Stock Price Prediction Using RNN-Based Predictive Analytics

Candidate No.: 54588, 56019

April 2023

Abstract

Analysing time series stock market data and predicting stock prices have always been of great interest in the world of economics and finance. However, the complexity and volatility of the market has made stock price prediction a challenging task. Our study aims to apply deep learning models, specifically Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRU) to stock market data and compare their predictive performances against other baseline models such as Autoregressive Integrated Moving Average (ARIMA) and Multi-layer Perceptron (MLP).

To help improve the prediction accuracy, we have included and tuned a large set of hyperparameters in our models. We have also produced a correlation heatmap with the stock data, which was used to guide the selection of the most suitable set of features.

Overall, LSTM and GRU outperform the baseline models when predicting into the future. In the development, both univariate and multivariate models are explored and single-step and multi-step-ahead predictions are developed. Through modifications to the original LSTM and GRU modeling strategies, we are able to achieve an acceptable prediction accuracy for a period of three weeks into the future.

Contents

1 Introduction

1.1 Problem Statement

Understanding the patterns in stock market data has always been a fascinating topic for academia and investors alike. With the fast development of stock markets, investors require efficient and effective tools to help them analyse stocks, giving them enough time to capture arbitrage opportunities and make informed investment decisions that minimise risks. However, predicting stock trends has proven to be a challenging task due to the complex and dynamic nature of the market. In essence, stock price data is a time series with inherent uncertainties.

A lot of research efforts have been expended attempting to predict the trend of the stock market. Many methods have been proposed with varying degrees of success, ranging from linear regression, statistical modeling such as ARIMA, traditional AI modeling such as MLP, and modern deep neural networks such as LSTM and GRU.

To both consolidate our knowledge on AI and to develop useful techniques for stock price prediction, we have designed and implemented several techniques including linear regression, ARIMA, MLP, LSTM, and GRU. The historical stock data of Coca-Cola Co (KO) has been used to evaluate the accuracy of our predictions.

1.2 Literature Review

Time series forecasting are usually achieved using two main approaches[21]: statistical models such as Autoregressive Integrated Moving Average (ARIMA), and neural network models such as Long-Short-Term-Memory (LSTM) and Gated Recurrent Unit (GRU). Statistical models are often used as a baseline model to compare predictive performances of different models [2]. ARIMA is a popular choice due to its simplicity and flexibility. Various studies have explored the use of ARIMA model for stock price prediction and it has shown promising results. In a study by Ariyo et al. [1], ARIMA model was used to predict stock prices for two companies listed on the New York Stock Exchange (NYSE) and Nigeria Stock Exchange (NSE) and they concluded ARIMA model has strong potential for short-term prediction and can compete well with emerging forecasting techniques. However, ARIMA model comes with limitations such as the inability to capture non-linear data very well and the assumption of constant standard deviation in [7].

With the development of the modern AI techniques, recent studies have concentrated on deep learning-based algorithms to improve time series forecasting performances. LSTM and GRU have demonstrated superiority over ARIMA in several studies. In particular, a study by Siami-Namini et al.[18] compared the performance of ARIMA and LSTM model in minimising the error rate in prediction, and it demonstrated that LSTM model reduced error rate by 84-87%. Another recent study by Pirani et al. [15] fitted ARIMA, LSTM, Bi-directional LSTM and GRU model to predict stock prices for IBM, SNOWMAN, HDFC and BHEL stocks and their results suggested that GRU has outperformed the other models, while ARIMA has the highest average RMSE among the 4 companies' stock price prediction.

However, the performance of these deep learning model is inconsistent, with some studies suggesting ARIMA's better performance in predicting financial time series data. For instance, a study by Yamak et al. [22] compared the performances of ARIMA, LSTM and GRU model in predicting Bitcoin's price and it demonstrated that ARIMA has the best performance with RMSE of 302.53, as compared to GRU with RMSE of 381.34.

In light of these findings, our project aims to use ARIMA and MLP models as a baseline for stock prediction, while evaluating performance of deep learning models such as LSTM and GRU. By comparing these models, we aim to identify the best performing model for stock prediction.

2 Architecture

2.1 Statistical Models

2.1.1 Linear Regression

Initially, we considered fitting a linear regression model as our baseline due to its simplicity and widespread use in data analysis. However, to ensure our data meets the Gauss-Markov (GM) assumptions for linear regression, we decided to conduct several tests.

Firstly, we conducted the Shapiro-Wilk Test and produced a Q-Q plot to test for normality in our data [16]. The p-value of the Shapiro Wilk Test is below 0.05, indicating that the residuals not are normally distributed. Moreover, the Shapiro Test Statistic is that not close to 1, and our Q-Q plot shows that the distribution of the residuals doesn't match the normal distribution line very well. Hence we conclude that the residuals are not normally distributed.

We then conducted the Breusch-Pagan test for homoskedasticity to determine if the variance of the residuals was constant across all values of the predictor variables [3]. The p-value is below 0.05, meaning that our data failed the test, and is not homoskedastic.

Finally, we test for linearity through observing the relationship between residuals and predicted values on a scatter plot. The plot shows a downward sloping shape, meaning that the data are not randomly scattered, and thus our data is non-linear.[20]

Therefore, from the tests above, we concluded that our data did not satisfy all GM assumptions required for linear regression. In addition, observing from the correlation plot in EDA section Figure 8, a majority of our predictors are correlated to each other, leading to multicollinearity. Hence, we rejected the use of a linear regression model as our baseline, as it would be unsuitable for our data.

2.1.2 ARIMA

The ARIMA (Autoregressive Integrated Moving Average) model is a widely used statistical model for time series forecasting and we have chosen this to be our baseline model. The model effectively captures the temporal dependencies of a time series through three components: Autoregression (AR), differencing terms, and moving average (MA) [12].

The Autoregression (AR) component models the influence of past values on current values, and has an order of p, denoting the number of lagged observations to consider for autoregression. The AR component can be represented as follow:

$$y_t = \phi_1 y_{t-1} + \phi_2 y_{t-2} + \ldots + \phi_p y_{t-p} + \epsilon_t$$

where y_{t-1} is the first lag of the series and ϕ_1 is the coefficient of the first lag.

The Moving Average (MA) component models the influence of past errors on current values, and has an order of q, denoting the size of the moving average window. It is given by the equation:

$$y_t = \epsilon_t + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + \ldots + \theta_q \epsilon_{t-q}$$

where ϵ_{t-1} is the error term of the first lag.

A key requirement of the ARIMA model is that the input data must be stationary. This is to remove any obvious correlation and collinearity with the past data, reducing incorrect bias during model training. To achieve stationarity, we would employ different techniques such as differencing to transform non-stationary data into stationary data. d is the order of differencing, denoting the

number of times the raw observations are differenced.
Combining the three components, we can represent the ARIMA model in a single equation:

$$(1-L)^d y_t = c + \phi(L) y_t + \theta(L)\epsilon_t$$

where $\phi(L) = 1 - \phi_1 L - \phi_2 L^2 - ... - \phi_p L^p$ and $\theta(L) = 1 + \theta_1 L + \theta_2 L^2 + ... + \theta_q L^q$ are the lag polynomial representations for the AR and MA components and c is a constant term.

2.2 Neural Networks

2.2.1 Multilayer Perceptron (MLP)

Multilayer perceptron is a standard neural network architecture and has been deployed for many AI tasks [23]. MLPs consist of multiple layers of interconnected nodes with connections between layers. Each node calculates a weighted sum of its inputs and applies an activation function (e.g. ReLU, Sigmoid or Softmax) to produce its output. The activation function introduces non-linearity to the network, allowing it to learn complex relationships in the data. A simple, one hidden layer MLP structure is shown below [23].

