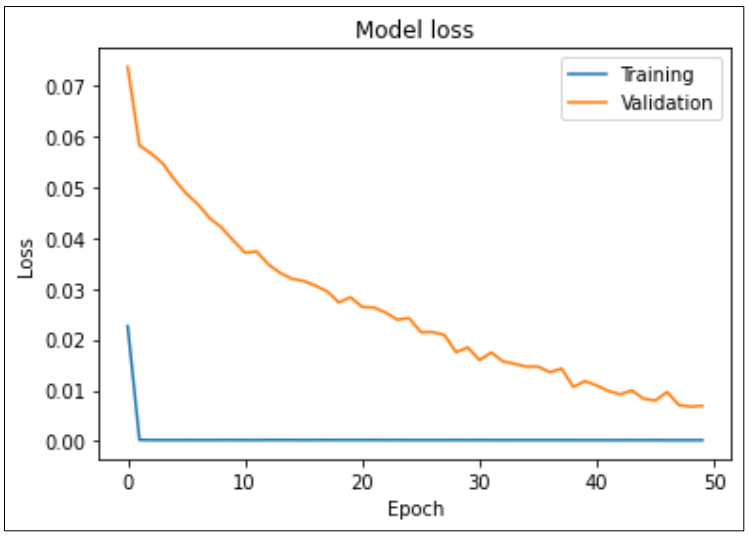


Figure 12 - Example Model Loss plot comparing Training and Validation loss

3.2.1.3 Model Testing & Evaluation of Results

After the model has been trained, its fit is assessed by feeding in the training data and comparing the model's predicted values to the actual values. Prior to this, the last *T* samples of the training dataset must be concatenated with the testing dataset to ensure the model has prior knowledge upon which to base its initial sample. *T* relates back to the time step or window size described in section 3.2.1.1. This modified testing dataset must then be normalised using the same scalar as the training dataset to prevent data leakage, then regularised into the same [B, T, D] structure as the training set for input into the LSTM-Attention model. Once the model's predictions have been generated, they will be reverse-normalized and plotted against the actual values using MatPlotLib to provide a visual representation of the accuracy of the forecasts. The real and projected values will then be processed using Ski-kit Learn to provide performance metrics to measure and compare model performance. Information on the particular performance measures employed is included later in Section 3.3. The code to predict model values is shown below, and after the predictions are generated they are reverse normalised using the same scalar to transform them back to real values.

```
[27] # Find the predicted values
     predicted_stock_price = model.predict(X_test_S)

     # Reverse normalise the values
     predicted_stock_price = scaler_y.inverse_transform(predicted_stock_price)
```

An example of the final plot of real values against predicted values at the end of model testing is shown below in Figure 13.

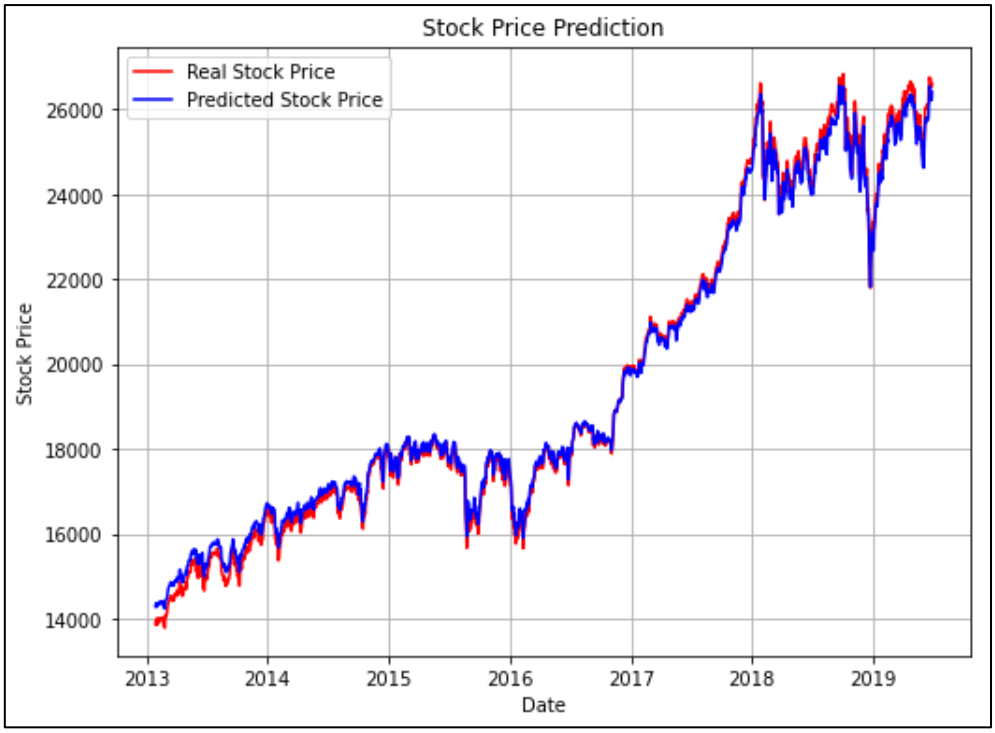


Figure 13 - Example of the final plot of Real vs Predicted stock prices generated by the model

3.2.2 Trading Model

To evaluate the performance of the WLSTM+Attention model on stock market prediction, a simulation is developed implementing two different trading strategies. The first is called a 'naïve' strategy, though the strategy itself is relatively advanced compared to prior research in

the field but is named ‘naïve’ due to the lower number of features (six) the model bases its predictions off compared to more sophisticated trading algorithms. This ‘naïve’ strategy purchases stock if the value is expected to appreciate and shorts stock if the value is expected to fall, implying selling stock it does not have yet in hopes of buying it later at a lower price. The second is a random trading strategy which will buy and sell randomly, acting as a baseline strategy.

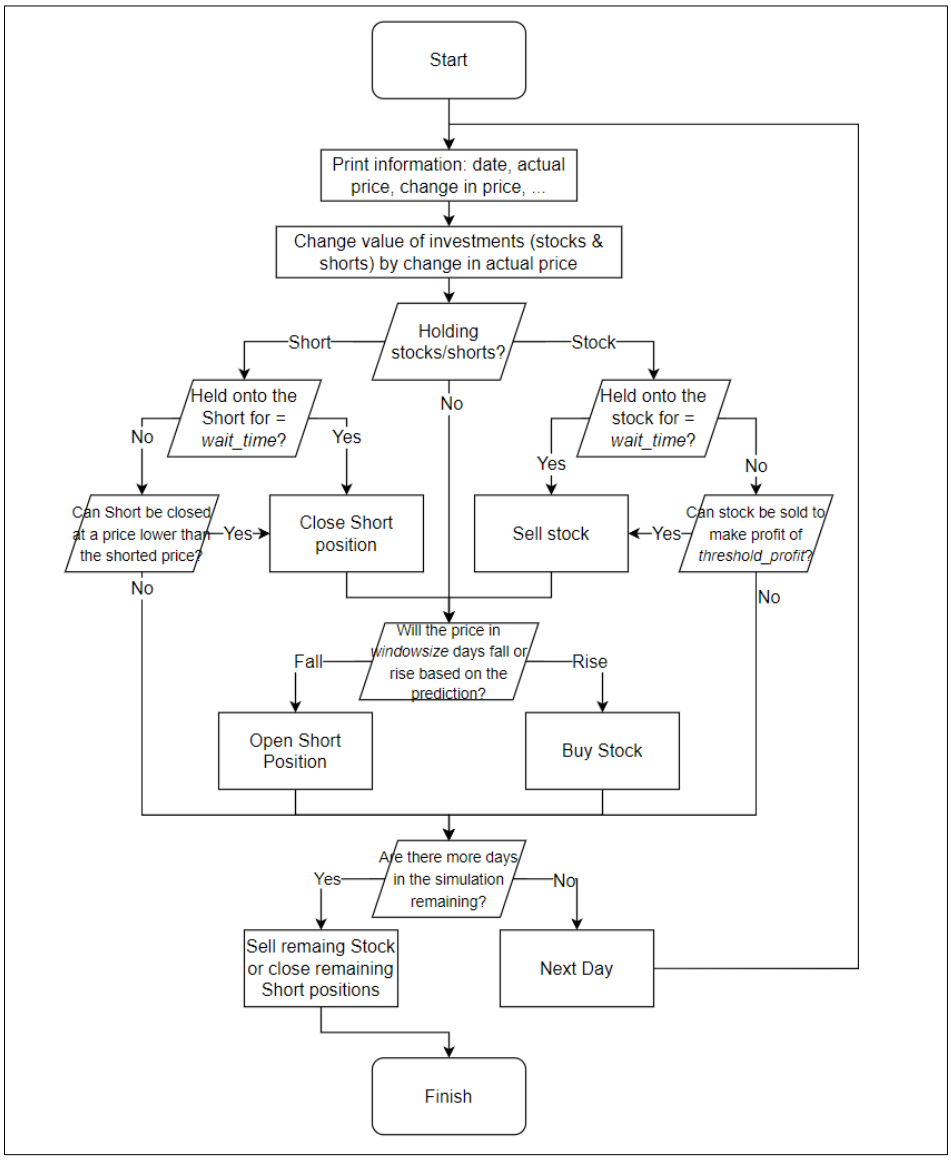


Figure 14 - Flowchart of the simple Trading Strategy

A flowchart of the naive trading strategy is shown in Figure 14. The system begins the simulation with a predetermined sum of money, in this example 1,000,000 USD, and then attempts to estimate the stock's closing price for the next day. If the stock price is anticipated to increase, the system will transmit signals to acquire 100,000 USD worth of stock, and if the price is anticipated to decrease, the model will short 100,000 USD worth of stock. The system will maintain a position for *wait_time* days following a purchase, and if during that period the acquired stock can be sold for a profit of *threshold_profit* the system will instantly sell the stock, or if during that period stock can be purchased at a price lower than the shorted price the system will immediately close the short position. If the *wait_time* period has expired and the stock or short position has not been sold or closed, the system will promptly terminate the position and, if required, incur a loss. At the beginning of every day, the system will calculate the percentage change in the actual stock price and increase or decrease the value of the

stock or shorts it has acquired by the same amount. The duration of the simulation is proportional to the number of samples in the testing dataset used to create the model predictions. Upon completion of the simulation, all final positions are closed in order to calculate the final profitability metrics covered next in Section 3.3. Important to note, the proposed methodology is naïve and based on the assumptions that transaction costs are not considered and have no impact on profitability, and that created buy and sell orders are instantly executed and have no impact on the market. The random strategy functions similarly, except that instead of forecasting the value for the following day the system creates a random number that determines whether the model will buy, sell, or hold on a particular day. Recording the portfolio's value at the conclusion of each day enables us to plot a graph of the naïve and random portfolio value as the testing period progresses, as seen in Figure 15.

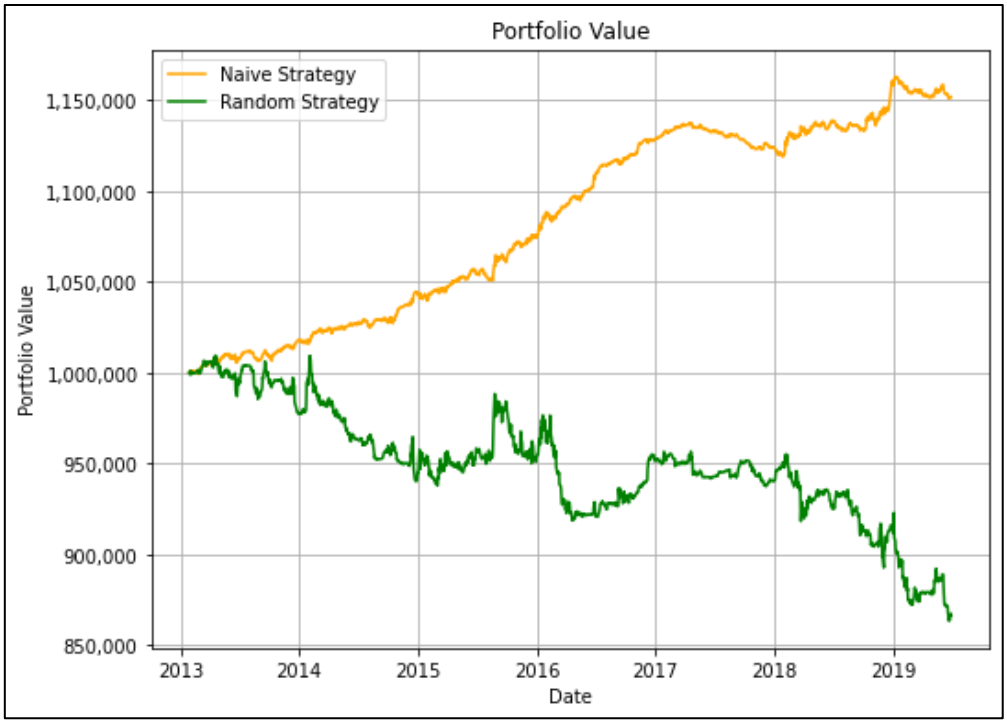


Figure 15 - Example Portfolio Value graph produced at Trading Strategy Completion.

3.3 Performance metrics

To ensure accurate replication of the results found by Qui et al, the same four performance metrics used by the authors shall be used to evaluate the accuracy of the implemented WLSTM+Attention model: mean square error (MSE), root mean square error (RMSE), mean absolute error (MAE), and coefficient of determination (R^2). As the first three metrics quantify error, the closer they are to 0 the more accurate the model's predictions are, however as R^2 measures variance the closer it is to 1 the better the model's fit is.

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2 \quad (9)$$

$$RMSE = \sqrt{MSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2} \quad (10)$$

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|\hat{y}_i - y_i| \quad (11)$$

$$R^2 = 1 - \frac{\sum_{i=1}^{N}(\hat{y}_i - \bar{y}_i)^2}{\sum_{i=1}^{N}(y_i - \bar{y}_i)^2} \quad (12)$$

Error measurements merely evaluate the capacity of the prediction model to accurately estimate future values, but what truly matters in financial market practice is the profitability of the trading strategy. To evaluate the success of the model's market simulation forecasts, the total return (R), or profitability is calculated, as illustrated in Equation (13). $Portfolio_T$ reflects the value of all assets in the account at the conclusion of simulation period T, whereas $Portfolio_{t_0}$ represents the value of all assets at the beginning of simulation period t_0. A larger return indicates greater profitability and a more effective model implementation. Using the total return as opposed to analysing successful transactions enables prospective losses from mistakenly forecasted samples to be offset by gains from successfully anticipated samples, and vice versa. As an objective on this project is to test the proposed model on different datasets than those used by Qui et al., a profitability per day (PPD) metric is also calculated to ensure the presence of a metric that can allow profitability comparisons between datasets which have a much greater number of testing samples, as most markets will naturally increase in value over time so a dataset with more testing samples has a greater probability of ensuring profitable returns. PPD is calculated by finding the profit generated during the trading simulation and dividing it by the total number of trading days t_0 as shown in Equation (14) below.

$$R = \left(\frac{Portfolio_T}{Portfolio_{t_0}} - 1\right) * 100 \quad (13)$$

$$PPD = \frac{Portfolio_T - Portfolio_{t_0}}{T} \quad (14)$$

Despite the ubiquitous use of error metrics such as RMSE, MSE, and MAE in time series forecasting studies, recent studies such as Micankow et al. [55] have questioned the use of such metrics in financial forecasting research. Their argument is that the aforementioned error metrics only evaluate the accuracy of the predictions (the difference between the anticipated and observed values) and lack information regarding the direction in which the predicted and observed values have moved since the last sample. Profitability is ensured in simulations of financial forecasting not by making the most accurate estimate of the following day's price, but by properly anticipating the direction of price change. Some models may fail to predict prices near to the actual values, but still successfully anticipate the direction (and possibly magnitude) of change. As a result, these models are unfairly penalised by the error metrics outlined above, although still providing profitable trading strategy simulations. The problem with this is that the majority of academics will still pick model parameters and evaluate model performance based on optimisation of the chosen error metrics rather than simulation profitability. To counter this, the Mean Directional Accuracy (MDA) is implemented, which compares the forecasted direction (either upwards or downwards) to the actual realised direction, a commonly used metric for forecasting performance in economics and finance. MDA is computed by summing the change between the estimated forecast value and the actual forecasted value passed through a sign function for each sample and finding the mean, as in Equation (15).

$$MDA = \frac{1}{N}\sum_{i=1}^{N} sign(y_i - \hat{y}_i) \quad (15)$$

4. Evaluation and Critical Appraisal

This chapter begins with a quick investigation of the datasets utilised in this study. The next section aims to replicate the results of Qui et al. and so experiments are performed to find the optimal parameters, the results of which are then evaluated and discussed. The next section functions similarly but instead uses three different datasets. This chapter is then concluded with a Critical Appraisal that contrasts the work with its original goals and with comparable works in the public domain.

4.1 Datasets

All datasets in the project are obtained from Yahoo! Finance with the baseline stock history dataset containing six variables. The opening price is the price of the first security traded after the market opens, likewise the closing price is the price of the last trade before the market closes. High and Low represent the day's highest and lowest stock prices respectively. Adjusted close is the closing price after splits and dividend distributions are accounted for, while Volume is the total number of shares traded for the given day. An example of a Yahoo! Finance dataset is illustrated in Table 4.1.

Date	Open	High	Low	Close	Adj Close	Volume
2000-01-03	11501.84961	11522.00977	11305.69043	11357.50977	11357.50977	169750000
2000-01-04	11349.75000	11350.05957	10986.45020	10997.92969	10997.92969	178420000
…	…	…	…	…	…	…
2019-06-28	26605.92969	26638.34961	26522.26953	26599.96094	26599.96094	499350000
2019-07-01	26805.85938	26890.64063	26616.21094	26717.42969	26717.42969	267670000

Table 4.1 - Partial Stock Data Sample for DJIA Index

As the primary objective of this project is to attempt to reproduce the findings of Qui et al. [4], the model will be tested and trained using the same datasets. The authors use three datasets during their experiments, the S&P 500, Dow Jones Industrial Average (DJIA), and Hang Seng Index (HIS), the dates of which are listed in Table 4.2 alongside the Train/Test Split used and the resulting number of training and testing samples. The alternative datasets to be processed are the National Association of Securities Dealers Automated Quotations Stock Market (NASDAQ), Deutscher Aktienindex (DAX), and Financial Times Stock Exchange 100 Index (FTSE). These datasets span from the beginning of the millennium to the beginning of the year 2020 and a train/test split of 80% is used to allow for a greater number of testing samples than those used by Qui et al.

Index Name	Start Date	End Date	Train/Test Split Date	Train #	Test #
S&P 500	03/01/2000	01/07/2019	16/05/2019	4874	31
DJIA	03/01/2000	01/07/2019	16/05/2019	4874	31
HSI	02/01/2002	01/07/2019	16/05/2019	4273	31
NASDAQ	01/01/2000	01/01/2022	31/12/2015	4024	1006
DAX	01/01/2000	01/01/2022	18/12/2015	4057	1015
FTSE 100	01/01/2000	01/01/2022	01/04/2016	4085	1021

Table 4.2 - Starting, Ending and Split Dates of the Datasets used in this Project

In Figure 16, each feature of the DJIA dataset is plotted using seaborn to explore the change in each feature as time progresses separately, shown below.

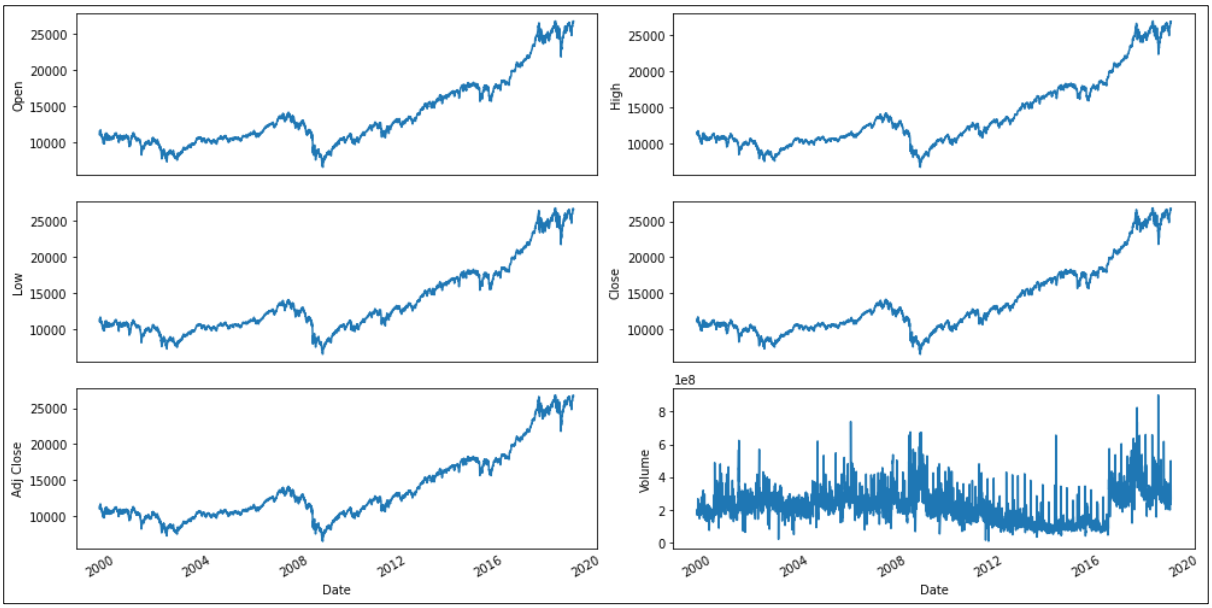


Figure 16 - Visualisation of each feature in the DJIA Dataset

4.1.1 Replication of Qui et al.

In this section, the model is tested using the same datasets as Qui et al. to validate and replicate the authors results, namely the S&P500, DJIA and HSI datasets. The subsequent section seeks to determine the optimal parameters for replicating the findings and the section that follows presents and evaluates the model's outcomes across all datasets.

