The University of Nottingham

UNITED KINGDOM · CHINA · MALAYSIA

Stock Market Prediction Application with Machine Learning Algorithms based on Financial Data: On Software Engineering Perspective

Submitted in May 2024, in partial fulfillment of the conditions of the award of the degrees B.Sc.

- **Ock Ju Won** -

School of Computer Science

Faculty of Science and Engineering

University of Nottingham

I hereby declare that this dissertation is all my own work, except as indicated in the text:

Signature ________________________

Date ____/____ /______

Acknowledgement

I would like to express my special thanks and respect to my supervisor Dr. Doreen Sim Ying Ying as well as the module convenor professor K.R. Selvaraj who has given me the excellent opportunity, insights, and knowledge to conduct this intriguing project, *Stock Market Prediction Application with Machine Learning Algorithms based on Financial Data: On Software Engineering Perspective.* Also, I would like to appreciate other lecturers and professors of the school since their knowledge and insights have profoundly increased my understanding of Machine learning and Artificial Intelligence which would consequently affect my future steps in the workspace and academic field as well.

Secondly, I would like to show my gratitude wholeheartedly to my family for offering warm advice and care during the project. Their assistance enormously supported me mentally and physically to successfully finish my work.

Lastly, I would like to acknowledge the School of Computer Science, Faculty of Science and Engineering, and the University of Nottingham Malaysia for providing me with incredible opportunities, including this final year project and the chance to meet amazing people along the way.

Abstract

The stock market is a system where currency, stocks, capital, and other financial products are bought and sold by individuals. To maximise the benefits for companies and investors, many individuals, especially traders, have utilised various mathematical models and analytical techniques to analyse stock prices, typically aiming to purchase stocks expected to increase in value in the future and avoid those anticipated to decline. Financial data is a typical example of Time Series Data. Time Series Forecasting (TSF) is concerned with predicting the distribution of future variables based on past observations, and adopting Deep learning models for the stock data can be conducted for the prediction and analysis of the datasets. Stock data includes challenges for accurate prediction due to the potential existence of high noise, non-stationary data, and non-linear tendency. So, throughout the project, the LSTM model has been implemented as the basic foundation since it is widely used for stock data prediction, and by customising the LSTM model with the CNN model it is available to compare the better option based on the predicted profits which have been calculated based on Sharpe Ratio. The CNN-LSTM and ARIMA models were adopted for this project to compare. Even though the CNN-LSTM has shown the most appropriate model for the project, the paper describes 1) the features and the reason why for selecting the CNN-LSTM model and 2) the experiments made with the constructions of the data preprocessing parts and the model itself.

Keywords: Deep Learning, Time Series Forecasting, CNN-LSTM, ARIMA, Regression Analysis

Table of Contents

Introduction

The stock market is a system where currency, stocks, capital, and other financial products are bought and sold by individuals. To maximise the benefits for companies and investors, many individuals, especially traders, have utilised various mathematical models and analytical techniques to analyse stock prices, typically aiming to purchase stocks expected to increase in value in the future and avoid those anticipated to decline.

Stock data is a typical example of Time Series Data. Time Series Forecasting (TSF) is concerned with predicting the distribution of future variables based on past observations. The models employed for this task can be broadly categorised into Traditional, Machine Learning, and Deep learning models.

While it is feasible to employ traditional models like the ARIMA model for predicting a specific company's stock price, stock data includes complexities such as high noise, non-stationarity, and non-linearity, making accurate prediction challenging. Furthermore, traditional models face several issues. First, it applies regression to a fixed set of factors from the most recent historical data to generate the predictions [1]. Secondly, traditional methods are iterative and are often sensitive to how the process is seeded [1]. Third, stationarity is a strict condition, and it is difficult to achieve stationarity of volatile time series by merely addressing drift, seasonality, autocorrelation, and heteroskedasticity [1]. This necessitates the use of Machine Learning models like RNN, LSTM, GRU, and Transformer. However, dilemmas exist with these models as well: 1) the prediction accuracy of RNN family models and Transformer significantly varies depending on the implementation of Time series (financial data) look-back window size and look-ahead window size, 2) longer look-ahead window sizes tend to favour Transformer model precision, shorter sizes demonstrate the efficiency of RNN family models [1]. Such findings reminded me that every model that was considered an appropriate model for forecasting time series data has potential limitations to predict accurately when the environments or inserted datasets are changed. The fact inspired me to sum up certain models and customise them to reinforce each model’s weaknesses and construct a certain model as a better prediction model.