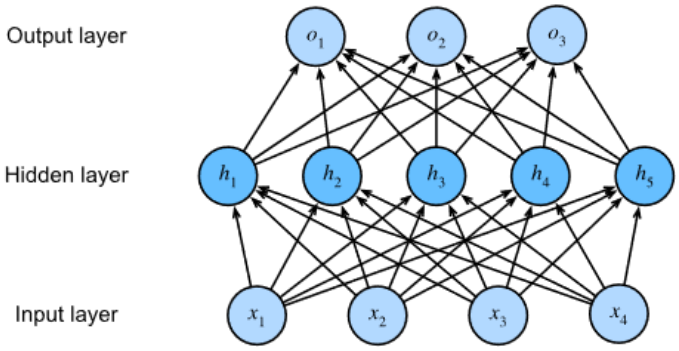

Figure 1: Simple MLP structure [23]

Multilayer Perceptron was one of the earliest techniques used by researchers attempting to predict stock prices [10]. Due to lack of past performance memory, MLPs were used to forecast the price on the next data point, not for long term future. Our plan is to use it as a baseline.

2.2.2 Long Short-Term Memory (LSTM)

In order to handle sequential data, such as time series, a number of modern neural networks have been developed, including the Recurrent Neural Network (RNN). RNNs have a recurrent (memory) component that allows the network to remember the past information, which then informs the current output [23]. This memory ability differs it from the conventional neural networks. However, a common issue with the RNNs is the so called vanishing and exploding gradients problem. In 1997 a special RNN called the long short-term memory (LSTM) model was developed by Hochreiter and Schmidhuber [9], where they have successfully solved this problem.

LSTM networks use a series of gates to control the flow of information through the network, including input gates that allow information to enter the network, forget gates that allow information to be discarded, and output gates that allow information to leave the network. The sigmoid activation functions are added to compute the values of the three gates.

Making use of the three gates, the LSTM has three main components: the input node, memory cell, and hidden state. The input node is responsible for taking in the current input to the LSTM network, which feeds the input into the memory cell. The memory cell is the central unit, which interacts with the input node and the hidden state. The hidden state is the output of the memory

cell and is used to make predictions or to provide context for the next input. The figures below illustrate the structure of the three components [23]. They work together to store and process the information and makes decision on what should be passed on to the next stage. The following illustrates the LSTM internal structure, showing the three main components [23].

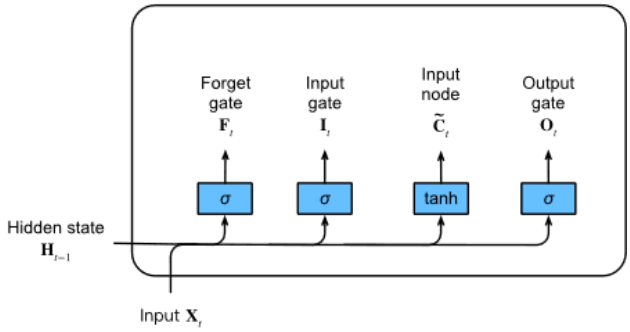

Figure 2: LSTM input node

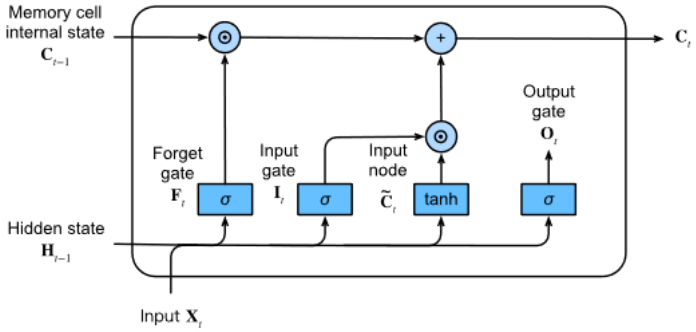

Figure 3: LSTM memory cell

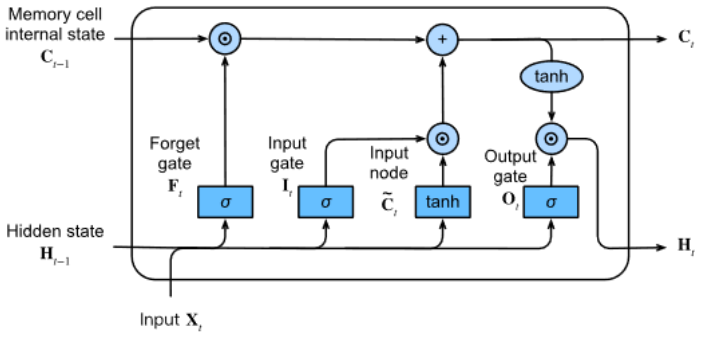

Figure 4: LSTM hidden state

This gating mechanism allows LSTMs to selectively remember or forget information over long time intervals, making them particularly useful for tasks with long-term dependencies. As stock prices change over time but also with connections to the history of the stock performances, LSTM is a popular model for stock price prediction ([6]; [10]; [17]).

LSTM networks can be modelled to handle either a univariate problem or a multivariate problem. For the former, the model takes only the historical data as the input; and for the latter it can take different feature data in addition to the historical price data, such as the opening price of the day and trading volume. What's worth noting is the network will take data sequences, rather than individual prices, which means care needs to be taken in preparing the input data in the implementation. The details of implementation will be discussed in the Methodology Section.

2.2.3 GRU

Gated Recurrent Unit is another specific RNN. With the popularity of LSTM, researchers during the 2010s began experimenting different variants with a purpose to simplify the architecture and speed up the computation while retaining the key advantages of the LSTM networks. In 2014 Cho et al proposed the Gated Recurrent Unit (GRU) architecture ([4]).

Regarding the architecture, LSTM's three gates are replaced by two GRU gates: the reset gate and the update gate. These two gates are also given sigmoid activations. The reset gate controls which information from the previous time step should be forgotten, while the update gate controls which information from the current time step should be added to the hidden state. Using the gated mechanism, GRU has two states. The candidate hidden state is an intermediate state. It combines information from the previous hidden state with the current input to form a new candidate state. It is updated by the reset gate and passed to the update gate. The hidden state is the output of the GRU cell, which is the final hidden state after information has been passed through the reset and update gates. The structure of the two main states [23] is illustrated below.

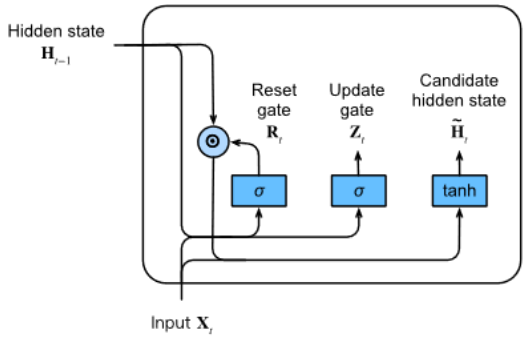

Figure 5: GRU candidate hidden state

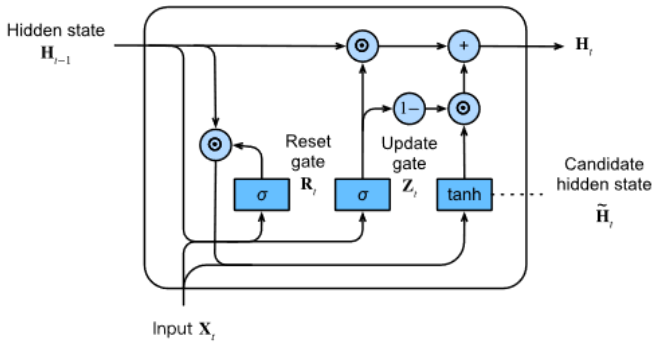

Figure 6: GRU hidden state

Compared to LSTM, GRU has fewer parameters, making it faster to train and less prone to over-

fitting, while still maintaining strong performance on tasks. Due to its efficiency and effectiveness, GRU has become a popular choice for time series tasks.