4.1.1.1 Parameter Selection

A variety of studies have been conducted to identify the appropriate parameters for the implemented WLSTM+Attention model to accurately replicate the results found by Qui et al. There are four primary parameters that may be modified to improve the accuracy of the model's forecasts: Hidden nodes refers to the number of nodes in the LSTM+Attention hidden layer, Window size is the size of each sequence of input data to be input into the LSTM model (the time step), epochs refers to the number of cycles over the whole training dataset that the model will execute during training, and batch size is the number of training samples sent to the network at one time. The performance measures outlined Section 3.3 are used to evaluate each combination of parameters using the DJIA dataset and the results are displayed in Table 4.3 using a colour scale to make it easy to compare the best performing trials. In addition, Appendix A contains plots for all the listed experiments.

Experiment	Nodes	Epochs	Batch Size	Window Size	MSE	RMSE	MAE	R2	MDA	Profitability	PPD
1	10	100	25	1	7721602	2778.777	2755.84	-22.5791	0.586207	-0.35%	-121.496
2	10	100	25	5	1398159	1182.438	1107.95	-3.2695	0.517241	-0.35%	-121.496
3	10	100	25	10	1282720	1132.572	1038.186	-2.91699	0.586207	-0.35%	-121.496
4	10	100	25	25	8960659	2993.436	2931.128	-26.3628	0.448276	-0.35%	-121.496
5	10	100	25	50	3024063	1738.983	1671.938	-8.23445	0.517241	-0.35%	-121.496
6	10	100	25	100	357328.1	597.7693	544.1734	-0.09116	0.551724	-0.37%	-126.503
7	25	100	25	1	2011843	1418.395	1384.72	-5.14348	0.586207	-0.35%	-121.496
8	25	100	25	5	278300.8	527.5422	456.1327	0.150165	0.482759	-0.47%	-162.955
9	25	30	25	5	308839.4	555.7332	483.2196	0.05691	0.448276	-0.47%	-162.955
10	25	100	25	10	215920.1	464.672	413.8921	0.340654	0.551724	-0.62%	-215.105
11	25	100	25	25	251062.7	501.0616	435.2707	0.233341	0.517241	-0.32%	-109.877
12	25	100	25	50	273996.3	523.4465	388.4168	0.163309	0.586207	0.25%	87.23015
13	25	100	25	100	825156.1	908.381	834.3839	-1.51974	0.62069	0.06%	20.15656
14	25	100	1	100	149596.9	386.7776	319.9224	0.543182	0.448276	-0.42%	-146.542
15	25	100	5	100	130858.3	361.7434	300.7878	0.600404	0.551724	-0.42%	-143.877
16	25	100	10	100	150571.7	388.0358	344.209	0.540206	0.551724	-0.47%	-162.955
17	25	100	25	100	242170	492.1077	446.0176	0.260496	0.551724	-0.47%	-162.955
18	25	100	50	100	240213	490.1153	359.3152	0.266472	0.551724	0.36%	122.9784
19	50	100	50	1	491638.5	701.1694	634.0369	-0.5013	0.586207	-0.35%	-121.496
20	50	100	50	5	174319.4	417.5158	338.7747	0.467688	0.448276	-0.23%	-79.9403
21	50	100	50	25	300299.7	547.9961	390.8882	0.082988	0.586207	0.23%	78.22203
22	50	100	50	100	205176.6	452.9642	317.5029	0.373461	0.586207	0.43%	147.7092
23	5	100	50	100	4511232	2123.966	2060.111	-12.7758	0.551724	-0.35%	-121.496
24	5	100	25	100	2826717	1681.284	1623.826	-7.63183	0.62069	-0.35%	-121.496
25	5	100	25	50	4864005	2205.449	2114.199	-13.853	0.586207	-0.35%	-121.496

Table 4.3 - Experiment results for WLSTM+Attention testing using DJIA dataset.

The experiments began with 10 hidden nodes, but after unsatisfactory results, it was decided to raise the number to 25 which proved more effective. After steadily increasing the window size, it was determined that a larger window size of 100 produced stronger results; thus, it was then decided to modify the batch size to identify the model with the highest performance with this fixed window size. After increasing the number of nodes to 50 and varying the window size to determine the best settings, experiment 22 proved to be the most successful. Since Qui et al. state that they utilise ten hidden nodes, it was decided to reduce the number of hidden nodes to five in case the higher node sizes contributed to overfitting in the previous models. However, poor results in all three tests immediately disproved this hypothesis. The epoch size was not varied often as by viewing the plotted validation loss it was determined that the majority of implementations had already converged to a minimum, or the metrics generated with 100 epochs were so poor that it did not appear worthwhile to test again with varied epoch sizes. Similar results found for MDA, Profitability and PPD values between experiments were due to the small size of the testing set, which will be discussed later in Section 4.1.1.2.

Using a window size of 100, 50 hidden layers, 100 epochs, and a batch size of 50 yields the greatest performance across all metrics for the provided dataset, as shown in the table for experiment 22. Figure 17 illustrates the real price movement with the expected price change for this model alongside the portfolio value over time of the implemented trading strategy.

.

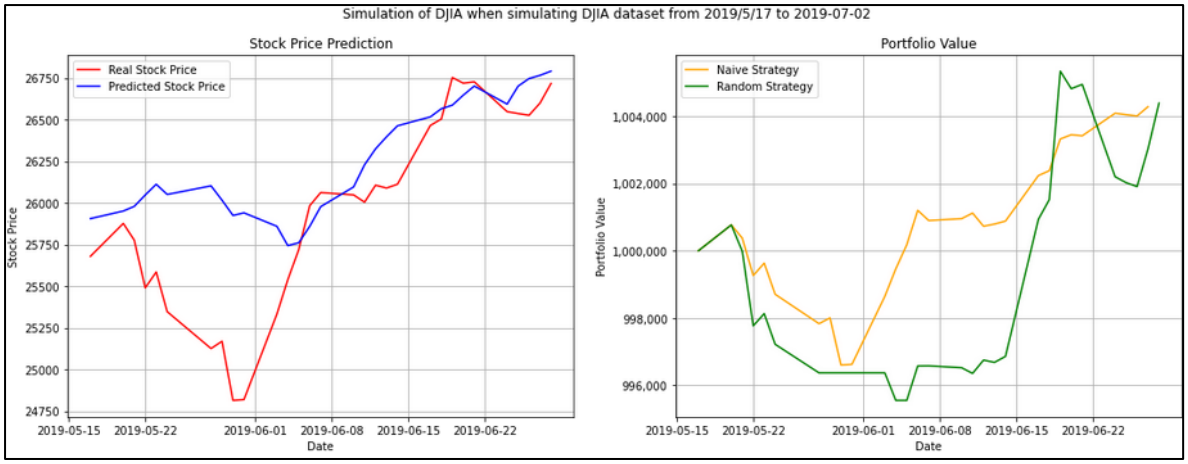


Figure 17 - Results from experiment 22 - the best performing experiment

4.1.1.2 Results and Discussion

To validate the results of Qui et al. the same three stock index datasets the authors use (discussed in Section 4.1) are processed using the three same models, namely a normal LSTM, an LSTM with wavelet transformation (WLSTM), and the proposed WLSTM+Attention Model. Each model is trained and evaluated, and the predicted values are plotted in Figure 18, Figure 19, and Figure 20 with the performance metrics listed and compared in Table 4.4, Table 4.5, and Table 4.6. Qui et al's results are highlighted in grey to aid in comparison.

S&P500	MSE		MAE		RMSE		R^2	
LSTM	0.1208	7555.563	0.2676	76.475	0.3475	86.923	0.8829	-1.089
WLSTM	0.1067	5122.188	0.2470	61.955	0.3267	71.569	0.8965	-0.416
WLSTM+ Attention	0.0546	1834.647	0.1935	35.419	0.2337	42.832	0.9470	0.492

Table 4.4 - Comparison of evaluation indicators between Qui et al. and this work for the S&P500 dataset

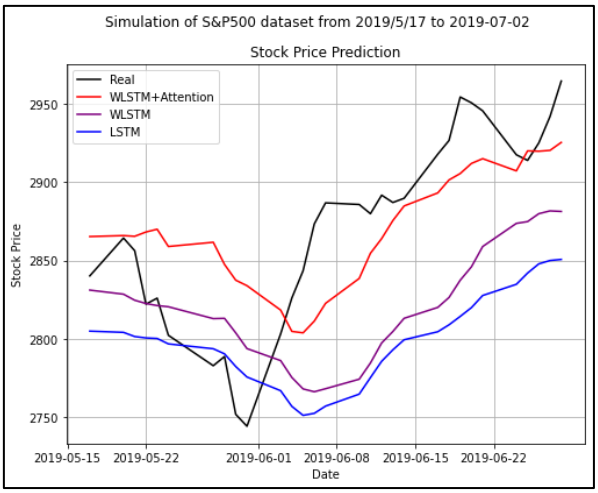


Figure 18 - Model predictions for S&P500 Dataset

DJIA	MSE		MAE		RMSE		R^2	
LSTM	0.0785	6394283.696	0.2360	2486.609	0.2802	2528.692	0.9235	-18.526
WLSTM	0.0488	1666991.246	0.1646	1215.938	0.2209	1291.120	0.9524	-4.090
WLSTM+ Attention	0.0388	124811.921	0.1569	289.017	0.1971	353.287	0.9621	0.619

Table 4.5 - Comparison of evaluation indicators between Qui et al. and this work for the DJIA dataset

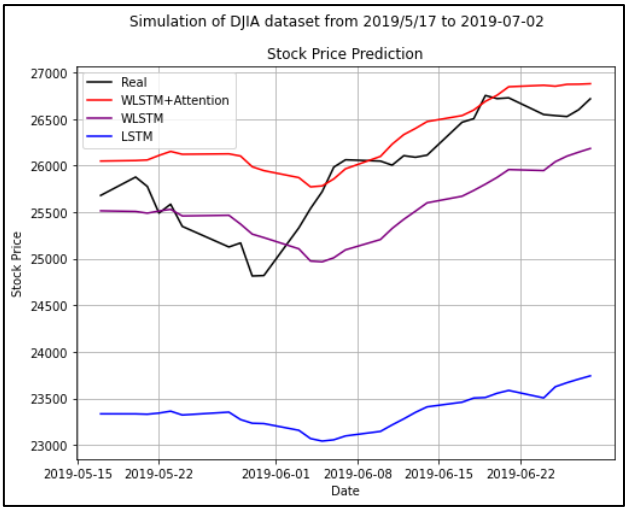


Figure 19 - Model predictions for DJIA Dataset

HSI	MSE		MAE		RMSE		R^2	
LSTM	0.1839	318216.995	0.3031	407.946	0.4288	564.107	0.8097	0.022
WLSTM	0.1335	219382.323	0.2667	386.548	0.3654	468.382	0.8618	0.326
WLSTM+ Attention	0.1176	192034.400	0.2433	384.323	0.3429	438.217	0.8783	0.410

Table 4.6 - Comparison of evaluation indicators between Qui et al. and this work for the HJI dataset

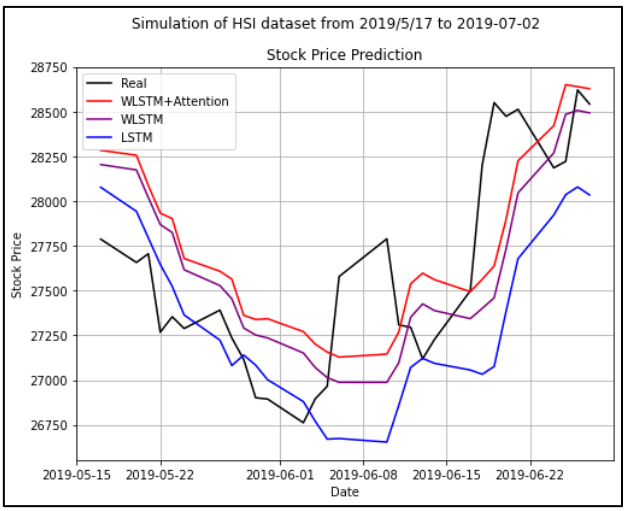


Figure 20 - Model predictions for HSI Dataset

As seen in the tables above, there is a significant difference between the authors' values and the best values this model was able to generate, with much larger MSE, MAE, and RMSE, values and smaller R^2 values across all implementations. In spite of this, the findings demonstrate the benefits of the wavelet transformation and attention mechanism since across all datasets the error metrics were poorest for the simple LSTM model, rose when the wavelet transformation was included, and peaked for the WLSTM+Attention model. The conclusion from these experiments is that I was unable to exactly duplicate the findings of Qui et al., despite conducting a significant number of tests and adjusting the model parameters across multiple iterations.

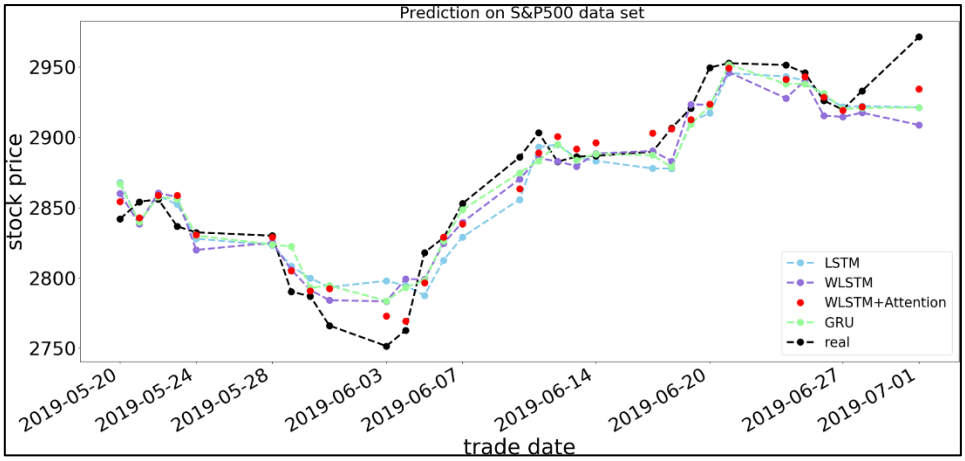


Figure 21 - Qui et al's forecasts for S&P500 Price. Taken from [4].

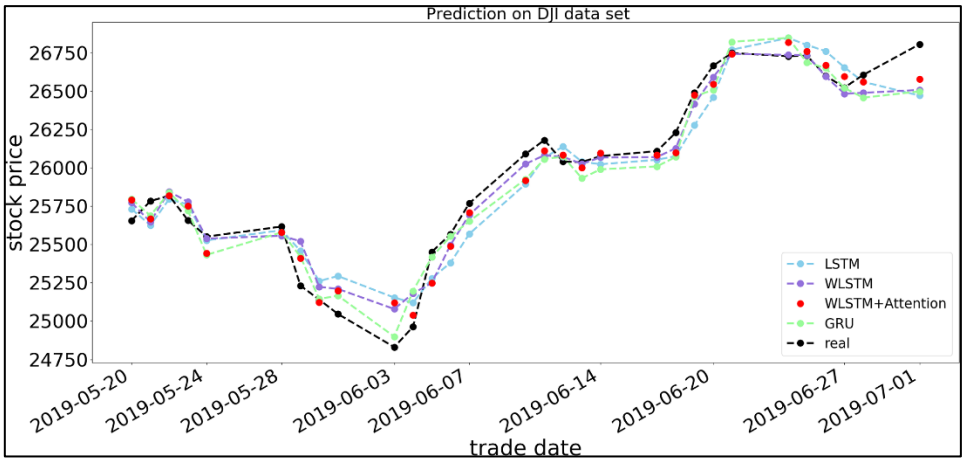


Figure 22 - Qui et al's forecasts for DJIA Price. Taken from [4].

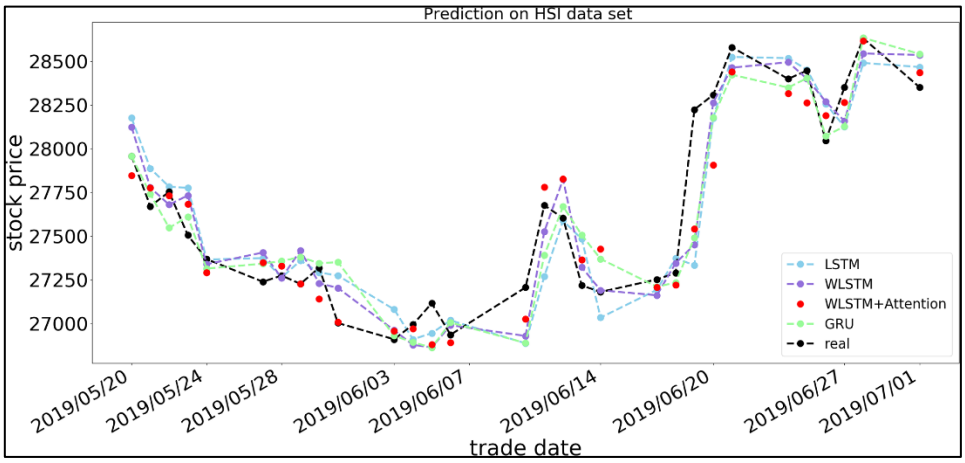


Figure 23 - Qui et al's forecasts for HSI Price. Taken from [4].

Despite the possibility that the problems lay with my own implementation, I believe I may have identified the cause for the vast disparity in our findings. As described in Section 3.2.1.1 of the system design, the majority of LSTM networks require the input data to be reorganised into a three-dimensional matrix prior to being fed into the network. The first dimension consists of the sequences, the second consists of the samples inside each sequence, and the third the number of features for each sample. This method then employs a "sliding window" that traverses the time series data, adding a sequence of multiple samples to the input data for each time step and employing the closing price of the next day's sample as the output value.

As seen in Figure 17, the window is then moved one step forward, and the procedure is repeated. While this is the most prevalent strategy in LSTM implementations, it is still feasible to enter data with only two dimensions, only utilising the previous day's value as the foundation for model prediction. However, a single point does not provide a trend, thus a sliding window is typically employed. In Qui et al's publication however, there is no mention of this sliding window strategy; this, together with the stark discrepancy between our results, leads me to conclude that the authors applied their model structuring their data two-dimensionally.

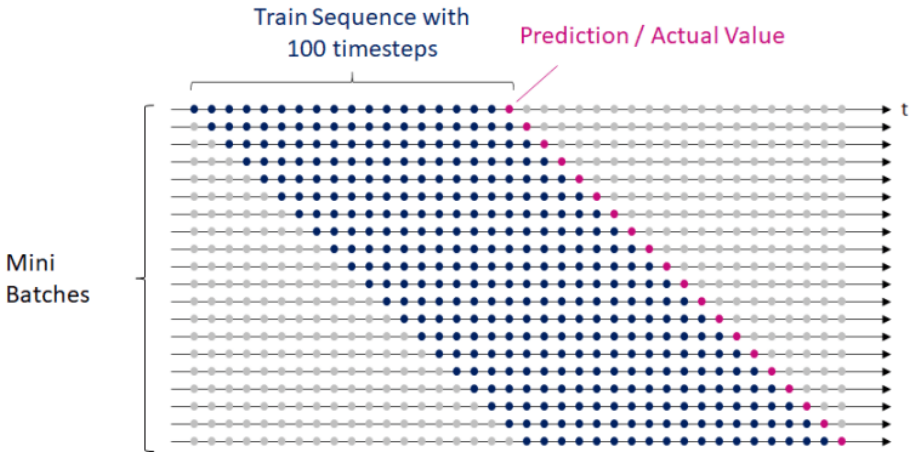


Figure 24 - Example of the 'Sliding Window' approach used for LSTMs. Taken from [72].