In this project, the stock data from five leading American IT companies, FAANG, were crawled and employed to develop deep learning models and for analysis. Various financial technical indicators also were adopted based on the five data, which are: “Open, High, Low, Close”, for the stock price analysis. The visualisation tools were adopted to compare the best efficiency and accuracy upon the prediction of the data. For the comparison, the ARIMA model will be compared with the CNN-LSTM model which is a combination with CNN and LSTM models. The stock data has been retrieved from 2014-03-26 to 2024-03-26, and the training and testing datasets have been divided into ratios of 80% and 20%.

Literature Review

Several research papers have proposed certain algorithms to predict stock market data. To determine the use cases of these methods and their related advantages and disadvantages, I have conducted a literature review. Chen, Fushimi, et al. [2] employed an LSTM model to assess Intel's stock price and used the Sharpe ratio to compare the Buy & Hold strategy and the LSTM-based strategy. Pang, Zhou, Wang, et al. [3] compared RNN and LSTM models and conducted an experimental analysis to find that the LSTM with an Auto-Encoder module (AE-LSTM) correctly predicted most of the data. The Beijing Air Quality Dataset was analysed by Shi, Jain et al. in [1] using RNN, LSTM, GRU, and Transformer. They noted that the LSTM model's performance varies with the size of the window settings and that Transformers are useful for long-term prediction while LSTM and GRU outperform RNN for short-term forecasting.

Basic stock price data indicators, including opening price, high price, low price, closing price, and volume, were used by Liu, Liao, et al. in [4] to forecast the rise and fall of China's CSI 300 index between December 2016 and January 2017. They included the exponential moving average and the moving average of the closing price as input variables and converted each indicator into a relative ratio value compared to other indicators. The prediction model they employed was an LSTM network, and they tested with different numbers of layers to maximise performance. A three-layer LSTM network produced the best accuracy, exceeding 72%. However, there were limitations on the reliability of the results due to the small amount of test data.

Selvin, Menon, Soman, et al. experimented with three different deep learning models: CNN, RNN, and LSTM with the approach of sliding the window. [5] CNN outperformed the other two models regarding accuracy by using only the information in the current window, which allowed the model to understand the dynamic changes and patterns occurring within it. On the other hand, RNN and LSTM forecast instances in the future using data from past delays.

I could have certain insights from these papers regarding 1) the possibility of combining LSTM with other models for comparative analysis; 2) the selection of RNN, CNN, or GRU as possible customisation models for comparative analysis; 3) the inclusion of technical indicators during the data preprocessing for deep learning models, such as the moving average or exponential moving average of the closing price or other related stock price data; and 4) the importance of normalisation during data preprocessing; and 5) the significance of being aware of issues like overfitting during dataset construction. Based on these findings, I could create a machine learning model that will assist in determining if this new approach or the ARIMA model which has been used traditionally a lot for the TSF performs better and is more profitable when it comes to stock price prediction.

Statistical Analysis & Machine Learning Algorithms

Through this section, the basic concepts and algorithms for the implemented models of the project: ARIMA, CNN-LSTM, and related models, are explained. It is important to explain the fundamental concepts and methods to understand the reasons for combining CNN with LSTM. ARIMA as the project.

1. CNN(Convolutional Neural Network)

Convolutional Neural Network, inspired by the structure of biological visual neurons, was first introduced in a paper by LeCun in 1989 [6]. CNN maintains the spatial information of images during learning, showcasing remarkable performance in image and video processing. It is a predominant neural network used in the field of computer vision. CNN consists of two main parts: the Convolution Layer and the Pooling Layer. The Convolution Layer applies convolution operations on the input image to extract feature maps that effectively represent the image's characteristics. The Pooling Layer reduces the size of the feature maps by aggregating the values in a specified region, a process known as sub-sampling. Average Pooling and Max Pooling are considered major methods to be conducted. Figure 1 illustrates the examples of Average Pooling and Max Pooling.