In terms of implementation, GRU is to a large extent similar to LSTM. It also takes sequences as inputs and can be modelled to handle uninvariate and multiverse cases.

3 Methodology

3.1 Outline

Our project is concerned with exploring AI methods for the purpose of stock price prediction. Our methodology and implementation are outlined below:

1. Identification of suitable data sources and undertaking financial data processing;
2. Feature data collection, selection with correlation heatmaps;
3. Experimenting with some traditional methods (statistical, neural network) to find out the baseline performance on stock price prediction;
4. Implementation of modern AI methods capable of dealing with time series data, such as stock prices;
5. Study of both univariate and multivariate models;
6. Hyperparameters tuning to select the best combination of hyperparameters;
7. Development of both single-step-ahead and multi-step-ahead predictions.

3.2 Data Preprocessing

There are several well-established financial data sources. We have used the Yahoo Finance site as our main data source, where we download, clean and format the historical datasets for indices, funds, and individual stocks to suit our computational requirements.

3.2.1 Data Scaling

Stock data vary enormously in scales, such as price range and trading volume. It can skew the training process unexpectedly if the raw data were used directly, as some data may be falsely regarded as more / less important by giving a much larger or smaller influences than necessary. We tackle this issue with data normalisation using Min-max scaling [23], which scales the data in a range between 0 and 1 or a user specified range.

3.2.2 Extract, Transform and Load (ETL)

The project focuses on stock price prediction for Coca Cola. We chose to take data from 2014 to 2022, with our main data sources from Yahoo Finance, and the Federal Reserve Economic Data site.

We explore two types of data. The first kind is more directly related to Coca Cola. For example, volume of Coca Cola stock, 100 day moving average of Coca Cola stock, sugar prices, stock price of competitors and retail sales. The second kind tries to capture general market conditions. For example, US GDP, CPI, market volatility, indices, and Gold. See Table 1 for details.

3.2.3 Exploratory Data Analysis (EDA)

Data Description

The data consists of 1965 rows, 34 columns and no missing values. The column headers provides a comprehensive range of features including basic stock information, technical indicators, macroeconomic data and comparable companies data. Summaries of each features can be found in Table 1 below.

Predictors	**Type**	**Description**
Open	Numerical	Opening price on trading day
High	Numerical	Highest price on trading day
Low	Numerical	Lowest price on trading day
Close	Numerical	Closing price on trading day
Adj Close	Numerical	Closing price on trading day adjusted for any corporate actions
Volume	Numerical	Volume on trading day
Prev_Volume	Numerical	Volume on previous trading day
Prev_Close	Numerical	Closing price on previous trading day adjusted for any corporate actions
Weekday	Numerical	day of the week (1- Monday, 2- Tuesday, etc)
200SMA	Numerical	200-day simple moving average
100SMA	Numerical	100-day simple moving average
20SMA	Numerical	20-day simple moving average
GDP	Numerical	US GDP data obtained from Federal Reserve Economic Data (FRED)
COKE	Numerical	Closing price of comparable company: Coca-Cola Bottling Co. Consolidated
MNST	Numerical	Closing price of comparable company: Monster Beverage Corporation
PEP	Numerical	Closing price of comparable company: PepsiCo, Inc.
AAPL	Numerical	Closing price of related company: Apple Inc
BRK-A	Numerical	Closing price of related company: Berkshire Hathaway Inc Class A
SHEL	Numerical	Closing price of related company: Shell PLC
XOM	Numerical	Closing price of related company: Exxon Mobil Corp
S&P500	Numerical	S&P 500 Index closing price (US)
NASDAQ	Numerical	NASDAQ Composite closing price (US)
Dow Jones	Numerical	Dow Jones Industrial Average closing price (US)
FTSE 100	Numerical	FTSE 100 Index closing price (UK)
DAX	Numerical	DAX Performance Index closing price (Germany)
Nikkei 225	Numerical	Nikkei 225 Index closing price (Japan)
Hang Seng Index	Numerical	Hang Seng Index closing price (Hong Kong)
Market Volatility	Numerical	CBOE Volatility Index
CPI	Numerical	US CPI data obtained from Federal Reserve Economic Data (FRED)
USD Index	Numerical	Nominal Broad US Dollar Index obtained from Federal Reserve Economic Data (FRED)
Retail Sales	Numerical	US Advance Retail Sales data obtained from Federal Reserve Economic Data (FRED)
Gold	Numerical	Gold closing price
Raw Sugar Futures	Numerical	Raw Sugar Futures closing price

Table 1: Summary of all features

Analysing relationship between predictors

Figure 7 displays Coca-Cola's stock prices in 10 years. We observed that there is a general upward trend in the stock data, with the exception of a significant decline in stock price in the Q1 2020 potentially attributable to COVID-19 pandemic.

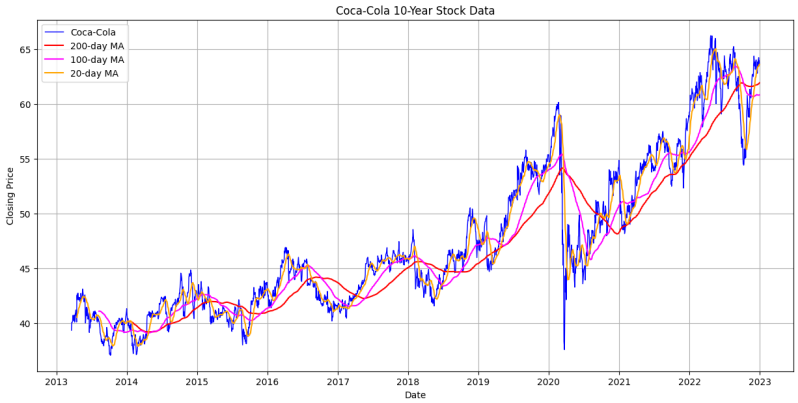

Figure 7: Coca-Cola 10 Year Stock Prices with Moving Averages

The correlation plot in Figure 8 shows that the Closing Price of Coca-Cola stock is highly correlated with various macroeconomic indicators such as GDP, S&P500, CPI and Retail Sales, with the majority of the correlation coefficients being higher than 0.8. This makes sense since demand for consumer products such as Coca Cola products tend to be sensitive to changes in the macroeconomic environment in US and around the world. When the economy is doing well, consumers have higher confidence and are more willing to spend on discretionary items like soda. Furthermore, we discovered another interesting correlation between Coca-Cola stock price and other stocks in different industries such as Apple Inc. (AAPL) and Berkshire Hathaway (BRK-A). This suggests that incorporating these stock prices as predictors could improve our models' prediction on Coca-Cola's stock price.