To prove this, I constructed a second version of my project using the same wavelet transformation and attention mechanism but did not transform the data into a three-dimensional matrix before inputting it into the model. Table 4.7 displays the results and performance data for this implementation across all datasets utilising a batch size of 50, 100 nodes, and 100 epochs. The model predictions and trading strategy portfolio value over time are plotted in Figure 25, Figure 26, and Figure 27. This version appears to correspond with the findings of Qui et al. far more closely than the prior implementation, as is evident from a simple visual inspection. The projected stock values for the DJIA dataset closely match the actual values, with the model accurately predicting price declines and spikes and only being wrong by a small margin, as evidenced by reduced MSE, RMSE, and MAE values and an R^2 value closer to one. An MDA closer to one also indicates that this model is more adept at forecasting the direction of price movement, and as a consequence, the trading strategy portfolio value increased gradually during the testing time, with the largest profit coming from the HJI dataset resulting in 0.873% profitability or 311.902 USD profit per day. These results, which are closer to those of Qui et al., led me to assume that Qui et al. executed their model by omitting the sliding window technique.

Dataset	MSE	RMSE	MAE	R^2	MDA	Profitability	PPD
S&P500	345.605	18.590	13.580	0.895	0.827	0.765%	264.01
DJIA	4859.602	69.711	55.204	0.984	0.828	0.726%	250.216
HJI	5822.317	76.304	63.945	0.981	0.857	0.873%	311.902

Table 4.7 - Performance Metrics for all datasets when omitting the sliding window technique

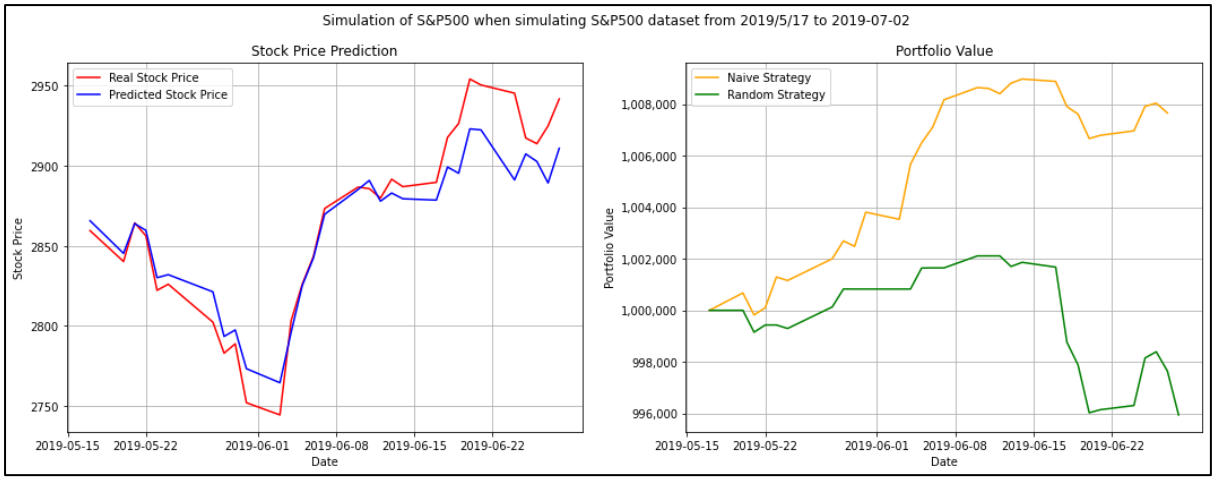


Figure 25 - WLSTM+Attention results for S&P500 dataset when omitting sliding window technique

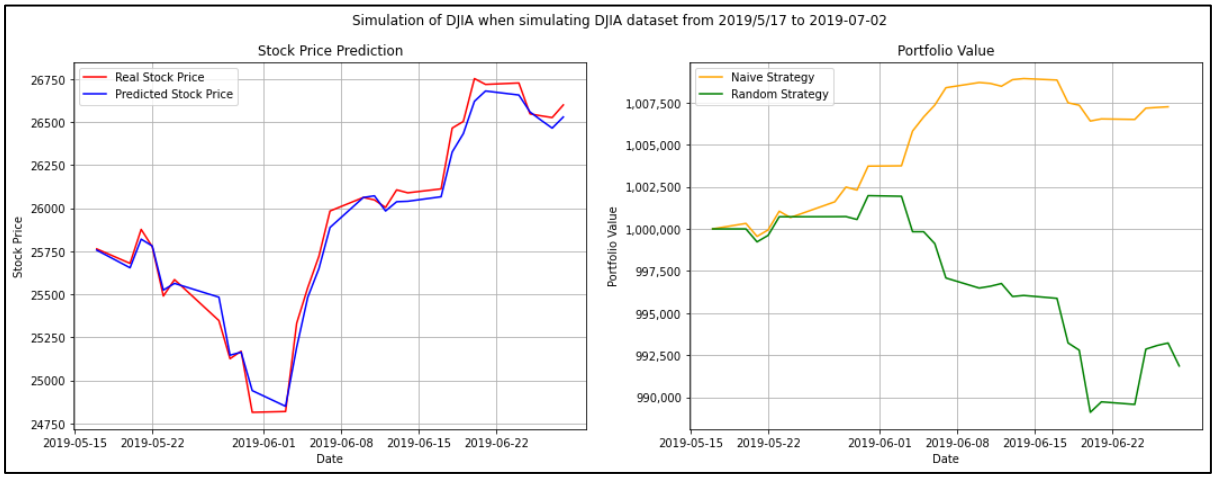


Figure 26 - WLSTM+Attention results for DJIA dataset when omitting sliding window technique

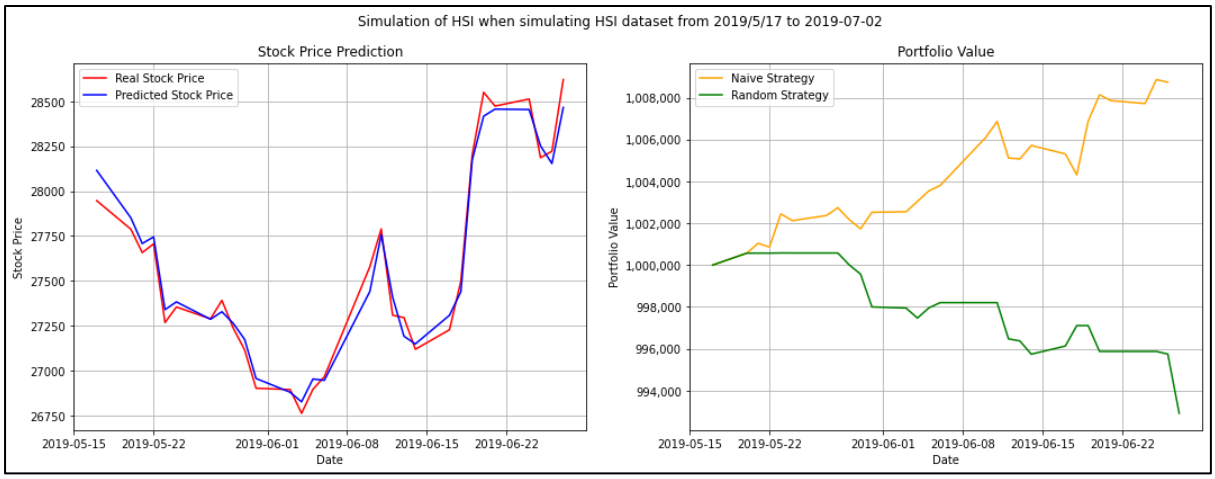


Figure 27 - WLSTM+Attention results for HJI dataset when omitting sliding window technique

Despite this, there is still a significant disparity between the performance metrics obtained by this 2D implementation and those reported by Qui et al., which brings me to another probable flaw I discovered in the author's work. First, while reading the article, I noted that the authors' formulae for RMSE and R^2 values are erroneous; nevertheless, this may be attributed to an honest error made while drafting the report. However, the performance metrics for all three datasets used by the authors are curiously low, given the nature of the data with which they are dealing. Take the authors' MSE value for the DJIA dataset, which is 0.0388 for the implemented WLSTM+Attention model, as an example. Mean squared error is defined as the

average squared difference between observed and projected values, therefore an MSE of 0.0388 indicates that, on average, the model's predictions are within less than one dollar of the actual values, which does not appear to be accurate. To illustrate this, I employed picture editing methods to eliminate the other models from Qui et al's DJIA forecasts in Figure 22 so that the disparities between the WLSTM+Attention predictions and the actual values could be more readily observed in Figure 28 below. It is now evident that although some forecasts are near to the actual values, the majority have quite a substantial discrepancy, with the first sample being around $100 off and the sample for 2019-06-03 being approximately $250 off. I find it improbable that the real average squared difference of these samples would approach 0.0388, and this holds true for all of Qui et al's measurements across all three datasets. I hypothesise that the authors did not reverse normalise their projected values prior to computing the error metrics, resulting in the extremely low error metrics discovered by the authors. However, this leads to further issues, as in data science projects there is no need to scale the target variable for the testing set as the authors would've needed to do to generate these results. Moreover, it appears the authors had no trouble reverse normalising their results prior to plotting their predictions, as Figure 21, Figure 22, and Figure 23 all have the correct scale when plotted.

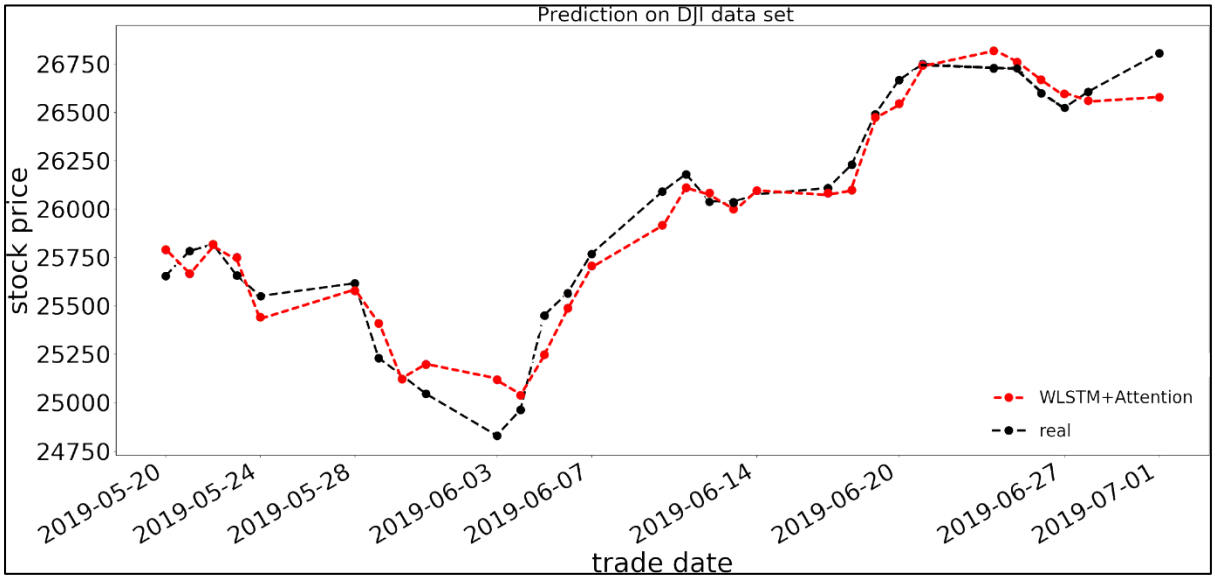


Figure 28 - Qui et al's forecasts for the DJIA dataset. Edited to only show the results of the WLSTM+Attention model and Real Values.

The limited number of testing samples is an additional concern for this work. Qui et al. divided each of their datasets so that only 31 samples were utilised for testing while the remaining samples were used for training. This computes to employing just 0.007203% of the HJI dataset and 0.006320% of the S&P500 and DJIA datasets, an extraordinarily low train/test split compared to the regularly employed values of 20%, 33%, and 50% [73]. The size of the test set influences uncertainty on the test, therefore by employing such a small number of testing samples we are unsure about the reliability of the authors predictions. Using such a small testing set can lead to problems with outliers, overfitting, and sampling bias; hence, we cannot place too much stock in the results provided by Qui et al. or this paper, as the smaller testing dataset may not truly reflect reality. To compensate for this and still provide proof of the usability of LSTM Neural Networks for stock market prediction, the developed model is evaluated on three additional datasets with a significantly greater train-test split in the next section.

4.1.2 Utilising Different Datasets

In this section, the model is tested using other datasets to still provide proof of the usability of the WLSTM+Attention model for stock market forecasting, namely the NASDAQ, DAX, and FTSE 100 datasets, using a much larger train/test split than Qui et al. As before, the subsequent section seeks to determine the optimal parameters for the model and the section that follows presents and evaluates the model's outcomes across all datasets.

4.1.2.1 Parameter Selection

While it is feasible to utilise the same parameters identified in Section 4.1.1.1 for these datasets, I feel the limited testing size hindered the feasibility of those findings as described at the end of Section 4.1.1.2. Therefore, to avoid the same pitfalls as Qui et al, the choice was taken to repeat a limited number of parameter experiments again to identify the optimal parameters while processing the alternative datasets which had a significantly greater number of testing samples. As before, the primary parameters that may be adjusted to increase the accuracy of the model's forecasts are Nodes, Epochs, Batch Size and Window Size. The performance measurements outlined in Section 3.3 are used to evaluate each combination of parameters using the NASDAQ dataset, and the results are shown in Table 4.8 using a colour scale to make it easier to compare the best performing trials. In addition, Appendix B provides charts for all the stated trials.

Experiment	Nodes	Epochs	Batch Size	Window Size	MSE	RMSE	MAE	R2	MDA	Profitability	PPD
1	50	50	5	10	12637.65	112.4173	83.47621	0.991297	0.527363	3.06%	30.47594
2	50	50	5	1	56086.44	236.8258	187.9785	0.961376	0.527363	-1.58%	-15.6864
3	50	50	5	25	15461.83	124.3456	99.82332	0.989352	0.542289	3.11%	30.91867
4	50	50	5	100	235041.8	484.8111	399.9931	0.838137	0.533333	-4.78%	-47.6018
5	50	50	1	10	58745.34	242.3744	191.8419	0.959545	0.535323	-4.10%	-40.7561
6	50	50	10	10	122045.1	349.3495	282.7484	0.915953	0.533333	-3.36%	-33.445
7	100	50	5	10	14403.38	120.0141	89.1713	0.990081	0.522388	1.78%	17.68122
8	100	50	5	25	25055.61	158.2896	121.2281	0.982745	0.529353	-0.56%	-5.58449
9	25	50	5	10	14734.08	121.384	92.60724	0.989853	0.533333	1.00%	9.931004
10	10	50	5	10	209209.6	457.3944	357.1587	0.855926	0.521393	-4.26%	-42.3464

Table 4.8 - Experiment results for WLSTM+Attention testing using NASDAQ dataset.

The experiment began with 50 nodes, 50 epochs, a batch size of 5 and a window size of 10, yielding one of the model's strongest performances. After modifying the window size, it was determined that increasing the window size resulted in usually inferior performance, therefore it was decided to modify the batch size instead. Changes were then made to the number of nodes, however after lower performance measurements it was determined that the initial value of 50 nodes was ideal. Again, the epoch size was not frequently varied because, based on the plotted validation loss, it was determined that the majority of implementations had already converged to a minimum, or because the metrics discovered with 50 epochs were so poor that it did not seem worthwhile to test with different epoch sizes.

Using a window size of 10, 50 hidden layers,50 epochs, and a batch size of 5 yields the greatest performance across all metrics for the provided dataset, as shown in the table. The parameters from experiment 3 could have also been used due to slightly higher profitability, however, experiment 1 performed better across all metrics so its parameters were chosen instead. Figure 29 in the next section illustrates the real price movement with the expected price change for this model, alongside the portfolio value over time of the implemented trading strategy.

4.1.2.2 Results and Discussion

In order to still provide proof of the usability of the WLSTM+Attention model for stock market forecasting the three different stock index datasets are processed using three models, namely

a normal LSTM, an LSTM with wavelet transformation (WLSTM), and the proposed WLSTM+Attention Model. Implementing each model separately will allow the evaluation of the wavelet transformation and attention mechanism on model performance. Each model is trained and evaluated, and the predicted values are plotted in Figure 29, Figure 30, and Figure 31 with the performance metrics listed and compared in Table 4.9, Table 4.10, and Table 4.11.

NASDAQ	MSE	MAE	RMSE	R^2	MDA	Profitability	PPD
LSTM	142754.911	311.295	377.829	0.902	0.526	-5.337	-53.105
WLSTM	43791.658	162.651	209.265	0.970	0.533	1.094	10.887
WLSTM+ Attention	12637.654	112.417	83.476	0.991	0.527	3.06	30.475

Table 4.9 - Performance metrics for NASDAQ Dataset

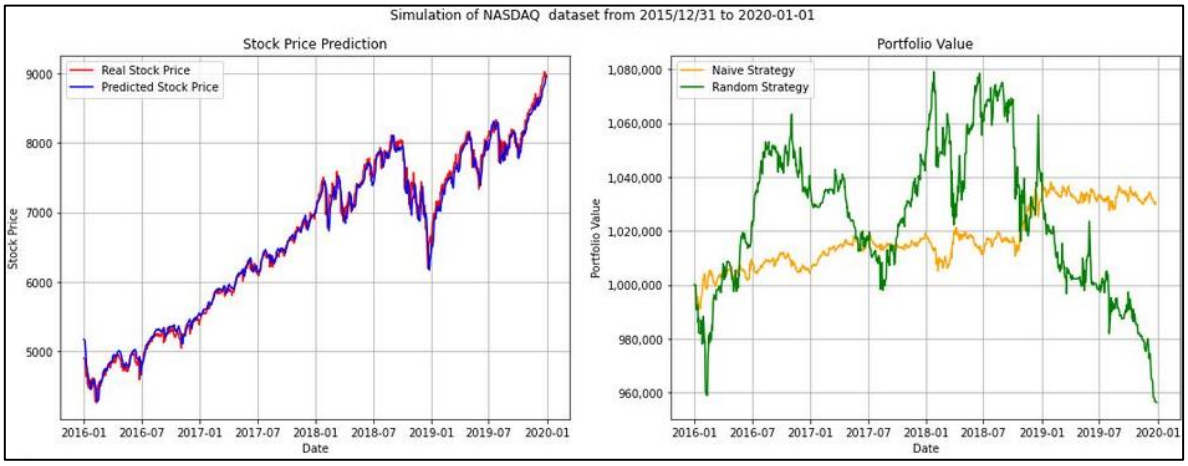


Figure 29 - WLSTM+Attention Results from NASDAQ dataset

DAX	MSE	MAE	RMSE	R^2	MDA	Profitability	PPD
LSTM	359555.346	539.797	599.629	0.697	0.507	-1.113	-10.971
WLSTM	45777.145	175.301	213.956	0.961	0.499	0.971	9.571
WLSTM+ Attention	45494.798	173.161	213.295	0.962	0.504	1.898	18.719

Table 4.10 - Performance metrics for DAX Dataset

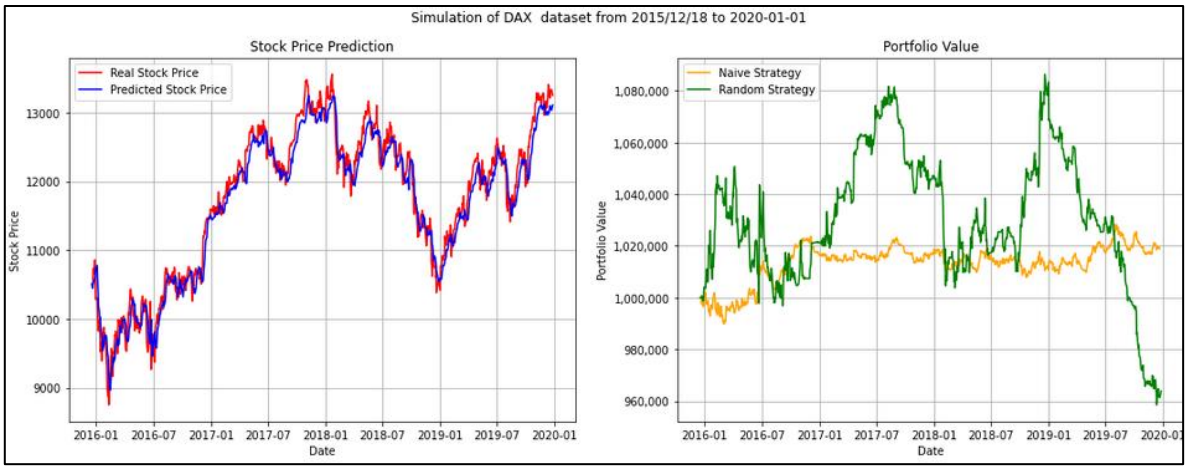


Figure 30 - WLSTM+Attention Results from DAX dataset

FTSE	MSE	MAE	RMSE	R^2	MDA	Profitability	PPD
LSTM	22980.986	132.803	151.594	0.895	0.501	0.222	0.222
WLSTM	16671.739	100.382	129.119	0.924	0.487	0.669	6.626
WLSTM+ Attention	10979.160	78.984	104.781	0.950	0.505	1.677	16.618