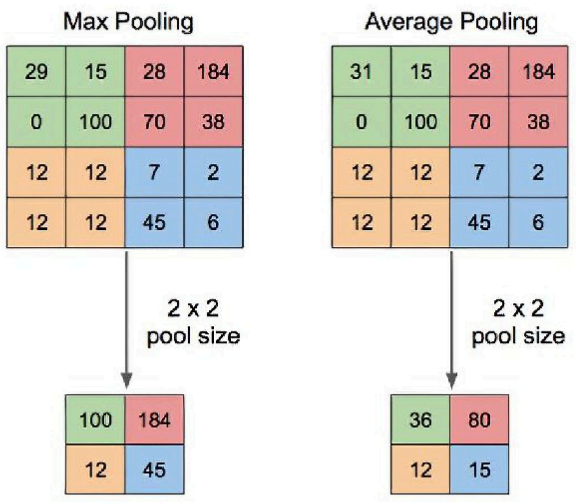


Figure 1. Illustration of Max Pooling and Average Pooling [7]

2. LSTM(Long Short Term Memory)

LSTM, derived from RNN (recurrent neural network), was proposed as an alternative to overcome the issues of gradient vanishing and exploding in RNNs [8]. According to research, LSTM captures the temporal characteristics (effects) of input data more effectively than other models and has been proven efficient in stock price prediction [9][10]. Figure 2 illustrates the structure of an RNN cell, and Figure 3 shows the structure of an LSTM cell. LSTMs have a more complex structure compared to RNNs, incorporating a Cell State in addition to the hidden state of the RNN. This modification allows for more sophisticated control of information from previous time steps.

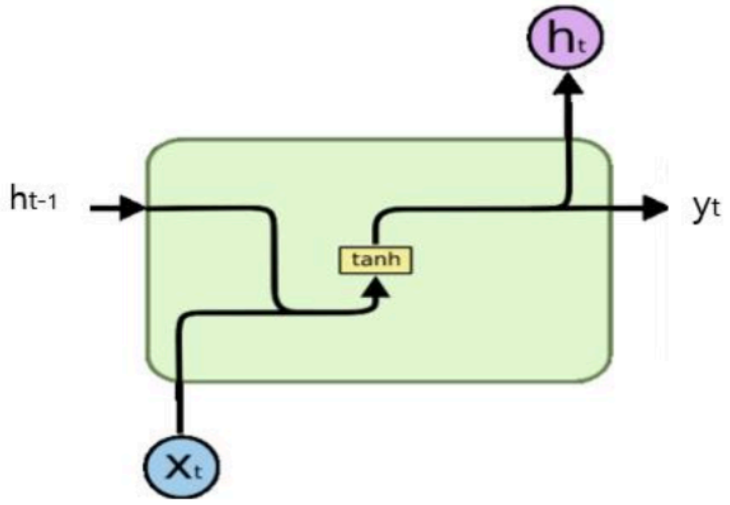


Figure 2. The architecture of the RNN Cell [11]

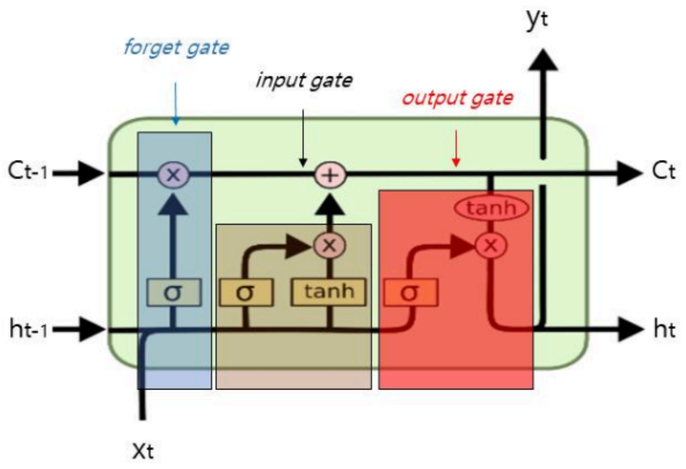


Figure 3. The architecture of the LSTM Cell [11]

Figure 3 represents a specific time step and explains the working mechanism of LSTM through the three gates displayed inside the cell. The forget gate takes the previous time step's hidden state and the current input: h_{t-1} and x_t . Subsequently, it processes them through a sigmoid activation function to output a value between 0 and 1. This process determines the extent to which the information h_{t-1}received from the previous cell should be forgotten. The operation process of the forget gate is as follows.