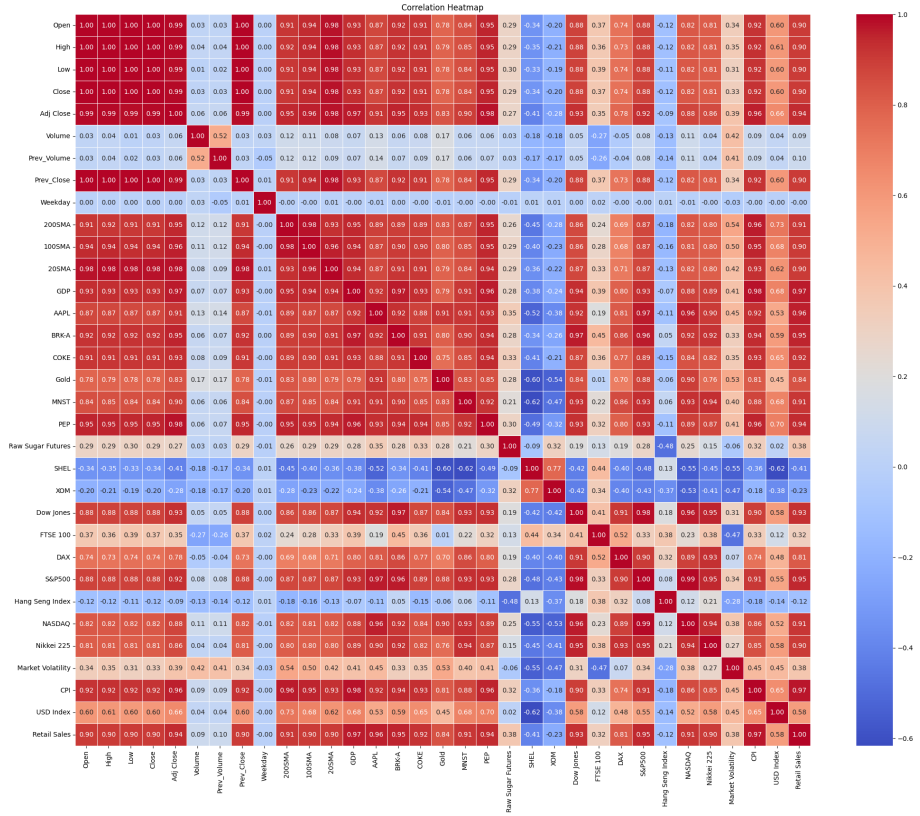

Figure 8: Correlation Heatmap of all features

3.3 ARIMA

The ARIMA model was selected as our baseline since it is more flexibility than Linear Regression model and it allows for the incorporation of autoregressive (AR), moving average (MA) and integrated (I) components depending on the characteristic of the time series data. Furthermore, this approach captures past fluctuations as well as overall trends, and deals with temporary abnormal changes in the data. We have taken inspiration on the implementation of ARIMA model through various sources such as YouTube videos and articles. We have concluded that the ARIMA model's algorithm is as presented in Figure 9 [5].

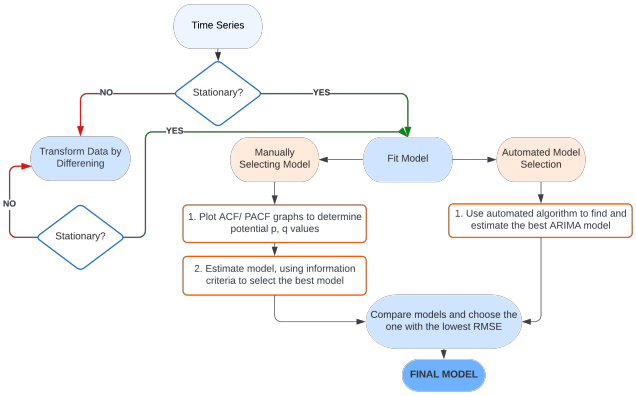

Figure 9: ARIMA Algorithm

3.3.1 Hypothesis Testing

To determine stationarity of our data, we conducted the Augmented Dickey Fuller (ADF) test. If the p-value of the test is ¡ 0.05, the null hypothesis can be rejected, and we can infer that the time series is indeed stationary.

ADF Statistic: −1.2383123986724418
p−value: 0.6569014655022156

Observing from our result above, our p-value is above 0.05, meaning that our data is indeed non-stationary so our data must be transformed.

To reduce the growing trend in the series as seen in Figure 10, we first take log of the series, followed by applying differencing of order 1, which subtracts the previous value in the time series from the current value. We then applied the ADF test again to verify our transformed data is stationary, allowing us to proceed with fitting the model.

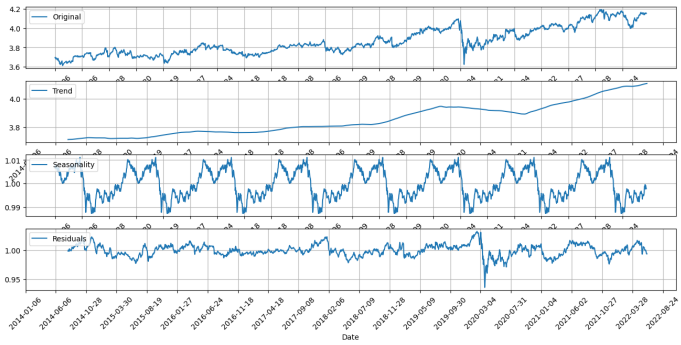

Figure 10: Decomposition of Stock Data into Trend, Seasonality and Noise

3.3.2 ARIMA Methods

There are 2 main methods to fit an ARIMA model[12]:
1. Manually choosing the p,q parameters through Autocorrelated Function (ACF)/ Partial Autocorrelated Function (PACF) graphs
2. Using `auto_arima()` function to find the optimal p,q parameters

We will be using both method to find the most accurate ARIMA model.
For the first method, the optimal p and q values were determined by observing the ACF/ PACF plots. The optimal p value is the lag at which the PACF plot shows a significant spike and then cuts off or drops significantly. The optimal q value is the lag at which the ACF plot shows a significant spike and then cuts off or drops significantly.

3.3.3 ARIMA Prediction Results

From our ACF, PACF plots, we could infer that the optimal (p,q) terms is (3,3). We then fitted an ARIMA model with order (3,0,3) and it results in a train RMSE of 2.9174 and test RMSE of 7.4583.

We then used the `auto_arima()` function and the optimal (p,d,q) terms is (5,0,1). Another ARIMA model with order (5,0,1) was fitted and the results are shown in Figure 11 below, with train RMSE of 2.8929 and test RMSE of 7.3977.

As the model from the `auto_arima()` function has lower RMSE, we conclude that the second model would be used as our final baseline model.

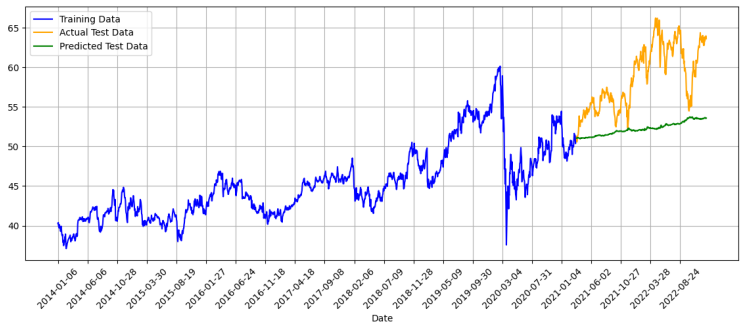

Figure 11: Stock Price Prediction against Actual Data

3.4 MLP

MLP works well for many interpolation tasks including those with a sophisticated nonlinear nature. However, stock price prediction is quite different, as the data is a time series and it is in essence an extrapolation problem, not an interpolation one, which can prove challenging for MLP. Our intention is to treat MLP as a baseline model, which can then be compared with other modern methods developed to tackle time series data.

3.4.1 Model Architecture

Our approach to avoiding overfitting while effectively capturing the underlying patterns is to begin with a simple MLP structure and adjust it as needed based on prediction outcomes. Inspired by the examples in [23], we developed a simple MLP model. The initial design consists of an input layer, a hidden layer and an output layer. A ReLU activation is applied. The following shows the initial result (see Figure 13). Unfortunately, the prediction accuracy is very poor.

After conducting multiple tests using different combinations of layers and units, we found that adding more units and layers did not improve the results. In fact, it has made the results worse. Our experiments showed that the best outcome was achieved using a very simple linear layer, as illustrated in the Figure 12.

3.4.2 K-fold Cross Validation

K-fold cross validation is a popular technique to assess the performance of a model during training. It divides a dataset into k equal folds (subsets), where k is an integer. The model is then trained and tested k times, with a different fold used for testing each time, and the remaining folds are used for training. K-fold cross-validation can help reduce overfitting, as it provides a more accurate estimate of the model's generalisation performance [23]. In order to both consolidate our knowledge and best the prediction accuracy, the K-fold cross-validation strategy was incorporated in the MLP training model.