Table 4.11 - Performance metrics for FTSE 100 Dataset

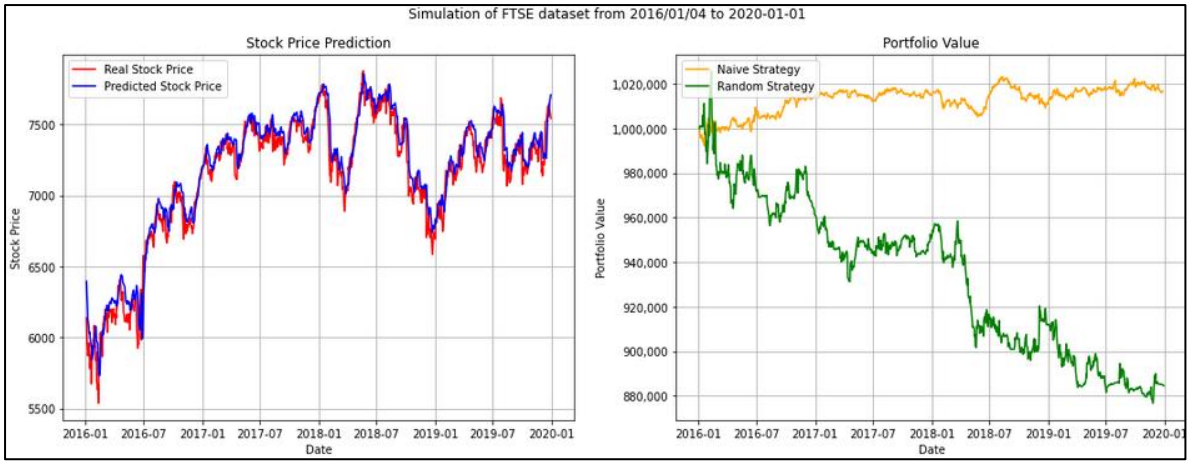


Figure 31 - WLSTM+Attention Results from FTSE 100 dataset

As is evident by a simple visual inspection, each model's predicted values more closely match the actual values than the findings obtained using Qui et al's datasets in Section 4.2.1.2; this is supported by reduced error metrics and R2 values that are closer to one. Importantly, across all datasets, the final WLSTM+Attention model was able to earn a profit, with the portfolio value rising steadily as the trading process progressed. Using the NASDAQ dataset yielded the highest performance, resulting in a profitability of around 3.06%, or approximately 30 USD each day. The conclusion from this is that there is promise for the application of LSTMs, wavelet transformations, and attention mechanisms to assist investors make trading decisions, since the model successfully predicts the following day's price based on past data and generates a profit in the trading simulation. Another key conclusion from these results are the benefits of the wavelet transformation and attention mechanism on model performance, since across all datasets the error metrics were poorest for the simple LSTM model, rose when the wavelet transformation was included, and peaked for the WLSTM+Attention model. The predictions for each model across all three datasets are plotted in Figure 32, Figure 33, and Figure 34 below for comparative purposes. Note how the LSTM model, depicted in blue, generates the weakest predictions, followed by the WLSTM model, depicted in purple, and finally the WLSTM+Attention model, depicted in red.

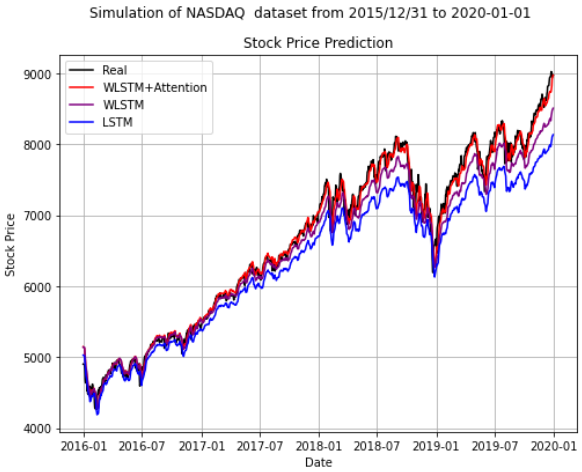


Figure 32 - Model predictions for NASDAQ dataset

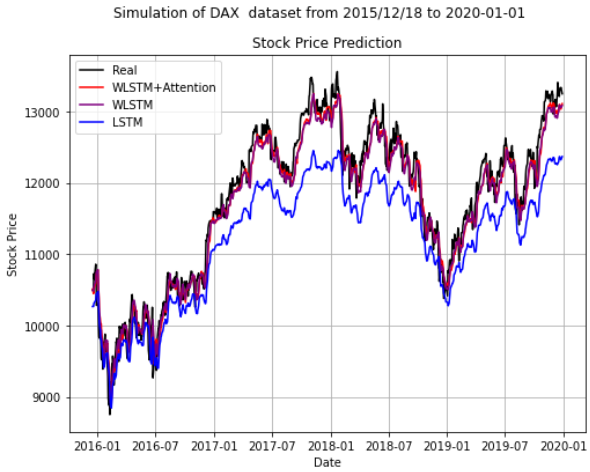


Figure 33 - Model predictions for DAX dataset

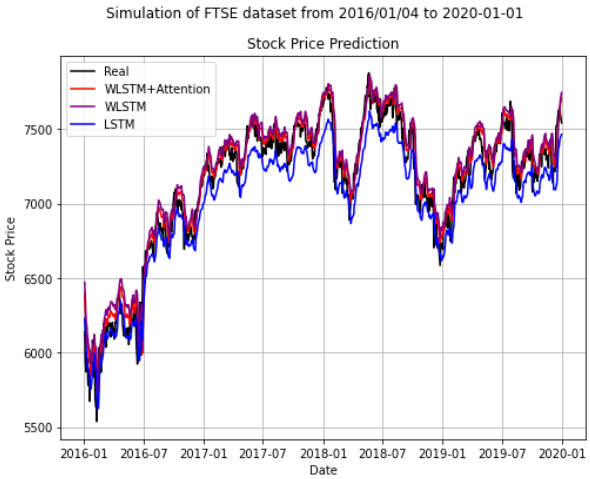


Figure 34 - Model predictions for FTSE 100 dataset

However, influenced by the increase in model performance when omitting the commonly used sliding window technique in Section 4.1.1.2 the decision was made to run all tests again while structuring the data two-dimensionally instead of three-dimensionally. If the model produced more accurate forecasts, this would show that the sliding window approach is not required for financial stock market prediction problems. All three datasets were inputted into this new model, using 50 hidden layers, 50 epochs, and a batch size of 5 based upon the experiments in Section 4.1.2.1. The results are listed in Table 4.12 below and the resultant predictions and portfolio values are plotted in Figure 35, Figure 36, and Figure 37. Evidently, the results are very successful. Across all models the error metrics dropped significantly with R^2 values very close to one, and from a visual inspection it's clear that the model's predicted values closely match the actual values across all three datasets, demonstrating that the sliding window technique is not necessary for financial prediction problems, and the model can accurately predict stock market values of the next day. An MDA value that is closer to one indicates that each model is accurately predicting the direction of change in price, which has a positive impact on the performance of trading strategies across all datasets. The greatest performance coming with the NASDAQ dataset which generated profitability (or return of investment) of 18.517% and PPD of 184.251 USD. The error metrics for the DAX dataset are slightly larger than the NASDAQ and FTSE, however an MDA which matches the other datasets show that although the model may be off in its predictions it still forecasts the direction of change correctly, and ultimately, this in what truly matters in financial predictions and still allows it to remain profitable in the trading simulation. In conclusion, these results show that there is immense potential in the use of the WLSTM+Attention model for stock market forecasting, and by implementing the model predictions into an algorithm investment strategy the outcome

was very profitable, resulting in average profitability of 16.608% across three different datasets.

Dataset	MSE	MAE	RMSE	R^2	MDA	Profitability	PPD
NASDAQ	1442.275	27.344	37.977	0.999	0.814	18.517%	184.251
DAX	4947.923	59.43	70.342	0.996	0.812	16.865%	166.319
FTSE	821.574	23.581	28.663	0.996	0.826	14.442%	143.135

Table 4.12 - WLSTM+Attention results when omitting sliding window technique

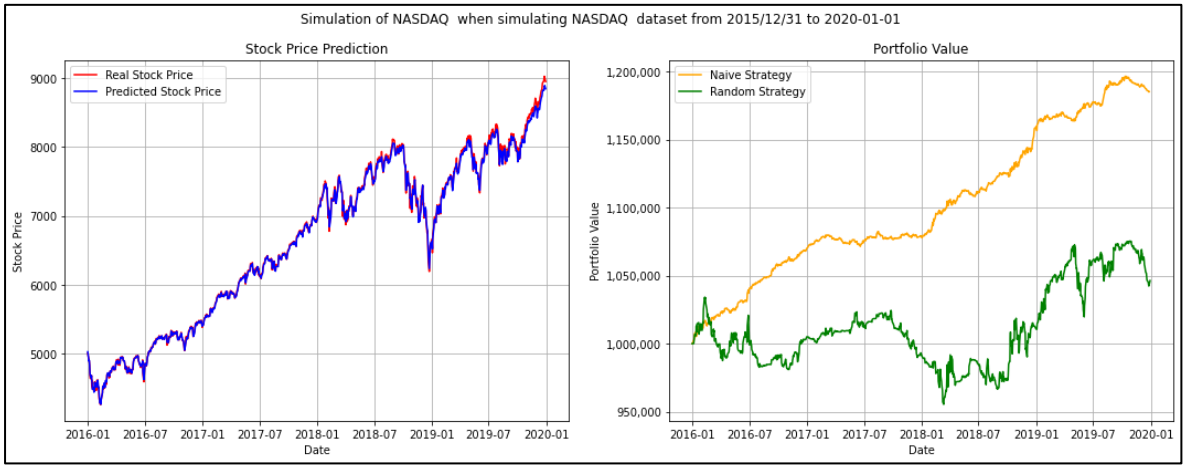


Figure 35 - WLSTM+Attention results for NASDAQ dataset when omitting sliding window technique

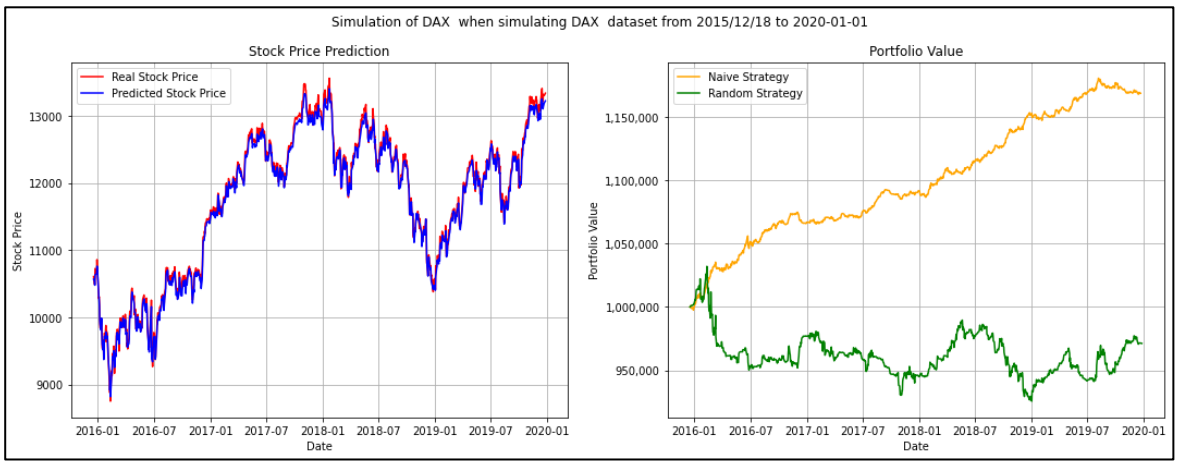


Figure 36 - WLSTM+Attention results for DAX dataset when omitting sliding window technique

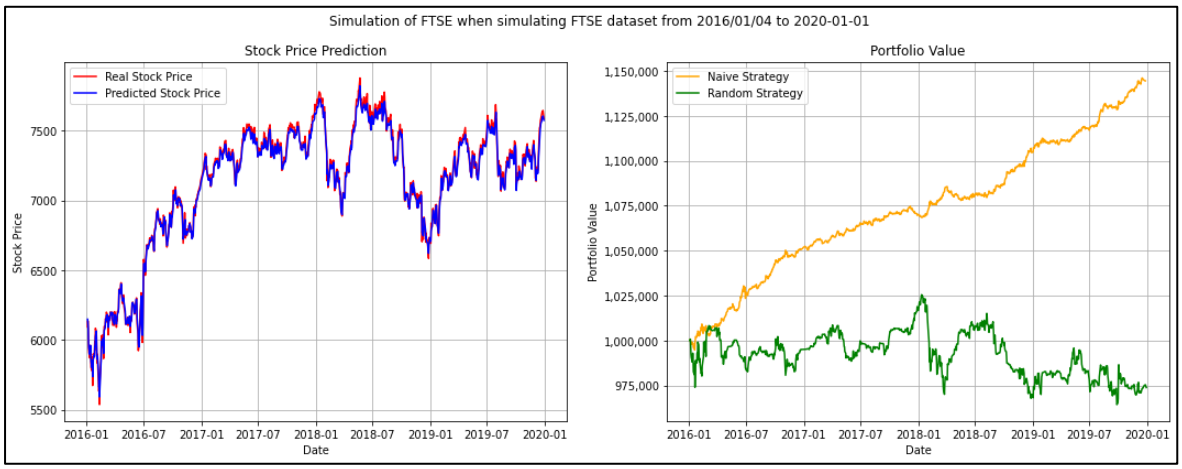


Figure 37 - WLSTM+Attention results for FTSE 100 dataset when omitting sliding window technique

4.2 Critical Appraisal

One of the primary objectives of this paper was to analyse the network's performance using the same stock market data as Qui et al. in an effort to duplicate their findings. The system was initially unable to replicate the results given by the authors after attempting to employ the same parameters as the authors and conducting considerable testing to discover a more suitable combination. However, it was presumed through the implementation of LSTM networks that the authors were organising their dataset in a two-dimensional manner rather than using the standard sliding window method. This was demonstrated by repeating the experiments using only two dimensions which resulted in more accurate forecasts, a profit in the trading simulation, and graphs which visually resembled those of Qui et al. Despite this, the error metrics for this implementation were still significantly different from those of the authors, and it was found by looking at relevant research in the field that the authors' numbers were impossibly low. It was then hypothesised that Qui et al. did not reverse-normalize their projected values before computing their error values, resulting in extremely small and misleading performance measures. A final concern with the work was the incredibly small amount of testing samples, with just 0.006909% of the whole dataset used for training on average. This can lead to a number of problems with outliers, overfitting, and sampling bias, which reduces the credibility of the conclusions reported by Qui et al. and the previous experiments since the smaller testing dataset may not accurately represent reality.

To compensate for this and still demonstrate the viability of LSTM Neural Networks for stock market prediction, the developed WLSTM+Attention model was then tested on three different datasets utilising a significantly bigger train/test split of 80%. When structuring the data three-dimensionally the model manged to yield slightly accurate predictions as indicated by improved MSE, RMSE, MAE, and R^2 performance measures. However, influenced by the increase in model performance when omitting the commonly used sliding window technique for Qui et al's datasets the decision was made to run all tests again while structuring the data two-dimensionally instead. These results were extremely successful, reducing all error metrics significantly and closely matching the actual values, as is evidenced by an average coefficient of determination (R^2) of 0.997 which matches Qui et al's values of 0.94, proving the replication was a success and the advantage of omitting the sliding window technique. This accomplishes the remaining primary objectives of the research and demonstrates the applicability of Deep Learning and LSTMs for financial forecasting and investor decision making. In comparison to other work, this work improves on research such as Bahandari et al. [38], Althelaya et al. [74], and Lu et al. [75] who achieve R^2 values of 0.9974 and 0.993 and 0.9646 respectively, highlighting the advantages of this implementation.

The secondary objective of this work involved implementation of an algorithmic investment strategy based on the model predictions to generate profitability metrics. This was similarly successful, as the final unstructured WLSTM+Attention model was able to generate significant profits based on its predictions across all datasets, demonstrating not only the model's strength but also its potential application in automated trading methods, reaching an average profitability across three datasets of 16.608%. This finding is very encouraging, and outperforms prior research in this sector such as Fazelia & Houghten's return on investment of 6.67% [41] and Fisher & Krauss' daily returns of 0.23% [53], however does fall short of Touzani & Douzi's annualised return of 27.13% [54] which is enhanced by a more complicated trading strategy. With the implementation of a more advanced trading strategy these model predictions could yield greater profitability, and methods with which to improve the trading strategy are discussed later in Section 5.2.

Furthermore, these findings provide more evidence for the application of wavelet transformations and attention mechanisms in LSTMs for stock market forecasting, as by testing each dataset on three distinct models, namely a basic LSTM, an LSTM with a wavelet transformation, and an LSTM with a wavelet transformation and attention mechanism, this

study demonstrated that the application of such techniques resulted in a higher degree of fitting and enhanced prediction accuracy. This follows on from work by Lie et al [47], Xu and Keseli [46], and Gu et al. [48] who all reported lower error metrics after implementation of an attention mechanism in their LSTM models, as well as the work of Skehin et al. [50] and Pesio and Honchar [49] who discovered an improvement in model performance following the application of a wavelet transformation to their datasets.

5. Conclusions

5.1 Summary

This paper develops a machine learning framework to anticipate stock market values based on the work of Qui et al. using three primary techniques. The first is a Long-Short Term Memory neural network with a specialised memory cell that eliminates the issue of long-term dependencies for time sequence data that is typical of other neural network approaches. The second strategy is an attention mechanism that allows the LSTM to selectively focus on particular input features by assigning variable weights to each input, and the third technique is a wavelet transformation to denoise the volatile historical financial data. All of the financial data utilised in this study are historical data from major stock indexes obtained from the online, open-source Yahoo! Finance; however, any other stock indices employing open-high-low-closing prices may also be used. Although one of the primary objectives of this paper was to replicate the findings of Qui et al., it was determined via extensive testing that this was not possible due to flaws in the authors' work that weaken the validity of their conclusions. In order to demonstrate the applicability of LSTM Neural Networks for stock market forecasting, the developed model was assessed on three more datasets with a substantially larger train-test split, which produced significantly stronger results achieving an average coefficient of determination of 0.997. The model's forecasts were then incorporated into an algorithmic investing strategy to create profitability measurements which supported the model's projections by reaching an average profitability of 16.608%, which is highly competitive with existing research. The findings of this study also offered additional support for the use of wavelet transformations and attention mechanisms in LSTMs for stock market forecasting, with the implemented WLSTM+Attention model outperforming the LSTM and WLSTM models on all datasets evaluated. This paper also argues for the use of different error metrics such as Mean Directional Accuracy to evaluate model performance in stock market prediction problems and provides evidence of the benefits of omitting the commonly used sliding window technique for forecasting financial data.

5.2 Suggestions for Future Work

Despite the success of this project, there are still areas that might be expanded upon in future efforts. Firstly, the implemented trading strategy is relatively naive; therefore, additional research could be conducted to develop a more realistic simulation, possibly including estimated brokerage fees, the effects of stock purchases on the current market, and filters so that stronger predictions result in a greater quantity of stock being bought or shorted with the goal of making the simulation even more realistic. During my research I also observed how every study that utilises a trading strategy implements their own different version which makes it difficult to compare the generated profitability metrics because they are dependent on the strength of the trading strategy utilised. Therefore, I would advocate a future paper that presents numerous standardised trading strategies and evaluation metrics for use with Machine Learning stock price forecasting. By standardising the trading strategies used researchers will be able to evaluate the success of their model in a realistic simulation and acquire a greater insight into how their implanted model compares to the work performed by other researchers using the same trading strategy. I also believe that the accuracy of the constructed LSTM model might be improved by including additional features from the dataset. Work like Lanbouri & Achchab utilise a second dataset containing 75 financial annual ratios for companies such as Earing Per Share (EPS), Return on Equity (ROE) and Price Earing Ratio (PER) [76] while Florian Müller uses the values from Yahoo! Finance to calculate extra statistical indicators such as Relative Strength Index (RSI), Simple Moving Averages (SMA) and Exponential Moving Averages (EMA) [72], all which provide the model more information about the stock with which to base its predictions on. Additional features alongside the aforementioned technical indicators could be macroeconomic indicators such as the level of

inflation, central bank interest rates or strength of the nation’s currency measured by exchange rates. Recent research has also focused on sentiment analysis, with the addition of predictions related to stock-related news and Twitter analysis allowing the model to generate stable predictions during major events such as value spikes or recessions, which could both potentially be implemented in this project to improve the model's forecasts. Additionally, the majority of stock trading employs a buy and hold strategy, which entails holding onto stocks for a long time despite market volatility.As a result, future studies should involve forecasting stock market values for a wide range of times, such as 10, 30, 100, or even a year in the future, as opposed to simply the day after.