3.4.3 MLP Prediction Results

Below are the MLP model predictions. Note that due to the nature of MLP, it can only predict the outcome for the next data point. Although the results look good, in the real world it is much more useful to be able to predict the closing price more in advance. Therefore we will do multi-step prediction with some later models.

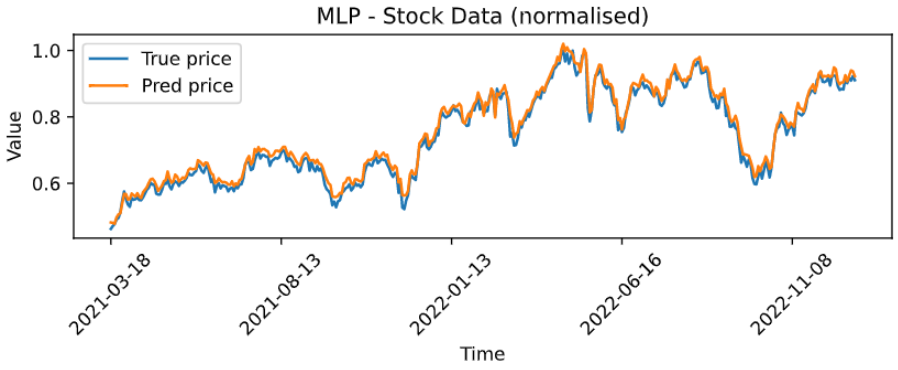

Figure 12: MLP predictions, 1 layer (1 day)

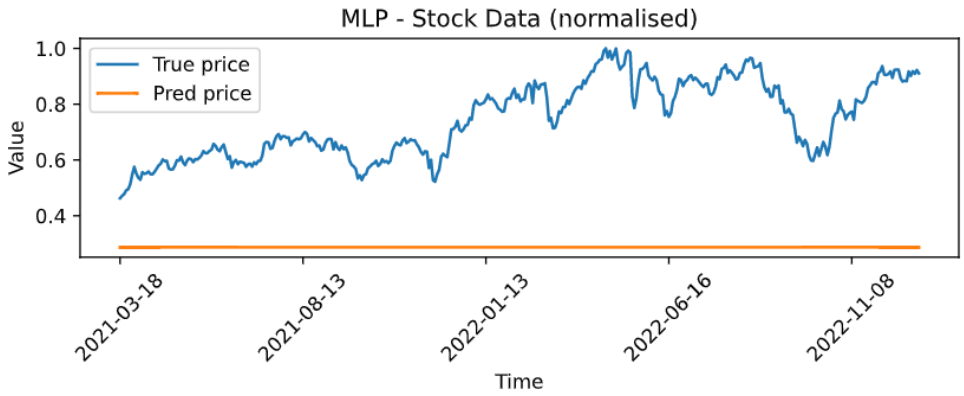

Figure 13: MLP predictions, 2 layers (1 day)

The results suggest that since the datasets and number of features are relatively small, sophisticated structured MLPs tend to suffer from overfitting, while simpler structures work more effectively.

3.5 LSTM

LSTMs as a special type of RNNs are especially suitable for dealing with sequence data and have been widely used for stock price prediction. Our implementation has been inspired by several sources, including ([23]; [2]; [13]; [6]).

3.5.1 Input Data Preparation

As LSTM takes sequences as the input, before feeding the stock price data to the training process, the first task is to convert the historical data of the stock concerned into sequences. We have adopted the rolling window technique ([17]). We take a window of size seq_len, slide it one time-step at a time, and produce a sequence. The rolling window approach allows us to capture temporal dependencies and patterns in the data.
One important parameter is the window size or sequence length seq_len. The model uses seq_len data points to forecast the next time step (for single step prediction) or several steps (for multiple step prediction). To convert the raw data into sequences, we have considered two scenarios:

- Univariate modelling, i.e. only the historical price data are used
  The historical data is a column vector, which will then be used to generate a pytorch tensor consisting of sequences. This data is split into training and testing tensors for LSTM model training.
- Multivariate modelling, i.e. feature data and historical price data are both used
  The historical price data will be treated as labels and some selected data, such as the trading volumes and FTSE 100, are treated as features. The difference between this scenario and the above one is we need two tensors, one for the feature sequence data and the other for the label sequence data.

3.5.2 Network Architecture

A multilayered LSTM architecture was implemented for our training model, where we call the LSTM mechanism from pytorch in our own LSTM class definition and training process. Like ordinary deep network computation, it includes both the forward and backward passes.

3.5.3 Optimisation

We have employed the ADAM (Adaptive Moment Estimation) optimiser, which is a popular stochastic gradient descent (SGD) optimisation algorithm commonly used in deep learning [23]. It maintains an adaptive learning rate for each weight parameter based on the historical first and second moment estimates of the gradients. ADAM is known for its fast convergence and robustness to noisy gradients.
As the prediction accuracy is simply the difference between the real ones and the predicted prices, the mean square error metric has naturally been adopted to compute the loss for the model.

3.5.4 Single Step vs Multi-Step Prediction

Most published work is only concerned with the prediction of the stock price for the next data point, i.e. it is a single step prediction ([14]). Although it helps the insight into the stock trend, in practical terms, the benefits are limited. There are a few research publications discussed about multiple step prediction [17], we have not managed to get hold of any useable code.
In our development, we have produced both single step prediction and multi-step prediction models. For the latter, we can think of two possible strategies:

- Extend the single step by adding the predicted price data to form a new sequence, which is then used to predict the next data point. Repeat the process until the needed number of steps is reached;
- Incorporate the future predictions as new label data, i.e. add a new column for each future step, which are used in training the model.

As discussed in ref[23], the accuracy of the first method deteriorates very quickly due to the issue of error compounding making it literally unuseable. The second method, on the other hand, offers much more accurate predictions, as the future step predictions form an integral part of the training process, which is implemented in our project.

3.5.5 LSTM Prediction Results

The following shows the overall results for predictions for the next day, and the predictions for one week in advance using the LSTM model. See Summary Results for univariate and multivariate versions for more predictions, up to 3 weeks (15 working days) in advance.

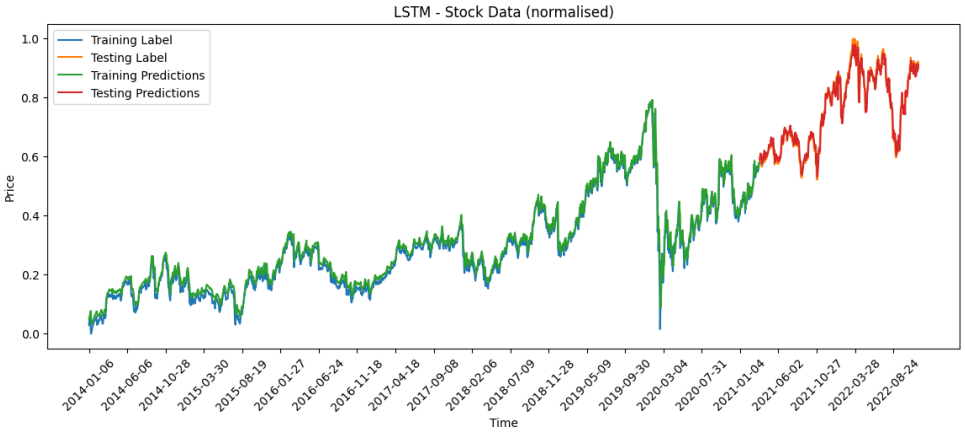

Figure 14: LSTM predictions, univariate (1 day ahead)

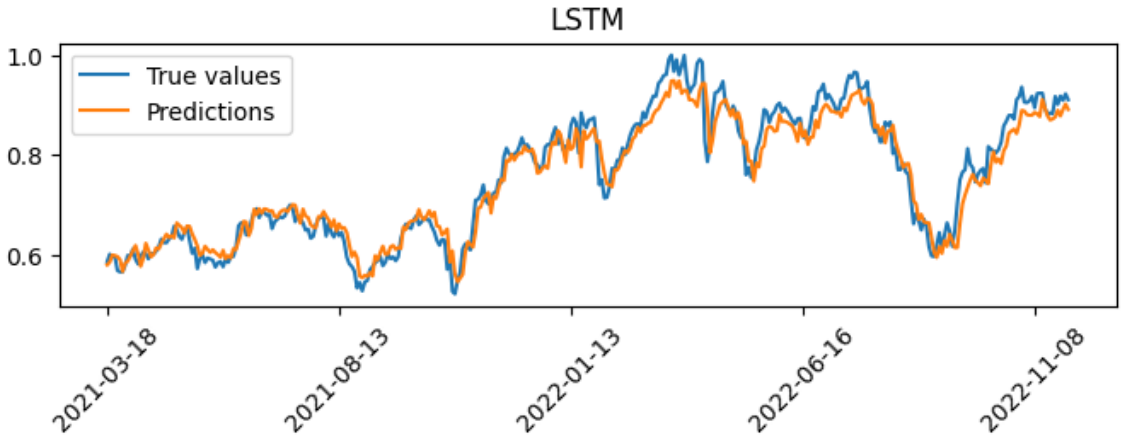

Figure 15: Zoom-in-view: LSTM predictions, multivariate (1 day ahead)

The next three images show the predictions up to 3 weeks in advance. It is clear that with multi-day prediction, the accuracy decreases. Despite the decrease, the accuracy still holds well, with a test

RMSE from 0.032 to 0.062 to 0.083 to 0.090.

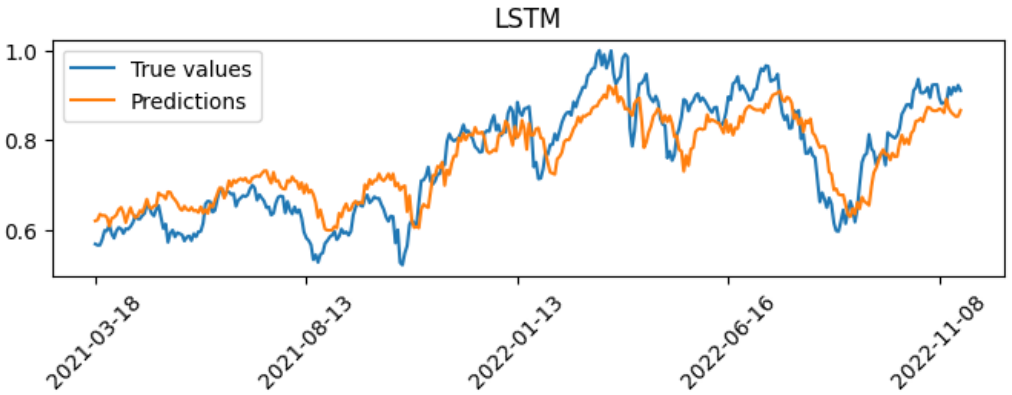

Figure 16: Zoom-in-view: LSTM predictions, multivariate (1 week ahead)

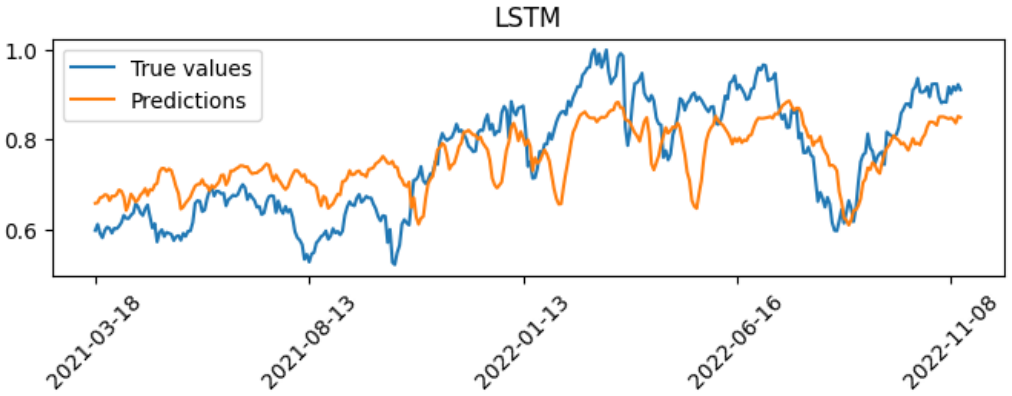

Figure 17: Zoom-in-view: LSTM predictions, multivariate (2 weeks ahead)

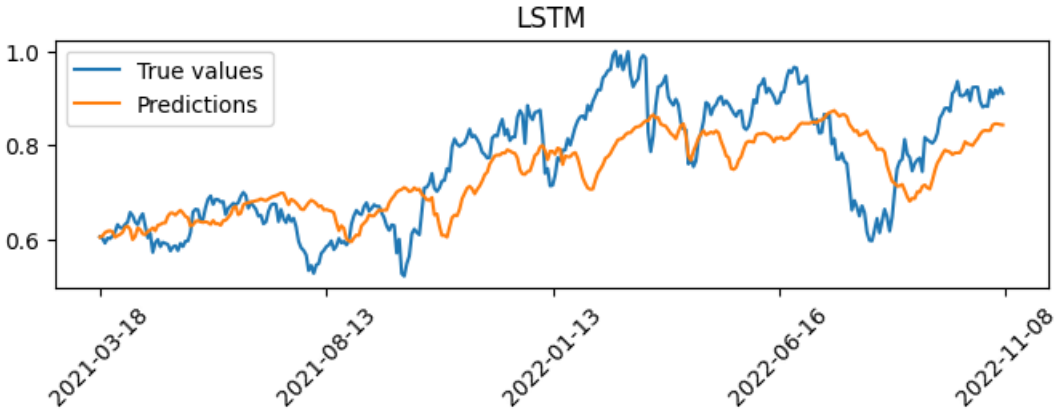

Figure 18: Zoom-in-view: LSTM predictions, multivariate (3 weeks ahead)

3.6 GRU

Although the internal structure of GRU is different from that of the LSTM, from the implementation point of view, the process is very similar. In our developments, the steps include:

- input sequence preparation for univariate and multivariate models
- network architecture, where we implemented our GRU class definition and training process making use of the pytorch GRU mechanisms
- optimisation: the ADAM optimiser is used
- future prediction steps: both single step and multiple step predictions are included

3.6.1 GRU Prediction Results

The following shows the overall results for predictions for the next day using the GRU model. Its overall performance is very similar to that of the LSTM, although for long term prediction, GRU performs slightly better. In the interest of space, the multi-step predictions are omitted from this section. The test RMSE for 1 day, 1 week, 2 weeks and 3 weeks are 0.036, 0.072, 0.083 and 0.085, respectively. See the Summary of Results section for comparisons.