Bibliography

[1] M. Lyck, "Why 80% of Day Traders Lose Money," 18 6 2020. [Online]. Available: https://marklyck.medium.com/why-80-of-day-traders-lose-money-78d51b10fe25. [Accessed 27 7 2022].

[2] CNN Money, "NASAA report says 70 percent of traders will lose nearly all their money," 9 8 1999. [Online]. Available: https://money.cnn.com/1999/08/09/markets/daytrade/. [Accessed 22 7 2022].

[3] H. Bessembinder, "Do Stocks Outperform Treasury Bills?," Journal of Financial Economics (JFE), Forthcoming, 2017.

[4] J. Qiu, B. Wang and C. Zhou, "Forecasting stock prices with long-short term memory neural network based on attention mechanism," *PLOS ONE,* vol. 15, no. 1, pp. 1-15, 3 1 2020.

[5] P. A. Sánchez-Sánchez, J. R. García-González and L. H. P. Coronell, "Encountered Problems of Time Series with Neural Networks: Models and Architectures," *Recent Trends in Artificial Neural Networks - from Training to Prediction,* 27 11 2019.

[6] A. Tondak, "Recurrent Neural Networks," K21Academy, 4 1 2021. [Online]. Available: https://k21academy.com/datascience/machine-learning/recurrent-neural-networks/. [Accessed 8 6 2022].

[7] D. Rumelhart, G. Hinton and R. Williams, "Learning representations by back-propagating errors," *Nature,* vol. 323, p. 533–536, 1986.

[8] Infolks Pvt Ltd., "Recurrent Neural Network and Long Term Dependencies," 14 7 2019. [Online]. Available: https://medium.com/tech-break/recurrent-neural-network-and-long-term-dependencies-e21773defd92. [Accessed 8 6 2022].

[9] D. Datta, P. E. David, D. Mittal and A. Jain, "Neural Machine Translation using Recurrent Neural Network," *International Journal of Engineering and Advanced Technology,* vol. 9, no. 4, pp. 1395-1400, 2020.

[10] A. Graves, A.-r. Mohamedl and G. Hinton, "Speech recognition with deep recurrent neural networks," *IEEE International Conference on Acoustics, Speech and Signal Processing,* pp. 6645-6649, 2013.

[11] A. Hajdarevic, L. Banjanovic-Mehmedovic, I. Dzananovic, F. Mehmedovic and M. Ayaz Ahmad, "Recurent Neural Network as a Tool for Parameter Anomaly Detection in Thermal Power Plant," *International Journal of Scientific & Engineering Research,,* vol. 6, no. 8, pp. 448-455, 2015.

[12] S. Hochreiter, "The Vanishing Gradient Problem During Learning Recurrent Neural Nets and Problem Solutions," *International Journal of Uncertainty Fuzziness and Knowledge-Based Systems,* vol. 6, no. 2, pp. 107-116, 1998.

[13] IBM, "What are REcurrent Neural Networks?," 14 9 2020. [Online]. Available: https://www.ibm.com/cloud/learn/recurrent-neural-networks. [Accessed 8 6 2022].

[14] C. Olah, "Understanding LSTM Networks," 27 8 2015. [Online]. Available: https://colah.github.io/posts/2015-08-Understanding-LSTMs/. [Accessed 8 6 2022].

[15] S. Hochreiter and J. Schmidhuber, "Long Short-term Memory," *Neural Computation,* vol. 9, no. 8, pp. 1735-1780, 12 1997.

[16] A. Graves, N. Jaitly and A.-r. Mohamed, "Hybrid speech recognition with Deep Bidirectional LSTM," *IEEE Workshop on Automatic Speech Recognition and Understanding,* pp. 273-278, 2013.

[17] A. Rao and N. Spasojevic, "Actionable and Political Text Classification using Word Embeddings and LSTM," 2016.

[18] Y. Bengio, K. H. Cho and D. Bahdanau, "Neural Machine Translation by jointly learning to align and translate," 1 9 2014.

[19] Z. Yang, Z. Hu, Y. Deng, C. Dyer and A. Smola, "Neural Machine Translation with Recurrent Attention Modeling," in *Proceedings of the First Conference on Machine Translation*, 2016.

[20] Y. Bin, Y. Yang, F. Shen, N. Xie, H. T. Shen and X. Li, "Describing Video With Attention-Based Bidirectional LSTM," *IEEE transactions on cybernetics,* vol. 49, no. 7, pp. 2631-2641, 25 5 2018.

[21] Y. wu, Z. Liu, W. Xu, J. Feng, S. Palaiahnakote and T. Lu, "Context-Aware Attention LSTM Network for Flood Prediction," in *24th International Conference on Pattern Recognition (ICPR)*, Beijing, 2018.

[22] Conceptually, "Signal and Noise," [Online]. Available: https://conceptually.org/concepts/signal-and-noise. [Accessed 8 6 2022].

[23] L. Fan, F. Zhang, H. Fan and C. Zhang, "Brief review of image denoising techniques," *Visual Computing for Industry, Biomedicine, and Art,* vol. 2, no. 7, 8 7 2019.

[24] Q. Tang, R. Shi, T. Fan, Y. Ma and J. Huang, "Prediction of Financial Time Series Based on LSTM Using Wavelet Transform and Singular Spectrum Analysis," *Mathematical Problems in Engineering,* 9 6 2021.

[25] E. W.Sun and T. Mein, "A new wavelet-based denoising algorithm for high-frequency financial data mining," *European Journal of Operational Research,* vol. 217, no. 3, pp. 589-599, 2012.

[26] J. Kuepper, "Day Trading Essentials Guide," Investopedia, 14 3 2022. [Online]. Available: https://www.investopedia.com/articles/trading/05/011705.asp. [Accessed 28 7 2022].

[27] E. Norris, "Scalping: Small Quick Profits Can Add Up," Investopedia, 7 2 2022. [Online]. Available: https://www.investopedia.com/articles/trading/05/scalping.asp. [Accessed 28 7 2022].

[28] "Social Trading," The Lazy Trader, [Online]. Available: https://thelazytrader.com/social-trading/. [Accessed 28 7 2022].

[29] P. K. Illa, B. Parvathala and A. K. Sharma, "Stock price prediction methodology using random forest algorithm and support vector machine," *International Conference on Applied Research and Engineering,* vol. 56, no. 4, pp. 1776-1782, 2021.

[30] S. Rath, B. K. Gupta and A. K. Nayak, "Stock Market Prediction Using Supervised Machine Learning Algorithm," *Advances in Distributed Computing and Machine Learning,* vol. 302, pp. 374-381, 2022.

[31] M. U. Ghania, M. Awaisa and M. Muzammula, "Stock Market Prediction Using Machine Learning (ML) Algorithms," *Advances in Distributed Computing and Artificial Intelligence,* vol. 8, no. 4, pp. 97-116, 2019.

[32] C. Krauss, X. A. Do and N. Huck, "Deep neural networks, gradient-boosted trees, random forests: Statistical arbitrage on the S&P 500," *European Journal of Operational Research,* vol. 259, no. 2, pp. 689-702, 2017.

[33] X. Zhong and D. Enke, "Predicting the daily return direction of the stock market using hybrid machine learning algorithms," *Financial Innovation,* vol. 5, no. 24, pp. 5-24, 2019.

[34] J. M.-T. Wu, Z. Li, G. Srivastava, M.-H. Tasi and J. C.-W. Lin, "A graph-based convolutional neural network stock price prediction with leading indicators," *Semantic eSystems: Engineering methods, techniques, and tools,* vol. 51, no. 3, pp. 628-644, 2021.

[35] T. Fischer and C. Krauss, "Deep learning with long short-term memory networks for financial market predictions," *European Journal of Operational Research,* vol. 270, no. 2, pp. 654-669, 2018.

[36] M. Nabipour, P. Nayyeri, H. Jabani, A. Mosavi, E. Salwana and S. S., "Deep Learning for Stock Market Prediction," *Entropy,* vol. 22, no. 8, pp. 840-863, 2020.

[37] V. CK, "Applying Machine Learning Models in Stock Market Prediction," *EPRA International Journal of Research and Development* , vol. 5, no. 4, pp. 395-398, 2020.

[38] H. N. Bhandari, B. Rimalb and N. R. Pokhrelc, "Prediction Stock Price Based on Different Index Factors Using LSTM," *International Conference on Machine Learning and Cybernetics,* 2019.

[39] M. Nikou, G. Mansourfar and J. Bagherzadeh, "Stock price prediction using DEEP learning algorithm and its comparison with machine learning algorithms," *Intelligent Systems in Accounting, Finance and Management,* vol. 26, no. 4, pp. 164-174, 2019.

[40] M. Rana, M. M. Uddin and M. M. Hoque, "Effects of Activation Functions and Optimizers on Stock Price Prediction using LSTM Recurrent Networks," *Proceedings of the 2019 3rd International Conference on Computer Science and Artificial,* vol. 3, pp. 354-358, 2019.

[41] A. Fazeli and S. Houghten, "Deep Learning for the Prediction of Stock Market Trends," *2019 IEEE International Conference,* 2019.

[42] J. Long, Z. Chen, W. He, T. Wu and J. Rena, "An integrated framework of deep learning and knowledge graph for prediction of stock price trend: An application in Chinese stock exchange market," *Applied Soft Computing,* vol. 91, p. 106205, 2020.

[43] J. Li, H. Bu and J. Wu, "Sentiment-aware stock market prediction: A deep learning method," *2017 International Conference on Service Systems and Service Management,* vol. 14, 2017.

[44] H. Chung and K.-s. Shin, "Genetic Algorithm-Optimized Long Short-Term Memory Network for Stock Market Prediction," *Sustainability,* vol. 10, no. 3765, 2018.

[45] K. Zhanga, G. Zhong, J. Donga, S. Wang and Y. Wang, "Stock Market Prediction Based on Generative Adversarial Network," *Procedia Computer Science,* vol. 147, pp. 400-406, 2019.

[46] Y. Xu and V. Keselj, "Stock Prediction using Deep Learning and Sentiment Analysis.," *IEEE International Conference on Big Data (Big Data),* pp. 5573-5580, 2019.

[47] H. Li, Y. Shen and Y. Zhu, "Stock Price Prediction Using Attention-based Multi-Input LSTM," *Proceedings of Machine Learning Research,* vol. 95, pp. 454-469, 2018.

[48] Y. H. Gu, D. J. H. Yin, R. Zheng, X. Piao and S. J. Yoo, "Forecasting Agricultural Commodity Prices Using Dual Input Attention LSTM," *Agriculture,* vol. 12, no. 256, 2022.

[49] L. D. Persio and O. Honchar, "Artificial Neural Networks architectures for stock price prediction:," *INTERNATIONAL JOURNAL OF CIRCUITS, SYSTEMS AND SIGNAL PROCESSING,* vol. 10, pp. 403-413, 2016.

[50] T. Skehin, M. Crane and M. Bezbradica, "Day Ahead Forecasting of FAANG Stocks Using ARIMA, LSTM Networks and Wavelets.," 5 December 2018. [Online]. Available: https://www.semanticscholar.org/paper/Day-ahead-forecasting-of-FAANG-stocks-using-ARIMA%2C-Skehin-Crane/f1d7c9bdd403af65fa38013b0f160ba6a12c8e89. [Accessed 11 6 2022].

[51] Z. Li and T. Vincent, "Combining the real-time wavelet denoising and long-short-term-memory neural network for predicting stock indexes," *IEEE Symposium Series on Computational Intelligence (SSCI),* pp. 1-8, 2017.

[52] D. Jothimani, R. Shankar and S. S. Yadav, "Discrete Wavelet Transform-Based Prediction of Stock Index: A Study on National Stock Exchange Fifty Index," Indian Institute of Technology Delhi, India, Delhi, 2015.

[53] T. Fischer and C. Krauss, "Deep learning with long short-term memory networks for financial market predictions," *European Journal of Operational Research,* vol. 270, no. 2, pp. 654-669, 2018.

[54] Y. Touzani and K. Douzi, "An LSTM and GRU based trading strategy adapted to the Moroccan market," *Journal of Big Data volume,* vol. 8, no. 126, 2021.

[55] J. Michanków, P. Sakowski and R. Slepaczuk, "LSTM in Algorithmic Investment Strategies on BTC and," *Sensors,* vol. 917, p. 22, 2022.

[56] L. Troiano, E. M. Villa and V. Loia, "Replicating a Trading Strategy by Means of LSTM for Financial Industry Applications," *IEEE Transactions on Industrial Informatics,* vol. 14, no. 7, pp. 3226 - 3234, 2018.

[57] Google, "Welcome to Colab," [Online]. Available: https://colab.research.google.com/?utm_source=scs-index. [Accessed 15 7 2022].

[58] G. V. Rossum, "Python tutorial," Amsterdam, 1995.

[59] Jupyter, "Project Jupyter | Home," [Online]. Available: https://jupyter.org/. [Accessed 15 7 2022].

[60] S. V. D. Walt, S. C. Colbert and G. Varoquaux, "The NumPy array: a structure for efficient numerical computation," *Computing in Science and Engineering,* vol. 13, no. 2, pp. 23-30, 2011.

[61] W. McKinney, "Data Structures for Statistical Computing in Python," *Python in Science,* vol. 9, pp. 56-61, 2010.

[62] Plotly Technologies Inc., "Collaborative data science," 2015. [Online]. Available: https://plot.ly. [Accessed 19 07 2022].

[63] seaborn, "seaborn: statistical data visualization," [Online]. Available: https://seaborn.pydata.org/. [Accessed 2022 8 01].

[64] P. Barrett, J. Hunter and J. T. Miller, "Matplotlib – A Portable Python Plotting Package," *Astronomical Data Analysis Software and Systems,* vol. 347, no. 14, pp. 91-95, 2005.

[65] yfinance, "yfinance: PyPI," [Online]. Available: https://pypi.org/project/yfinance/. [Accessed 2022 8 1].

[66] G. R. Lee, R. Gommers, F. Waselewski, K. Wohlfahrt and A. O'Leary, "PyWavelets: A Python package for wavelet analysis," *The Journal of Open Source Software,* vol. 4, no. 36, p. 1237, 2018.

[67] F. Chollet, "Keras," 2015. [Online]. Available: https://keras.io. [Accessed 15 7 2022].

[68] P. Rémy, "Attention Mechanism Implementation for Keras," 25 4 2022. [Online]. Available: https://github.com/philipperemy/keras-attention-mechanism. [Accessed 15 7 2022].

[69] Yahoo Finance, "Yahoo Finance - stock market live, quotes, business and finance news," [Online]. Available: https://uk.finance.yahoo.com/. [Accessed 16 7 2022].

[70] S. Doshi, "Various Optimization Algorithms For Training Neural Network," Towards Data Science, 13 1 2019. [Online]. Available: https://towardsdatascience.com/optimizers-for-training-neural-network-59450d71caf6. [Accessed 16 7 2022].

[71] A. Gupta, "A Comprehensive Guide on Deep Learning Optimizers," Analytics Vidhya, 7 10 2021. [Online]. Available: https://www.analyticsvidhya.com/blog/2021/10/a-comprehensive-guide-on-deep-learning-optimizers/. [Accessed 16 7 2022].

[72] F. Muller, "Feature Engineering for Multivariate Stock Market Prediction with Python," relataly.com, 29 6 2020. [Online]. Available: https://www.relataly.com/feature-engineering-for-multivariate-time-series-models-with-python/1813/#h-selected-statistical-indicators. [Accessed 6 8 2022].

[73] J. Brownlee, "Train-Test Split for Evaluating Machine Learning Algorithms," Machine Learning Mastery, 26 8 2020. [Online]. Available: https://machinelearningmastery.com/train-test-split-for-evaluating-machine-learning-algorithms/. [Accessed 8 8 2022].

[74] K. A. Althelaya, E.-S. M. El-Alfy and S. Mohammed, "Evaluation of bidirectional LSTM for short-and long-term stock market prediction," *International Conference on Information and Communication Systems (ICICS),* 2018.

[75] W. Lu, J. Li, Y. Li, A. Sun and J. Wang, "A CNN-LSTM-Based Model to Forecast Stock Prices," *Artificial Intelligence for Smart System Simulation,* 2020.

[76] Z. Lanbouri and S. Achchab, "A new approach for Trading based on Long-Short Term memory technique," International Journal of Computer Science Issues, 2019.

[77] H. M, G. E.A., V. K. Menon and S. K.P., "NSE Stock Market Prediction Using Deep-Learning Models," *Procedia Computer Science,* vol. 132, pp. 1351-1362, 2018.

[78] P. B. S and M. S. P. M., "Stock Price Prediction Using LSTM," *Test Engineering and Management,* vol. 83, pp. 5246-5251, 2020.

[79] A. Moghar and M. Hamiche, "Stock Market Prediction Using LSTM Recurrent Neural Network," *Procedia Computer Science,* vol. 170, p. 1168–1173, 2020.

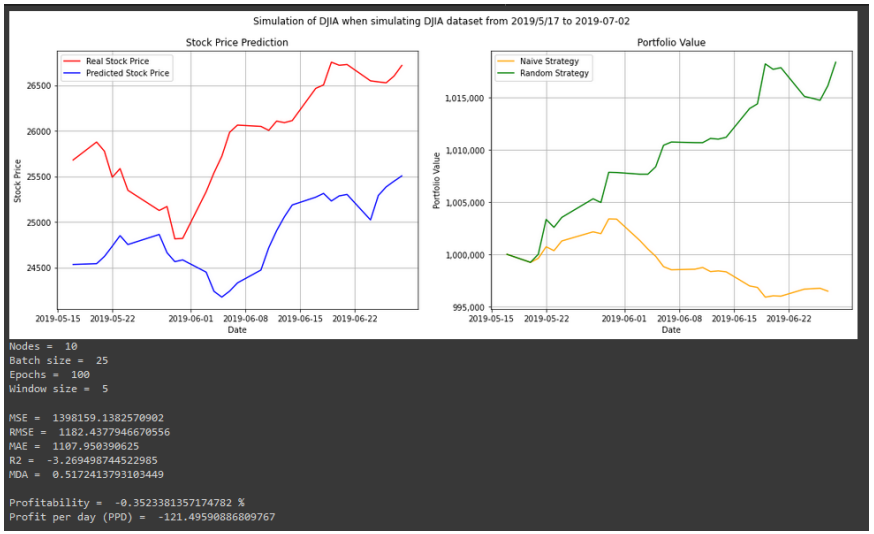
[80] A. C. J. Malar, M. D. Priya, M. K. Kumar, S. M. Arunsankar, K. V. Bilal and S. Karthik, "Deep Learning-based Stock Market Prediction," *Lecture Notes in Networks and Systems,* vol. 341, p. 709–716, 2022.

[81] S. Romero, "The Rise of Trading Apps and its Impact on Real Estate," 3 8 2021. [Online]. Available: https://www.cushmanwakefield.com/en/insights/the-edge/the-rise-of-trading-apps. [Accessed 27 7 2022].

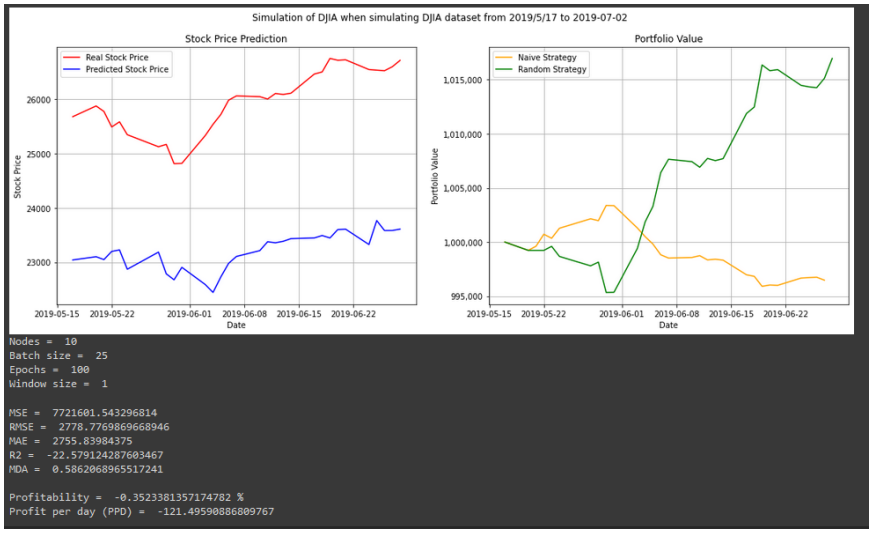
Appendix

A - 4.2.1.1 Parameter Experiment Results

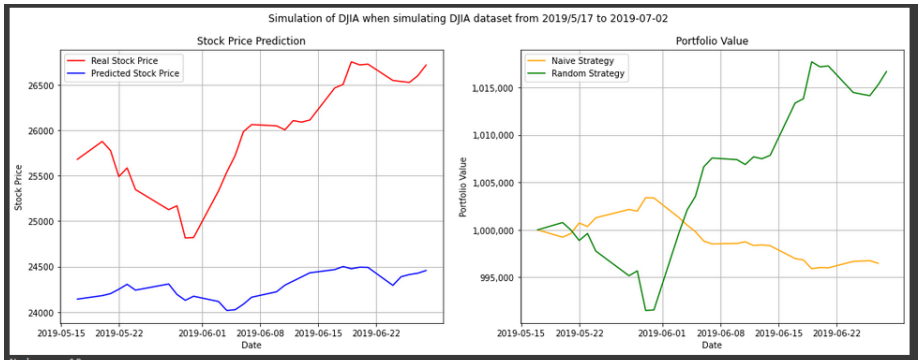
This appendix illustrates the experimental plots described in Section 4.1.1.1. The left figure depicts the difference between the expected (blue) and actual (red) values, while the right plot depicts the difference between the naive (orange) and random (green) trading strategies. The computed performance metrics are provided under the graphs.



Comparison to Qui et al - Experiment 1

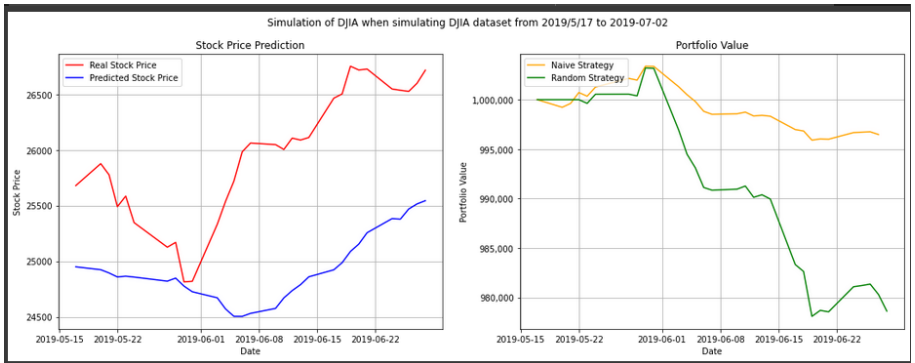


Comparison to Qui et al - Experiment 2



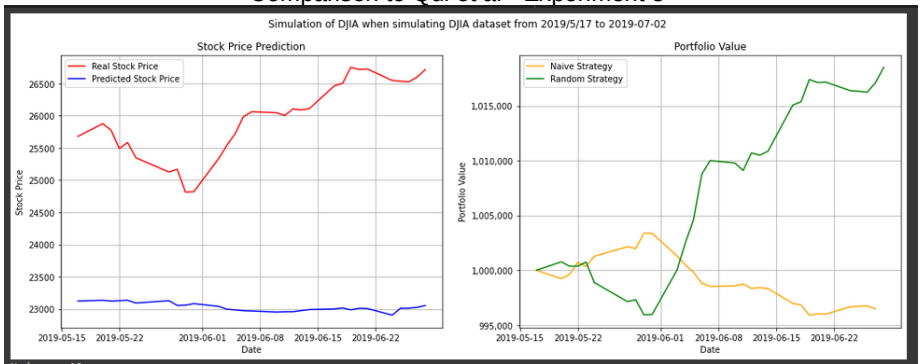

```
Nodes =  10
Batch size =  25
Epochs =  100
Window size =  50

MSE =  3024063.083246104
RMSE =  1738.9833476046008
MAE =  1671.9377604166666
R2 =  -8.23445206199658
MDA =  0.5172413793103449

Profitability =  -0.3523381357174782 %
Profit per day (PPD) =  -121.49590886809767
```

Comparison to Qui et al - Experiment 3



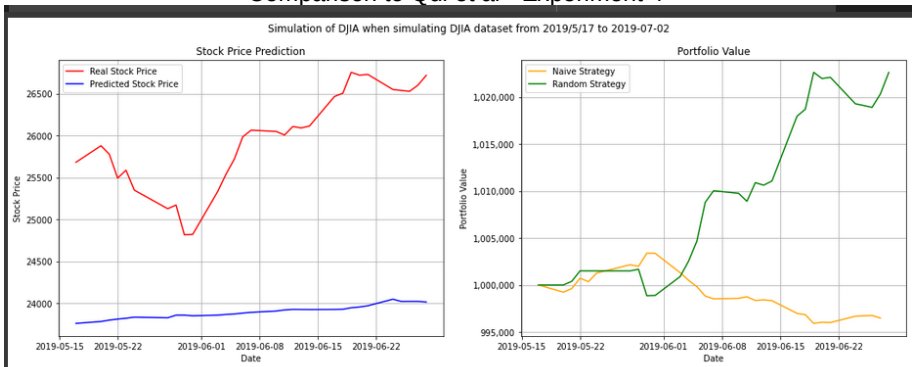

```
Nodes =  10
Batch size =  25
Epochs =  100
Window size =  10

MSE =  1282720.2162855784
RMSE =  1132.5723889825226
MAE =  1038.1857421875
R2 =  -2.9169878472006308
MDA =  0.5862068965517241

Profitability =  -0.3523381357174782 %
Profit per day (PPD) =  -121.49590886809767
```

Comparison to Qui et al - Experiment 4



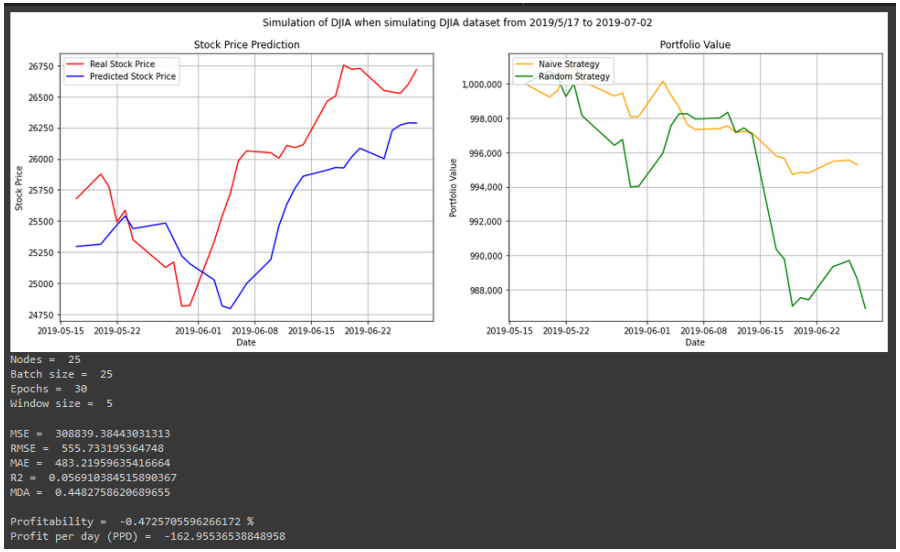

```
Nodes =  10
Batch size =  25
Epochs =  100
Window size =  25

MSE =  8960658.68209877
RMSE =  2993.4359325194805
MAE =  2931.1275390625
R2 =  -26.362780063083907
MDA =  0.4482758620689655

Profitability =  -0.3523381357174782 %
Profit per day (PPD) =  -121.49590886809767
```

Comparison to Qui et al - Experiment 5

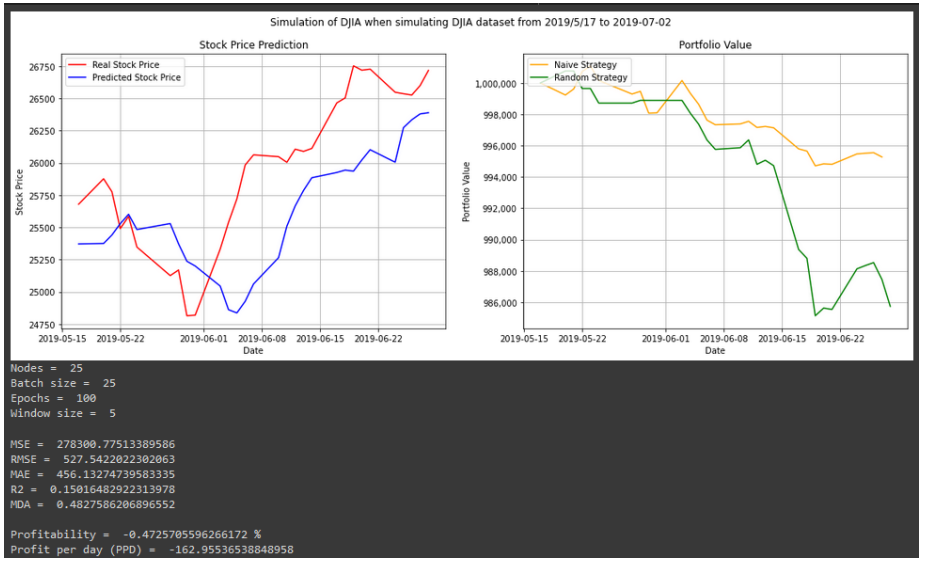



```
Nodes =  5
Batch size =  50
Epochs =  100
Window size =  100

MSE =  4511232.048620606
RMSE =  2123.9661128701196
MAE =  2060.110546875
R2 =  -12.775756307574134
MDA =  0.5517241379310345

Profitability =  -0.3523381357174782 %
Profit per day (PPD) =  -121.49590886809767
```