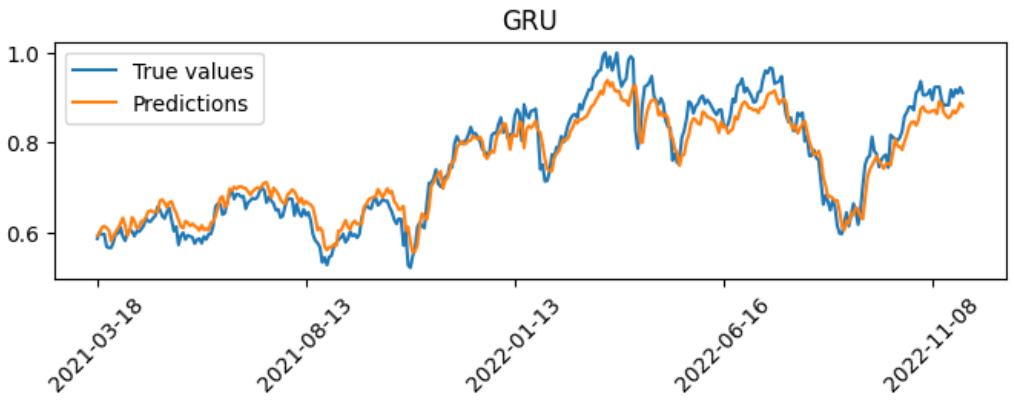

Figure 19: Zoom-in-view: GRU predictions, multivariate (1 day ahead)

3.7 Model Refinement for Improved Performance

3.7.1 Loss Metrics Selection

The most common metrics for training the models are mean squared error (MSE), mean absolute error (MAE) and root mean squared error (RMSE). MAE measures the average absolute difference between the predicted and actual values without considering their directions, and therefore are less sensitive to outliers than the other two metrics. MSE measures the average of the squared differences between the predicted and actual values. It is sensitive to outliers and can result in a larger penalty for larger errors. RMSE is similar to MSE, but after the squared root operation, the scale is in line with the predicted values.
We are interested not only in predicting the next step price, but more importantly are keen to predict stock prices multiple steps ahead, which is likely to have larger prediction errors. We therefore have selected the MSE as the training metric and used the RMSE to judge the testing accuracy.

3.7.2 Feature Selection

As discussed earlier we have collected a large number of potential feature data, ranging from macroeconomics data, e.g. the GDP, indices to fundamental stock data. To understand the relationship between these potential feature data with the stock concerned, we have produced a correlation heatmap which we have used to guide us on feature selection for stock predictions.
Having features helps us to extend univariate modelling to multivariate modelling, which is an essential step forward and is potentially able to offer higher prediction accuracy. However, the selection of most effective features is time-consuming and requires a lot of trial and error.
It is generally a good idea to include those features that are closely correlated with the predicted stock ([8]). This is because the models we use (LSTM and GRU) can learn the complex relationships between the input features and the output, and improve its ability to make accurate predictions. However, it is also important to consider the risk of overfitting, which can occur if the model is trained on features that are too closely related to the output. To strike a good balance, our feature selection strategy is to include those data which are relatively diversified in terms of correlation scale with the stock concerned. We have therefore included those that are closely correlated, e.g. open prices, previous day close prices and moving averages; those which are less closely related, e.g. trading volumes; those from the market competitors; and those which are generic, e.g. GDP.
To research on this issue comprehensively and decide which features should be included will require efforts for thoroughly testing the different combinations – this will be regarded as future work.

3.7.3 Hyperparameter Tuning

In the implementation of both LSTM and GRU, we have included a comprehensive set of hyperparameters, such as epoch number, batch size, number of layers, number of hidden units, learning rate, dropout probability and sequence length. There are several parameter tuning strategies, from the simple grid search to gradient based optimisation. Due to the time available for this project, we have decided to resort for the brute force grid search method.
During hyperparameter tuning, the whole dataset is split into three parts: training set, validation set, and test set. The training set is to fit the model on a range of hyperparameter combinations. The validation set is then used to evaluate the performance of the model for the combinations of hyperparameters. Once the tuning process is complete, the test set is used to evaluate the performance of the final model, which was trained on the combined training and validation sets. The ratio of the dataset split in our implementation is 64% for training, 16% validation, and 20% testing.
In order to develop an insight into the role of some individual hyperparameters and to save development time, some parameters are tested individually, such as the sequence length and epoch number.

4 Summary of Results

In the above, we have presented a selection of prediction results with individual methods. Here we briefly summarise their performances by showing the numerical comparisons.
The hyperparameters used for the predictions are:
Sequence length: 20, epoch number: 15, batch size: 32, dropout probability: 0.0, learning rate: 0.01, hidden units number: 32, number of hidden layers: 2
Table 2 shows the test RMSE of single step predictions with all the developed methods.

	ARIMA	**MLP**	**LSTM**	**GRU**
Univariate	7.458	0.014	0.024	0.027
Multivariate	N/A	N/A	0.036	0.032

Table 2: Test RMSE of various models for 1 day prediction

Table 3 summarises the test RMSE comparisons of single step vs multi-step and univariate vs multivariate predictions for LSTM and GRU models. Figure 20 shows this information visually. Overall,

we see that multivariate performs better than univariate, and GRU performs better than LSTM. Note that predictions become significantly less accurate if we extend the forecasting beyond 3 weeks.

	1 day	**1 week**	**2 weeks**	**3 weeks**
LSTM univariate	0.024	0.061	0.121	0.179
LSTM multivariate	0.032	0.062	0.083	0.090
GRU univariate	0.027	0.088	0.146	0.163
GRU multivariate	0.036	0.072	0.083	0.085

Table 3: Test RMSE for LSTM vs GRU for multiple step forecasting (Coca-Cola)

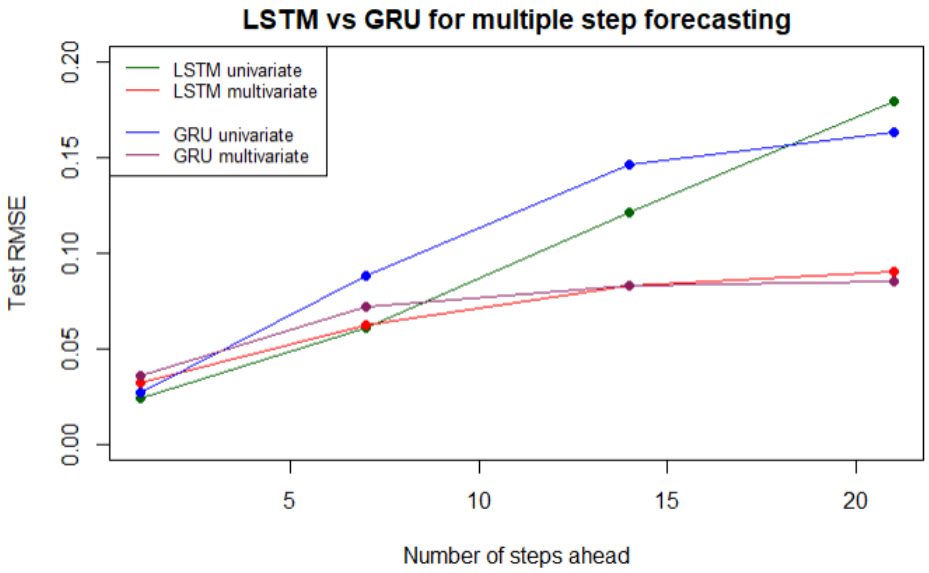

Figure 20: LSTM vs GRU for multiple step forecasting

We also have applied our methods to the prediction of a few other stocks to test the validity of our developments. Table 4 presents the test RMSE values of four stocks, where the predictions were computed using our multivariate GRU model.

	1 week	**2 weeks**
Coca-Cola	0.072	0.083
Apple 3Y	0.063	0.101
NYSE 3Y	0.065	0.085
Ballie Gifford Fund 5Y	0.043	0.087

Table 4: Test RMSE with multivariate GRU model for multiple step forecasting (different stocks)

From these comparisons, we can conclude:

- Both LSTM and GRU perform much better than AMIRA and MLP, especially for long term future predictions
- Univariate models perform similarly to multivariate ones for short term future predictions
- Multivariate models perform much better than single variate models for long term future predictions

- Prediction accuracy decreases with the increase of the number of steps ahead, which confirms our expectation
- Although our methods were developed initially on the Coca-Cola dataset, they also work well with other stocks.

A note on scaling
As our purpose is to evaluate the prediction accuracy for any stock data, we only visualise and present the predicted values in the normalised scale, which makes all predictions comparable regardless of the original units or quantity. However, if in a case where the true value is needed, it is very simple to convert the predicted normalised value to the real stock prices [23].