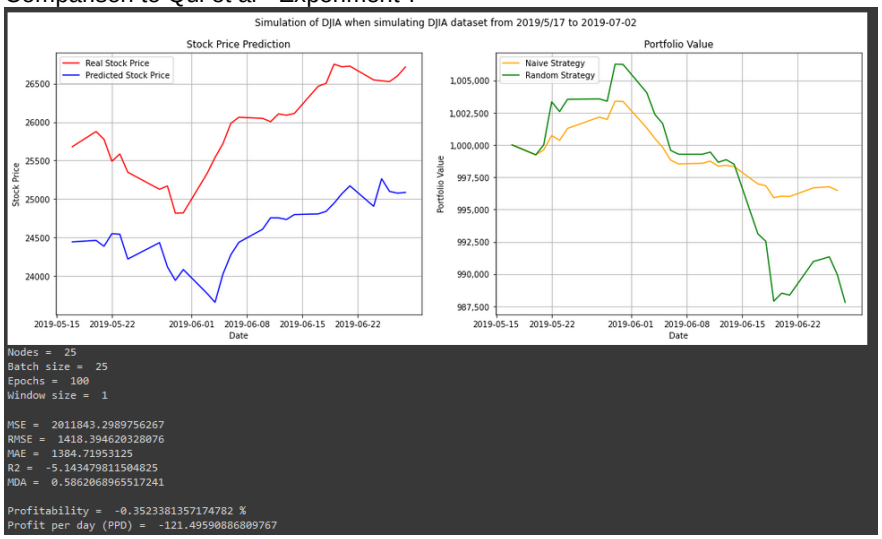
Comparison to Qui et al - Experiment 6



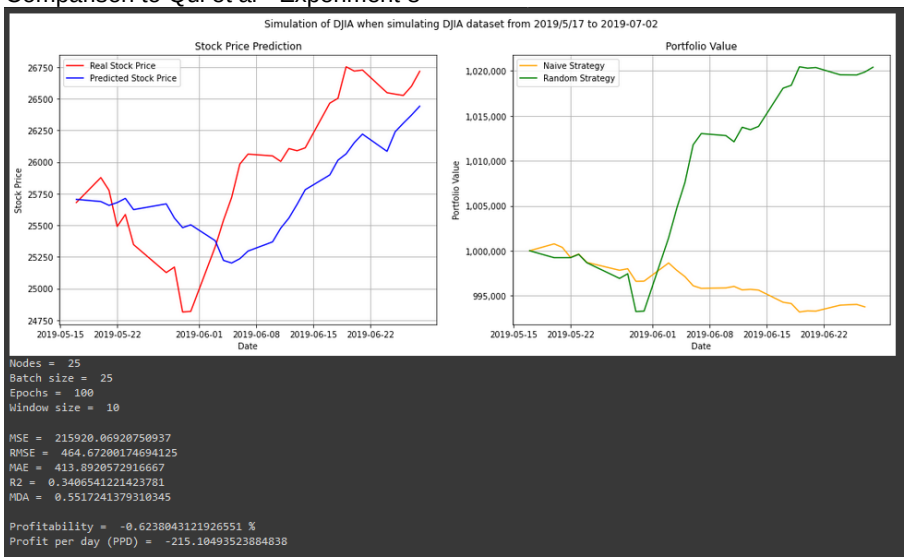
Comparison to Qui et al - Experiment 7



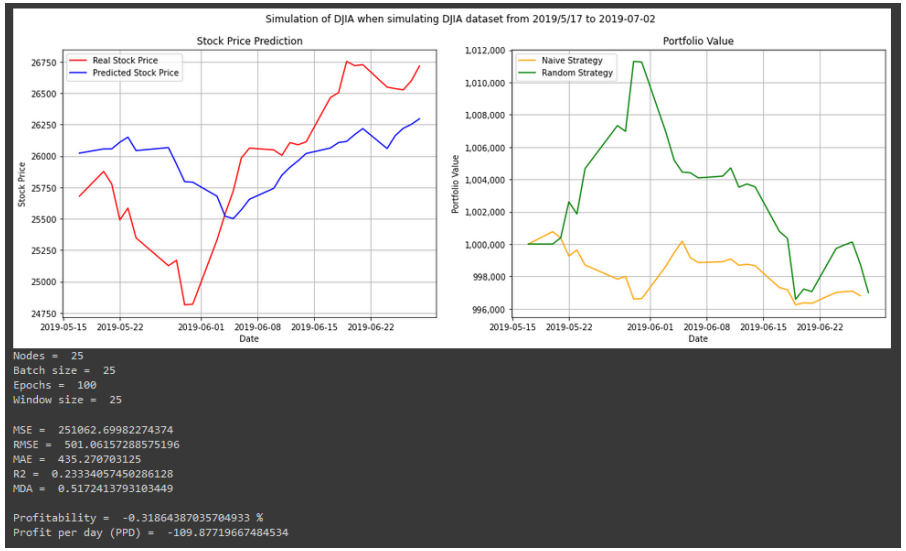
Comparison to Qui et al - Experiment 8



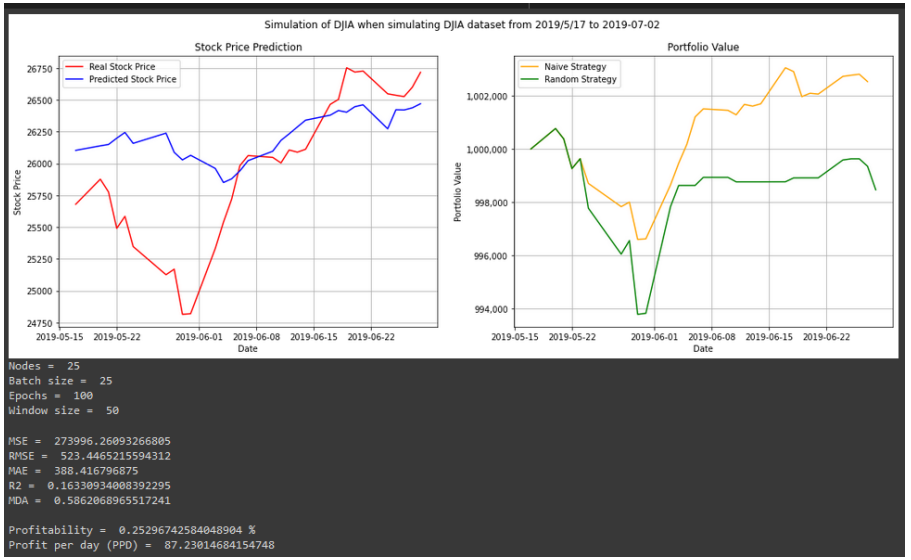
Comparison to Qui et al - Experiment 9



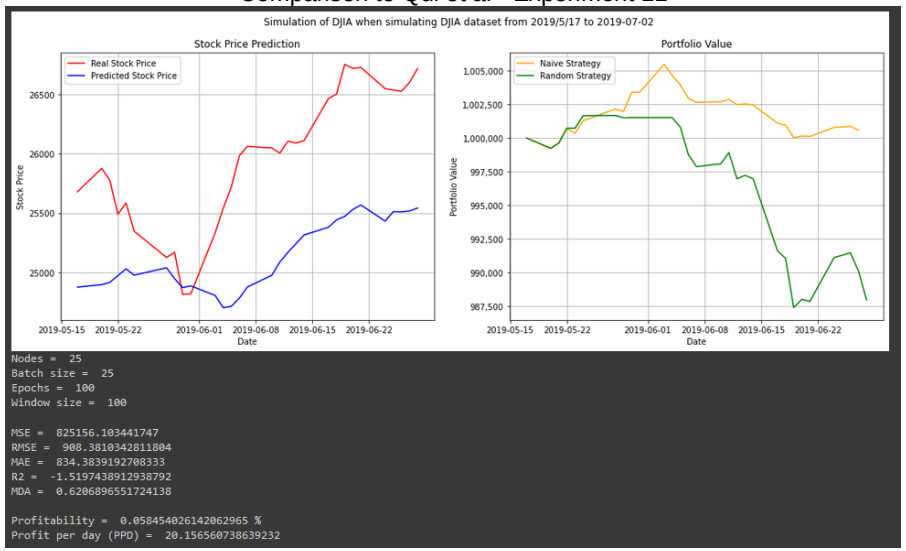
Comparison to Qui et al - Experiment 10



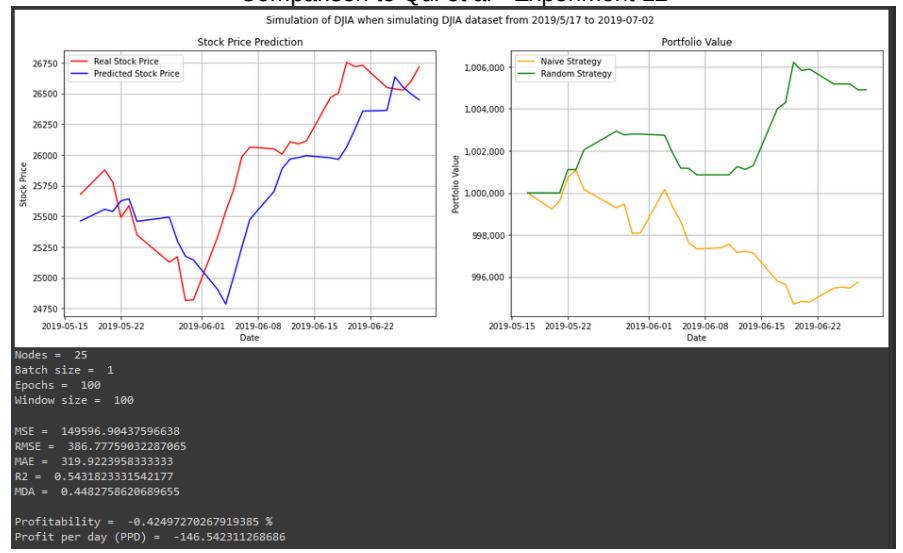
Comparison to Qui et al - Experiment 11



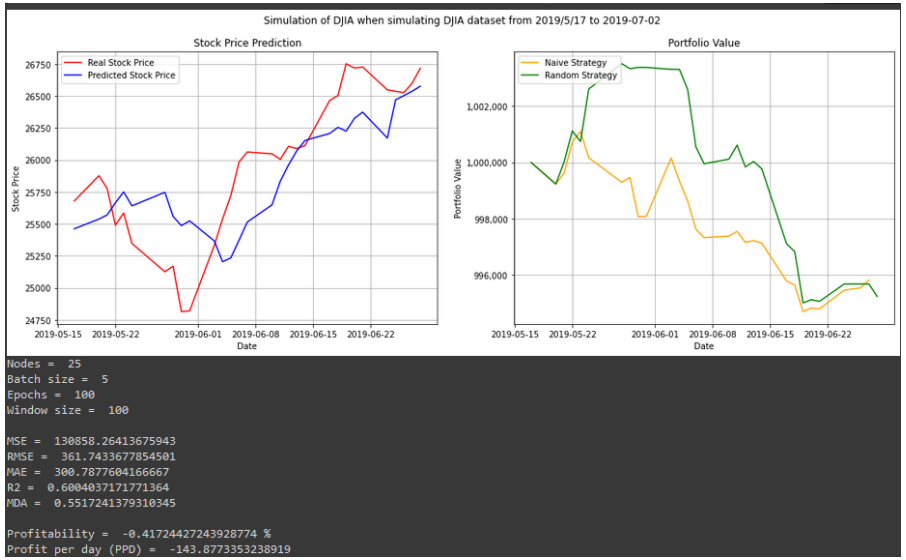
Comparison to Qui et al - Experiment 12



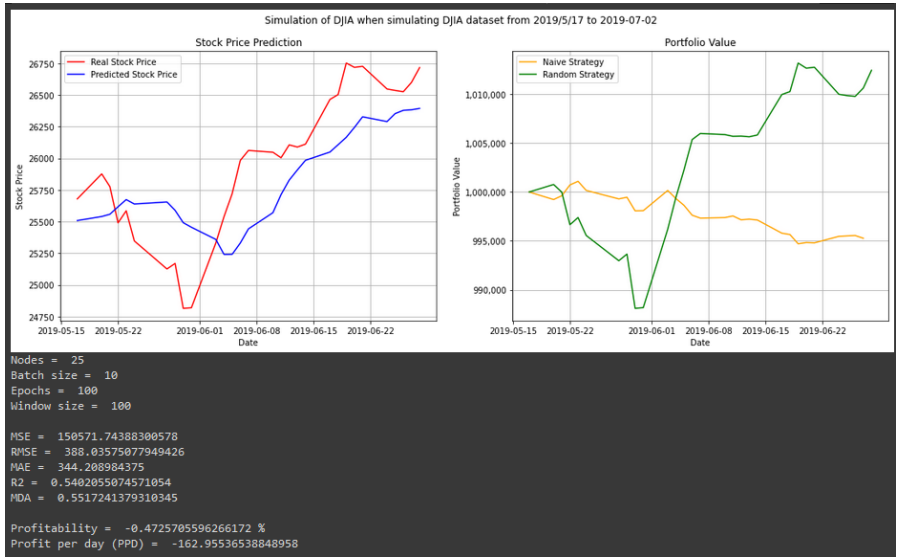
Comparison to Qui et al - Experiment 13



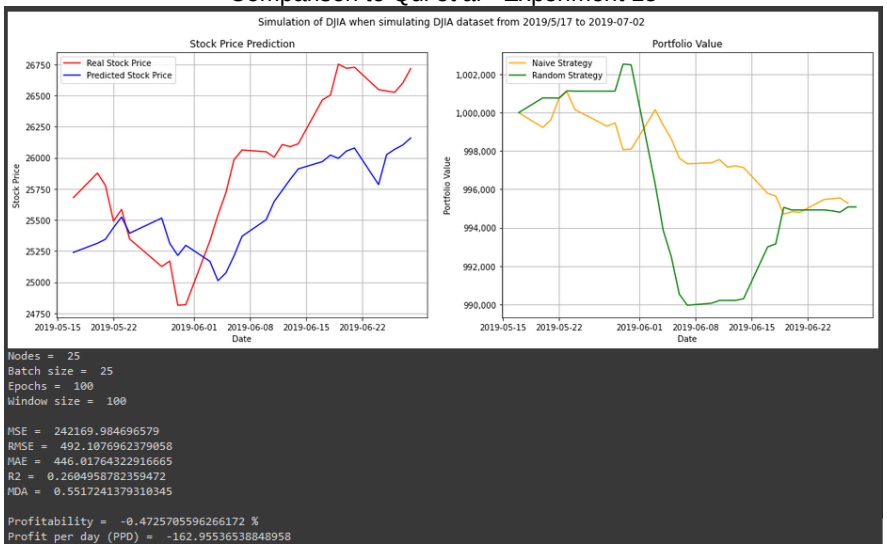
Comparison to Qui et al - Experiment 14



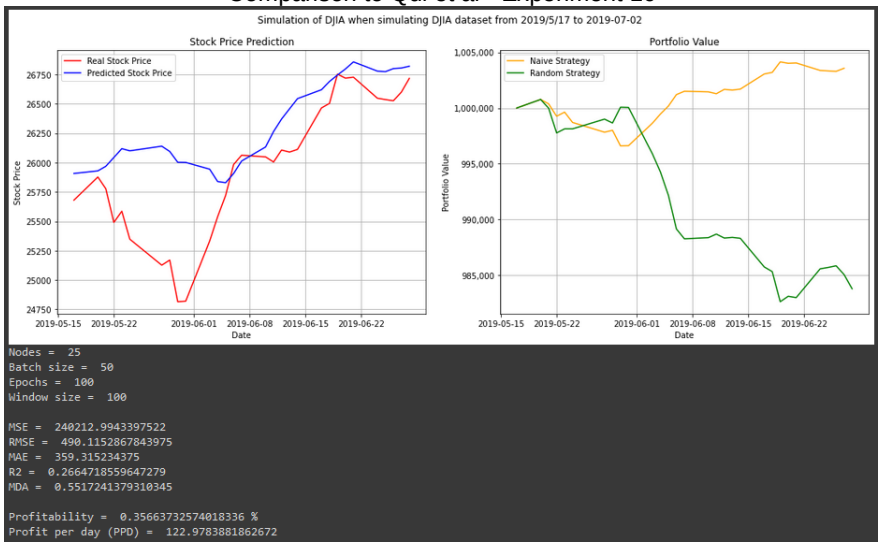
Comparison to Qui et al - Experiment 15



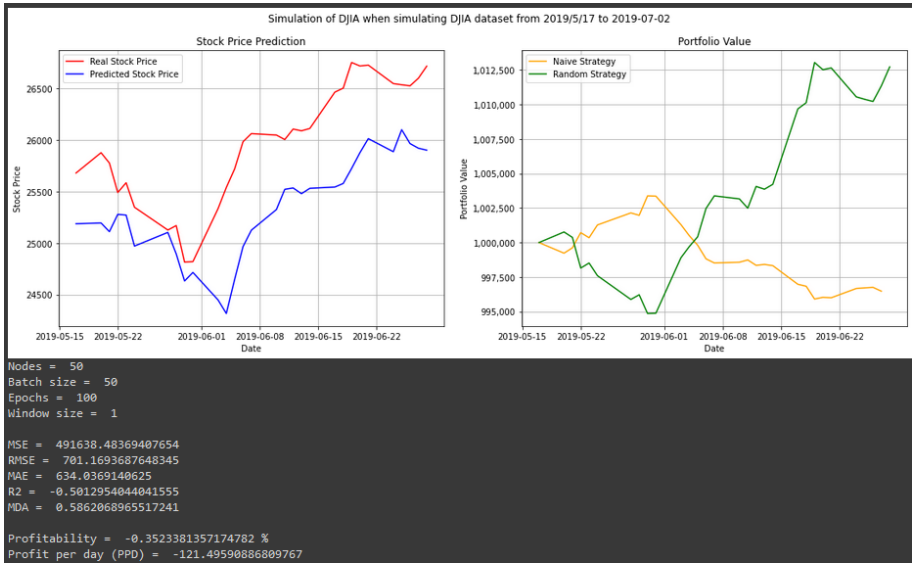
Comparison to Qui et al - Experiment 16



Comparison to Qui et al - Experiment 17

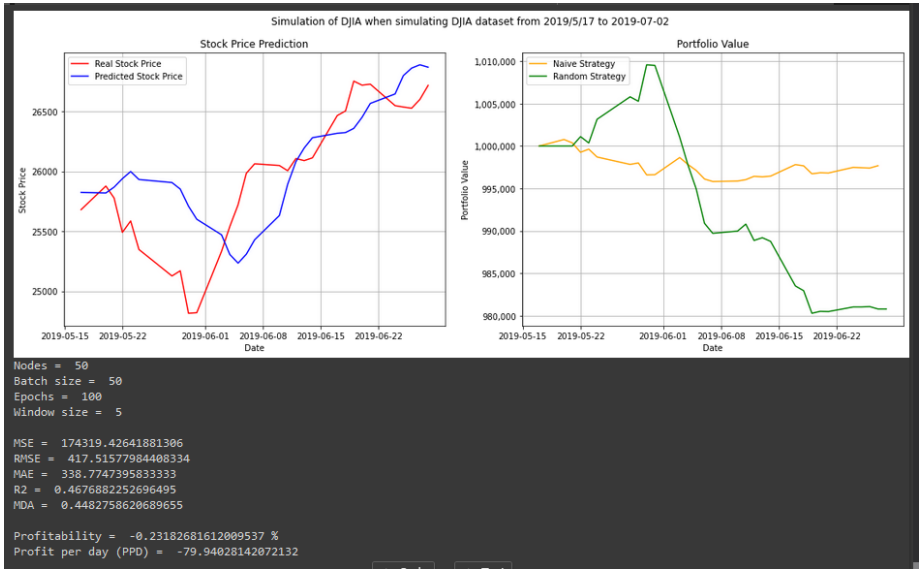


Comparison to Qui et al - Experiment 18



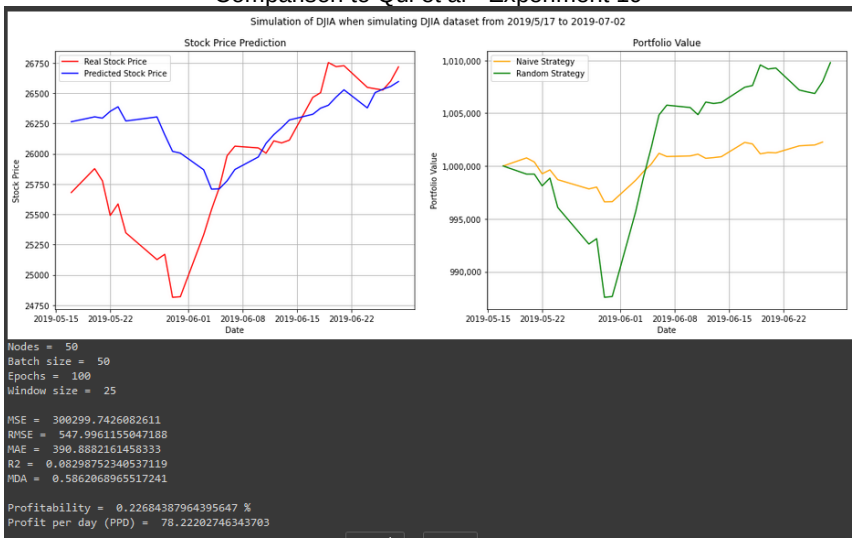

```
Nodes =  50
Batch size =  50
Epochs =  100
Window size =  1

MSE =  491638.48369407654
RMSE =  701.1693687648345
MAE =  634.0369140625
R2 =  -0.5012954044041555
MDA =  0.5862068965517241

Profitability =  -0.3523381357174782 %
Profit per day (PPD) =  -121.49590886809767
```

Comparison to Qui et al - Experiment 19



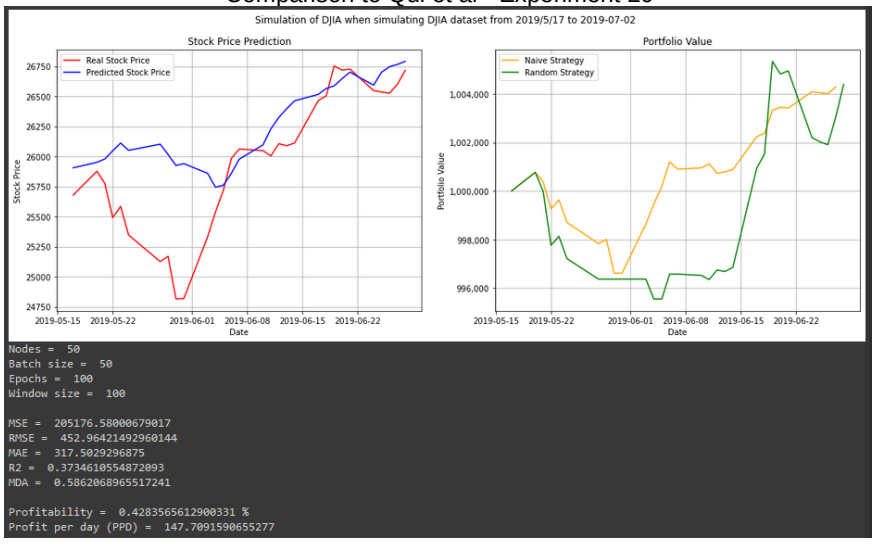

```
Nodes =  50
Batch size =  50
Epochs =  100
Window size =  5

MSE =  174319.42641881306
RMSE =  417.51577984408334
MAE =  338.7747395833333
R2 =  0.4676882252696495
MDA =  0.4482758620689655

Profitability =  -0.23182681612009537 %
Profit per day (PPD) =  -79.94028142072132
```

Comparison to Qui et al - Experiment 20



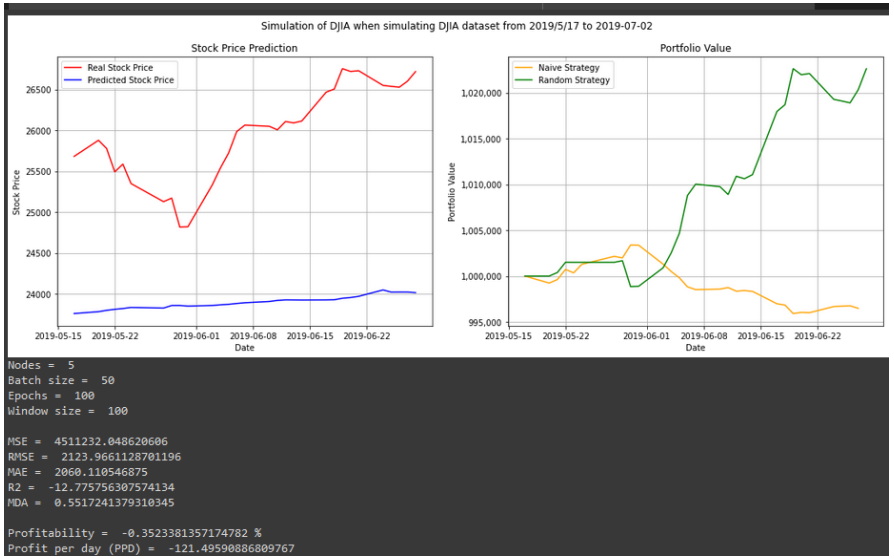

```
Nodes =  50
Batch size =  50
Epochs =  100
Window size =  25

MSE =  300299.7426082611
RMSE =  547.9961155047188
MAE =  390.8882161458333
R2 =  0.08298752340537119
MDA =  0.5862068965517241

Profitability =  0.22684387964395647 %
Profit per day (PPD) =  78.22202746343703
```

Comparison to Qui et al - Experiment 21

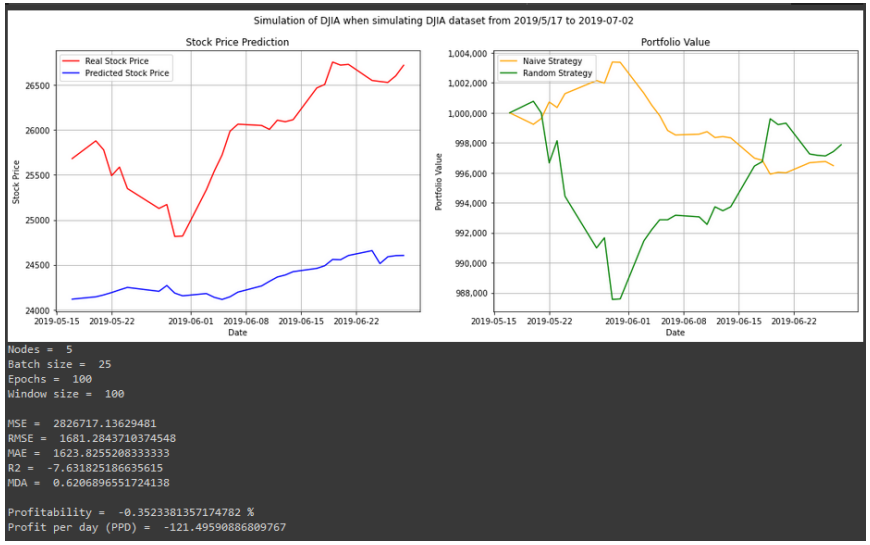



```
Nodes =  50
Batch size =  50
Epochs =  100
Window size =  100

MSE =  205176.58000679017
RMSE =  452.96421492960144
MAE =  317.5029296875
R2 =  0.3734610554872093
MDA =  0.5862068965517241

Profitability =  0.4283565612900331 %
Profit per day (PPD) =  147.7091590655277
```

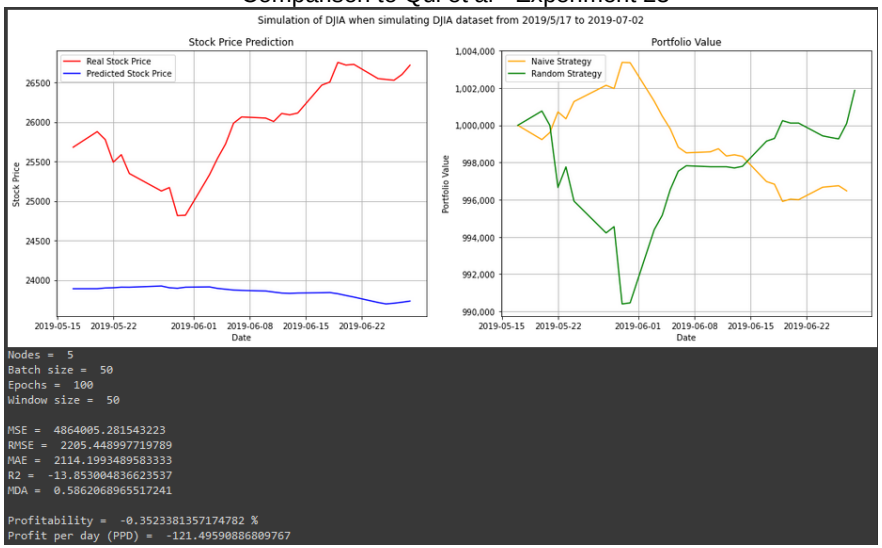
Comparison to Qui et al - Experiment 22



Comparison to Qui et al - Experiment 23



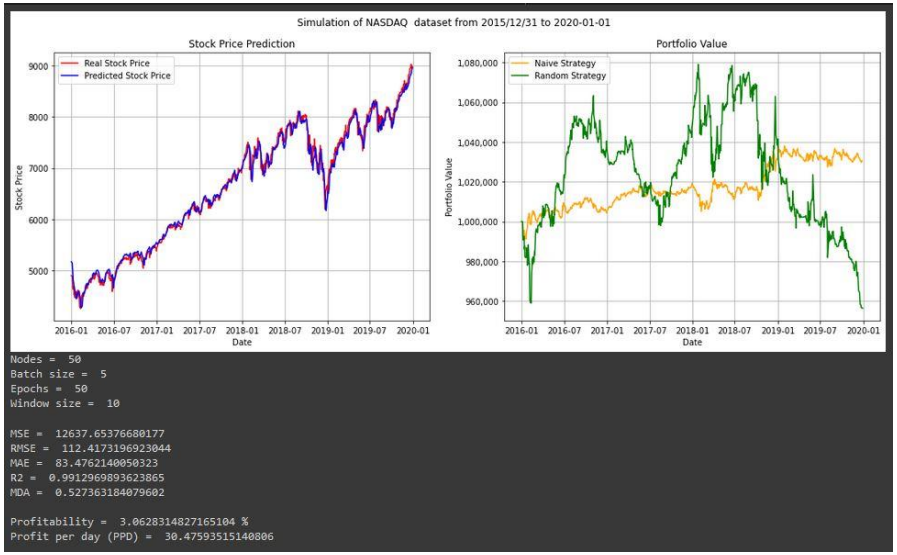
Comparison to Qui et al - Experiment 24



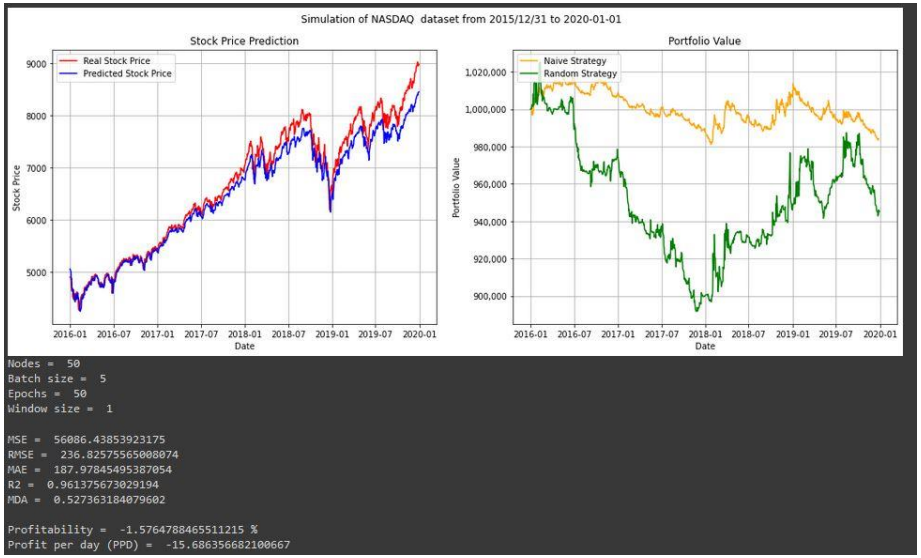
Comparison to Qui et al - Experiment 25

B - 4.2.2.1 Parameter Experiment Results

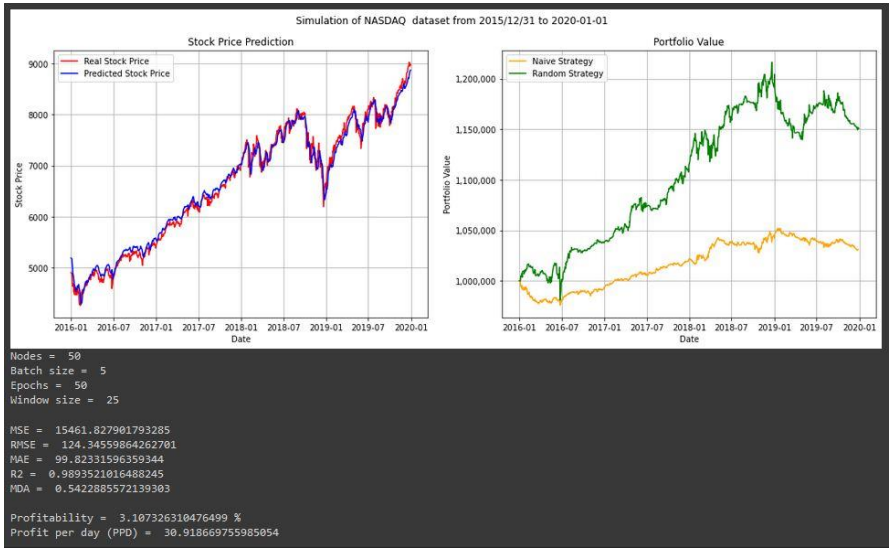
This appendix illustrates the experimental plots described in Section 4.1.2.1. The left figure depicts the difference between the expected (blue) and actual (red) values, while the right plot depicts the difference between the naive (orange) and random (green) trading strategies. The computed performance metrics are provided under the graphs.