5 Closing Remarks

5.1 Conclusion

In this project we have developed several stock prediction techniques, including MLP, ARIMA, LSTM and GRU. The test results show that both LSTM and GRU perform significantly better than the other methods.

We have implemented both univariate and multivariate models. For the former, the historical data sequences are used as the input for the neural network model and for the latter, feature sequence data are used. Generally speaking, including multiple features in a prediction model can provide more information and context for the model to make accurate predictions. It allows the model to consider the underlying patterns in the data. Our test results confirmed this conclusion, which is especially true for longer term future (multi-step-ahead) predictions.

We have managed to incorporate many hyperparameters into our models, which prove very important in producing accurate predictions. In addition, we have also implemented the tuning strategy for the hyperparameters, automatically computing the best combination of the hyperparameters.

We are very pleased to be able to implement the multi-step-ahead prediction techniques, while most of the published work can only predict the next data point. The ability to predict a stock for a long term future is essential for traders to make decisions with confidence. We are very happy that we are able to predict stock prices 3-weeks ahead with a relatively good accuracy.

Finally, through the development of this project, we have managed to put what we learned in classroom to practice. It has extended and consolidated our knowledge in AI techniques and has given us confidence in applying it in our future work.

5.2 Limitations and Future Work

Our future work will aim to improve prediction accuracy mainly for the long term future. On this point, some topics are particularly worth exploring, including:

- comprehensive hyperparamter tuning. Due to the time limitation, we implemented the simple grid search strategy. In the future some more sophisticated techniques should be employed.
- feature data selection. This is an important topic and different combination of features will result in different performances. In the future, we will collect and analyse more financial data.
- testing the developed methods to a wide range of stocks and stock types. Although we have successfully applied the developed techniques to several stocks showing promising outcomes, it would be a good to apply it to a wider and more diversified group of other stocks.

To further enhance the accuracy of our stock price prediction model, we suggest comparing our current RNN-based model's performances against other deep learning models such as Generative Adversarial Networks (GANs). There have been increasing studies in the application of GANs on stock data[24] [11] , but the results have been inconsistent. We could thus explore with different models such as LSTM as the discriminator and Convolution Neural Network (CNN) or MLP as the generator to identify the most optimal model that generates the most accurate stock data predictions.

In addition, given the increasing importance of news and social media in influencing stock market, the inclusion of Natural Language Processing (NLP) has gained attention from researchers[25] [19]. By analysing news articles and Twitter sentiment related to the stock, we could incorporate market sentiment as a predictor for stock price, which could potentially enhance our prediction accuracy, allowing us to build a more robust and comprehensive predictive model.

5.3 Statement about Individual Contributions

We agreed the project contribution was equally split. 54588 worked on the data collection and pre-processing, feature correlation analysis, as well as both the Linear Regression and ARIMA models. 56019 worked on the MLP, LSTM and GRU models, including univariate and multivariate development, as well as single-step and multi-step forecasting models.

6 References

References

[1] Adebiyi A Ariyo, Adewumi O Adewumi, and Charles K Ayo. Stock price prediction using the arima model. In *2014 UKSim-AMSS 16th international conference on computer modelling and simulation*, pages 106–112. IEEE, 2014.

[2] George S Atsalakis and Kimon P Valavanis. Forecasting stock market short-term trends using a neuro-fuzzy based methodology. *Expert systems with Applications*, 36(7):10696–10707, 2009.

[3] Trevor S Breusch and Adrian R Pagan. A simple test for heteroscedasticity and random coefficient variation. *Econometrica: Journal of the econometric society*, pages 1287–1294, 1979.

[4] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[5] J. Contreras, R. Espinola, F.J. Nogales, and A.J. Conejo. Arima models to predict next-day electricity prices. *IEEE Transactions on Power Systems*, 18(3):1014–1020, 2003.

[6] Deumig. Stanford-project-predicting-stock-prices-using-a-lstm-network, 2020.

[7] Arul Earnest, Mark I Chen, Donald Ng, and Leo Yee Sin. Using autoregressive integrated moving average (arima) models to predict and monitor the number of beds occupied during a sars outbreak in a tertiary hospital in singapore. *BMC Health Services Research*, 5(1):1–8, 2005.

[8] Mark A Hall. *Correlation-based feature selection for machine learning*. PhD thesis, The University of Waikato, 1999.

[9] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[10] Zexin Hu, Yiqi Zhao, and Matloob Khushi. A survey of forex and stock price prediction using deep learning. *Applied System Innovation*, 4(1):9, 2021.

[11] HungChun Lin, Chen Chen, GaoFeng Huang, and Amir Jafari. Stock price prediction using generative adversarial networks. *Journal of Computer Science*, 17(3):188–196, Apr 2021.

[12] Robert Nau. Notes on nonseasonal arima models, n.d. Accessed: 2023-04-23.

[13] Neural_Line. A tutorial video, 2021.

[14] Mahla Nikou, Gholamreza Mansourfar, and Jamshid Bagherzadeh. Stock price prediction using deep learning algorithm and its comparison with machine learning algorithms. *Intelligent Systems in Accounting, Finance and Management*, 26(4):164–174, 2019.

[15] Muskaan Pirani, Paurav Thakkar, Pranay Jivrani, Mohammed Husain Bohara, and Dweepna Garg. A comparative analysis of arima, gru, lstm and bilstm on financial time series forecasting. In *2022 IEEE International Conference on Distributed Computing and Electrical Circuits and Electronics (ICDCECE)*, pages 1–6. IEEE, 2022.

[16] J Patrick Royston. Some techniques for assessing multivarate normality based on the shapiro-wilk w. *Journal of the Royal Statistical Society Series C: Applied Statistics*, 32(2):121–133, 1983.

[17] Swapnil Shelake, Rishikesh Kondavathini, Mahedin Ansari, Punit Sonwadekar, Sunny Kumar, Prerna Goswami, and Faruk Kazi. Forecasting enhancement of wind power generation using adversarial networks: A data driven approach. In *2022 IEEE PES 14th Asia-Pacific Power and Energy Engineering Conference (APPEEC)*, pages 1–6. IEEE, 2022.

[18] Sima Siami-Namini, Neda Tavakoli, and Akbar Siami Namin. A comparison of arima and lstm in forecasting time series. In *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 1394–1401, 2018.

[19] Matheus Gomes Sousa, Kenzo Sakiyama, Lucas de Souza Rodrigues, Pedro Henrique Moraes, Eraldo Rezende Fernandes, and Edson Takashi Matsubara. Bert for stock market sentiment analysis. In *2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 1597–1601. IEEE, 2019.

[20] Gülden Kaya Uyanık and Neşe Güler. A study on multiple linear regression analysis. *Procedia-Social and Behavioral Sciences*, 106:234–240, 2013.

[21] Ju-Jie Wang, Jian-Zhou Wang, Zhe-George Zhang, and Shu-Po Guo. Stock index forecasting based on a hybrid model. *Omega*, 40(6):758–766, 2012.

[22] Peter T Yamak, Li Yujian, and Pius K Gadosey. A comparison between arima, lstm, and gru for time series forecasting. In *Proceedings of the 2019 2nd International Conference on Algorithms, Computing and Artificial Intelligence*, pages 49–55, 2019.

[23] Aston Zhang, Zachary C. Lipton, Mu Li, and Alexander J. Smola. Dive into deep learning. *arXiv preprint arXiv:2106.11342*, 2021.

[24] Kang Zhang, Guoqiang Zhong, Junyu Dong, Shengke Wang, and Yong Wang. Stock market prediction based on generative adversarial network. *Procedia computer science*, 147:400–406, 2019.

[25] Wenbin Zhang and Steven Skiena. Trading strategies to exploit blog and news sentiment. In *Proceedings of the international AAAI conference on web and social media*, volume 4, pages 375–378, 2010.