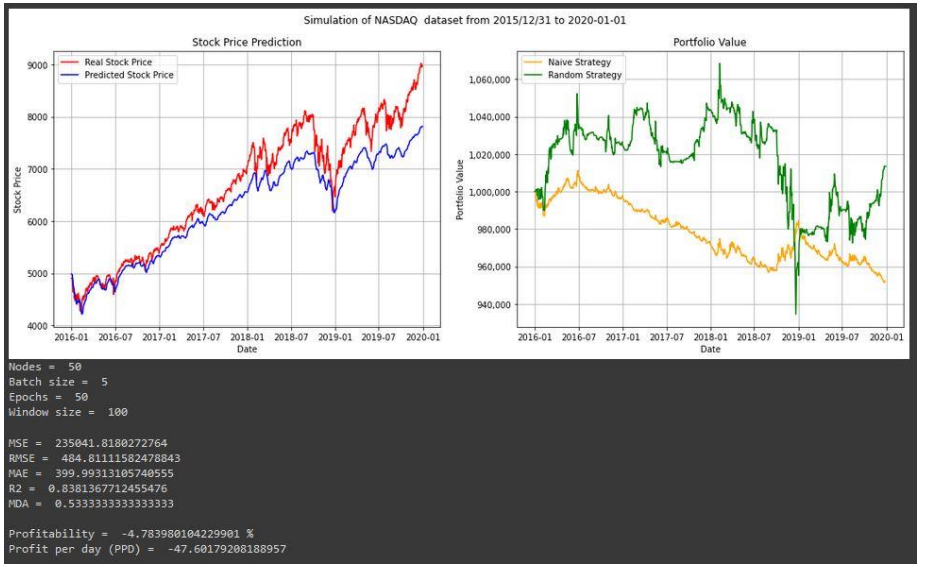
New datasets – Experiment 1



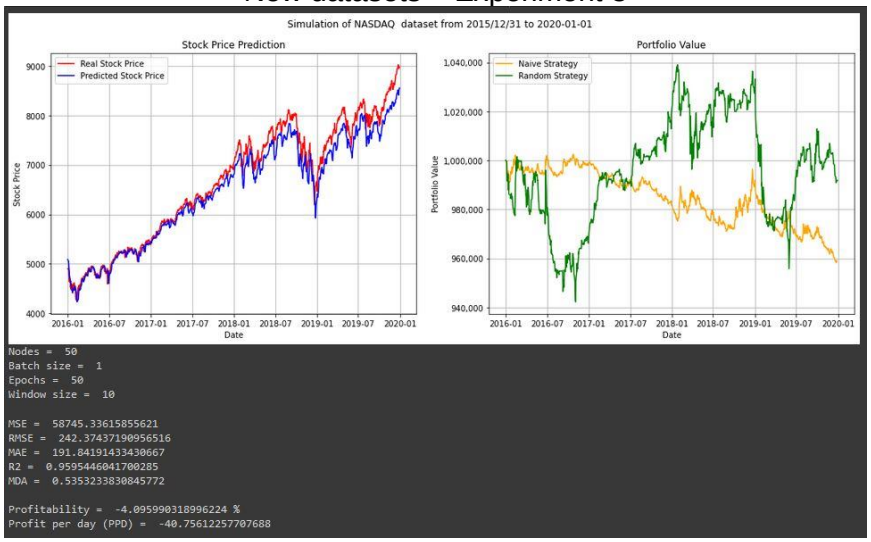
New datasets – Experiment 2



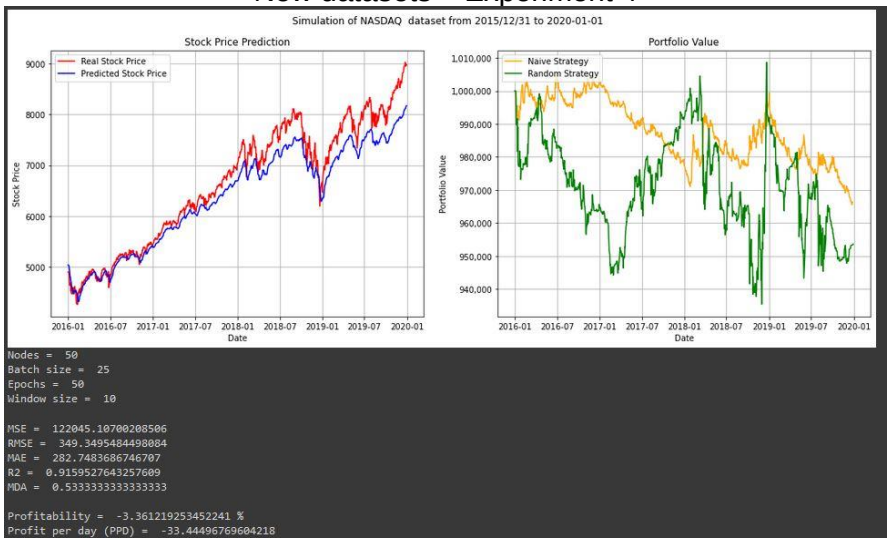
New datasets – Experiment 3



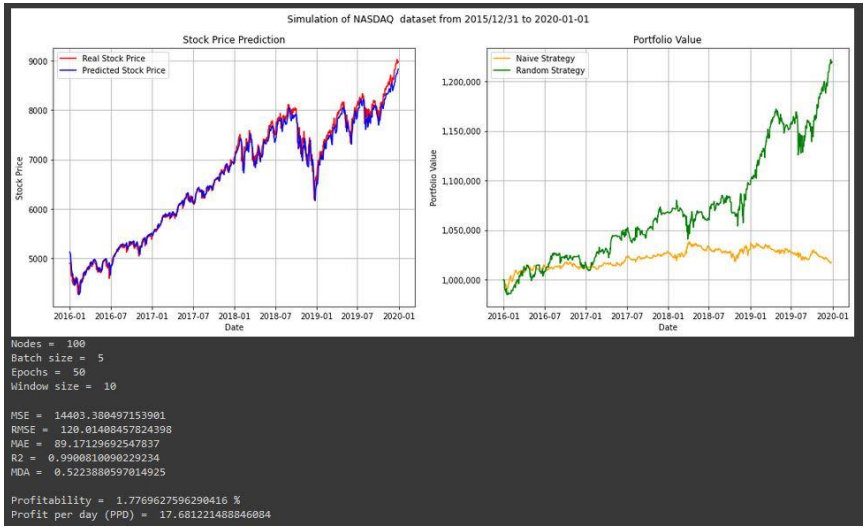
New datasets – Experiment 4



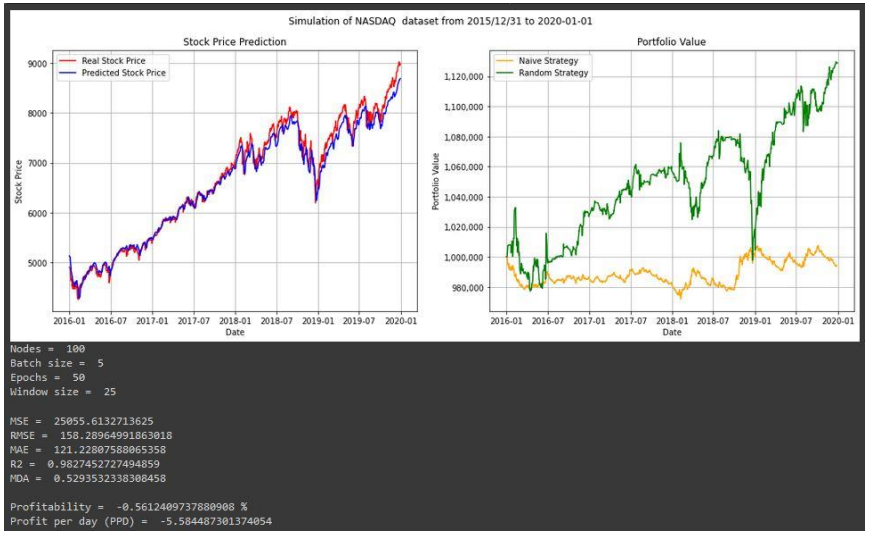
New datasets – Experiment 5



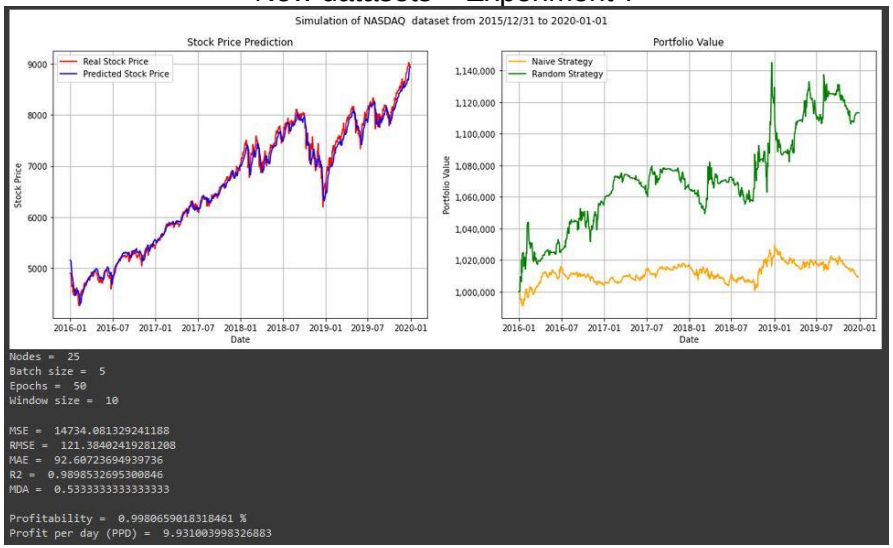
New datasets – Experiment 6



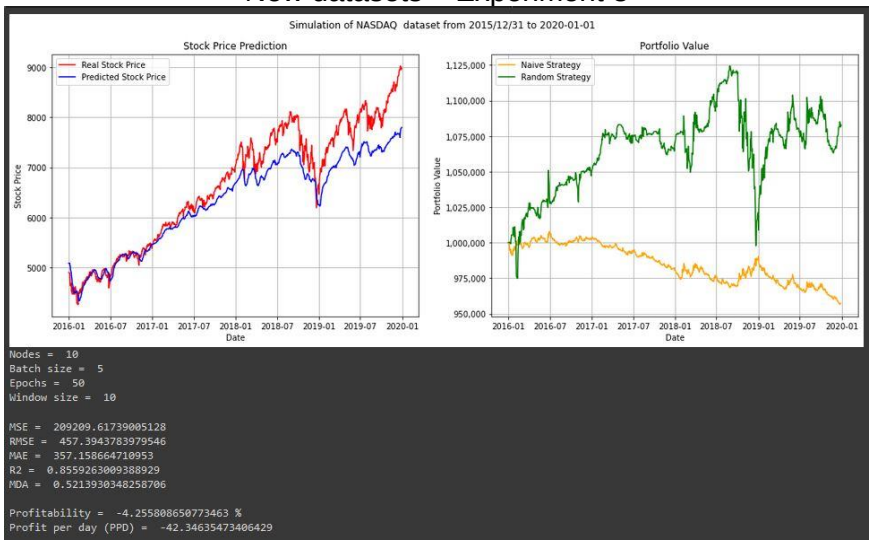
New datasets – Experiment 7



New datasets – Experiment 8



New datasets – Experiment 9

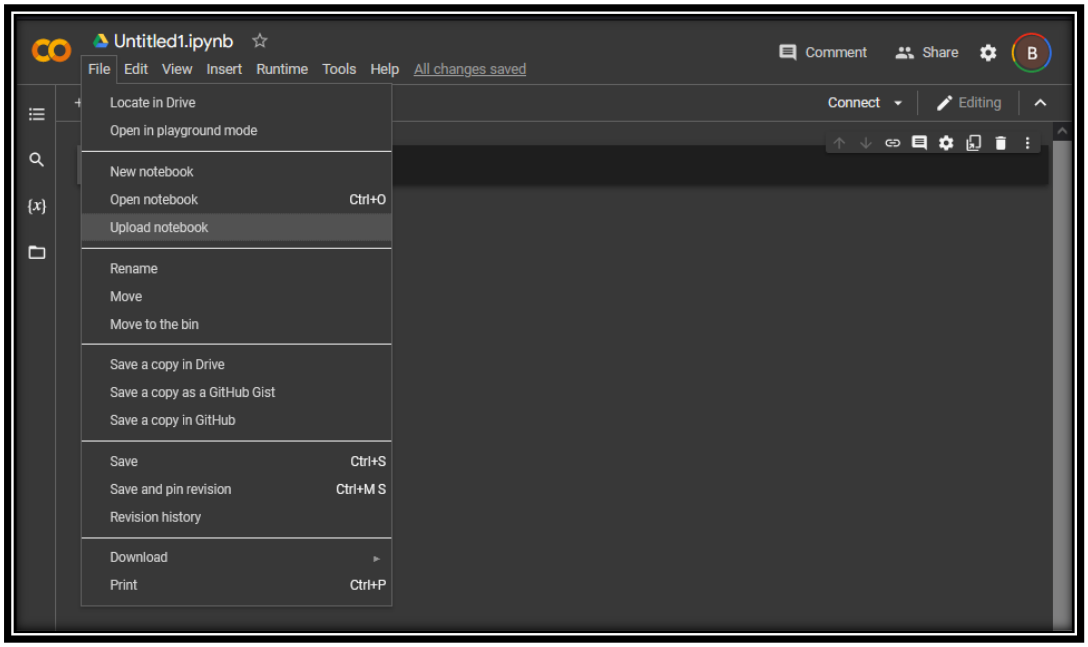


New datasets – Experiment 10

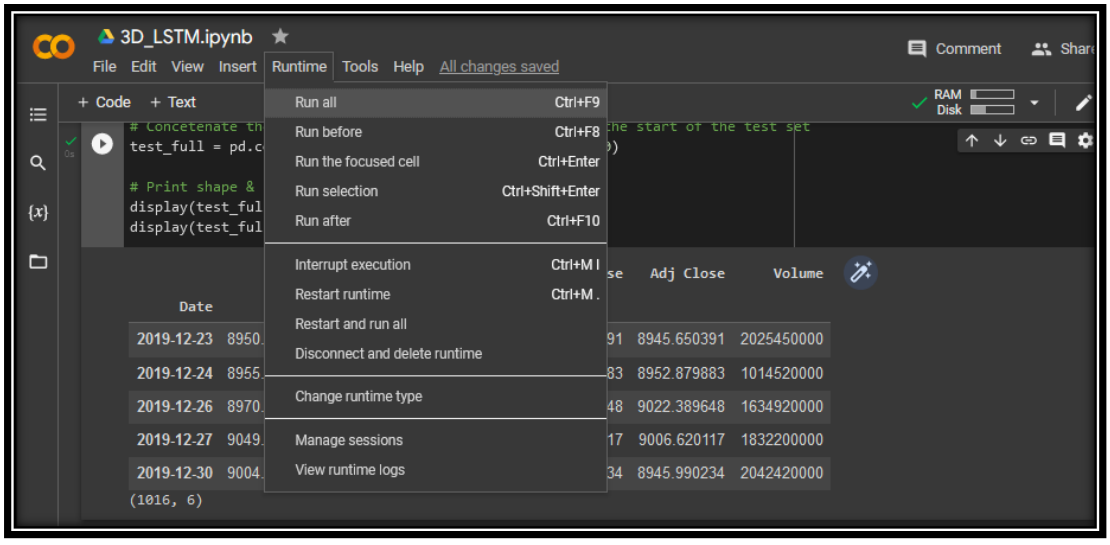
C - Operational Manual

This submission includes the following files and directories:

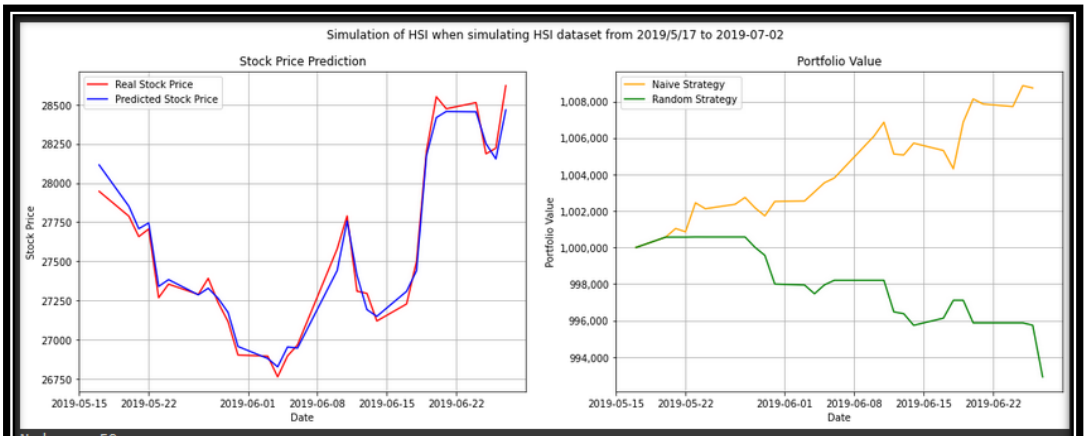
- 3D_LSTM.ipynb
  - The implemented WLSTM+Attention model utilising the sliding window technique, so the data is structured three-dimensionally before input.
- 2D_LSTM.ipynb
  - The implemented WLSTM+Attention model omitting the sliding window technique, so the data is structured two-dimensionally before input.



In order to run the project, follow the link to the Google Colab website (https://colab.research.google.com/). Click File -> Upload Notebook and choose the desired model to run. Once loaded, the model can be launched by Runtime -> Run All. Parameters for each project can be found and changed in specific cells in the notebook.



After the model is executed, the final block in the code shows the final plots and evaluation metrics for the project, as shown below:




```
Nodes =  50
Batch size =  10
Epochs =  100

MSE =  5822.317732317694
RMSE =  76.30411346918129
MAE =  63.945110452586206
R2 =  0.9804961287926041
MDA =  0.8571428571428571

Profitability =  0.8733275491055181 %
Profit per day (PPD) =  311.90269610911076
```