$$f_t = \sigma(W_f \cdot [h_t - 1, x_t] + b_f)$$

Subsequently, the left part of the input gate receives h_{t-1} and x_t, as inputs and determines how much of the new input information should be incorporated into the cell state through the sigmoid function. This process is represented as below.

$$i_t = \sigma(W_i \cdot [h_t - 1, x_t] + b_i)$$

In the right part of the input gate, h_{t-1} and x_t are received as inputs, and new input information to be reflected in the cell state is generated through the tanh function. This process is denoted as below.

$$g_t = tanh(W_c \cdot [h_{(t-1)}, x_t] + b_c)$$

Subsequently, the cell state is updated by combining the information on how much to forget and remember from the forget gate and the previous cell state. This is achieved by multiplying the forget gate's output with the previous cell state, thereby reflecting the information to be forgotten. Then, the updated information regarding the current input, as processed through the input gate, is added to this product to generate the current cell state. This process is denoted by the equation below.

$$C_t = f_t \cdot C_{t-1} + i_t \cdot g_t$$

Finally, the output gate undergoes two processes to determine the final output value. In the first process, the left side of the output gate employs a sigmoid function to decide the intensity of the cell state that will be propagated to the next time step.

$$o_t = \sigma(W_o \cdot [h_t - 1, x_t] + b_o)$$

In the second process of the output gate, the cell state derived from (6) is processed through a tanh function, and then the cell propagation intensity determined in (7) is multiplied by this result to generate the value that will be propagated to the next time step, $t + 1$. The process of generating this value is represented in the below equation.

$$h_t = o_t \cdot tanh(C_t)$$

3. CNN-LSTM(Convolutional Neural Network - Long Short Term Memory)

Although both CNN and LSTM algorithms are widely used independently, recent research has been exploring the integration of these two models into a combined CNN-LSTM architecture. The CNN model efficiently extracts features from input data, while the LSTM model effectively analyses time-series data. The CNN-LSTM model leverages the strengths and mitigates the weaknesses of each model, thereby potentially enhancing the performance beyond what each model could achieve on its own. Studies have shown that models combining CNN and LSTM exhibit superior predictive performance compared to standalone LSTM models [12]. Therefore, rather than implementing RNN or RNN to the LSTM model, it was decided to apply CNN to the LSTM model for the project. It will be assessed if this is considered a better model than the ARIMA model in future procedures.

4. ARIMA(Autoregressive Integrated Moving Average)

The ARIMA (AutoRegressive Integrated Moving Average) model is a prevalent statistical method in time series analysis. It was initially presented in a paper by Box and Jenkins in 1970 [13]. The model synthesises three principal components: Autoregression (AR), Integration (I), and Moving Average (MA). It incorporates a differencing process to convert non-stationary time series data into stationary data, thereby eliminating non-stationarities and rendering the data more amenable to prediction. The Autoregression (AR) component reflects the influence of past values on the current ones, while the Moving Average (MA) component accounts for the impact of past error terms. The Integration (I) process is implemented to normalise non-stationary time series data, utilising the differences between consecutive observations calculated at specific time intervals. A pivotal characteristic of the ARIMA model is its capacity to model the autocorrelations within time series data to forecast future values, proving beneficial for predictions in various domains like financial data, stock markets, and economic indicators. The ARIMA model can be succinctly represented with its mathematical formulation.

$$ARIMA(p,d,q) = AR(p) + I(d) + MA(q)$$

5. RNN(Recurrent Neural Network)

RNN is a type of artificial neural network that was proposed by Elman in 1990 [14]. Figure 4 illustrates the structure of an RNN network. In Figure 8, the values received at the input layer Xt are passed to the output layer while simultaneously becoming inputs to themselves again in a repetitive structure. This architecture allows for the learning of sequential information in the input data. Consequently, RNNs have been extensively utilised in areas where sequential information is crucial, such as early machine translation and stock price prediction. The study [15] conducted experiments using three types of artificial neural networks: PNN, TDNN, and RNN, for stock price prediction and found that the RNN performed the best. However, RNNs have limitations due to the vanishing gradient and exploding gradient problems that occur during the training process when dealing with long input sequences.

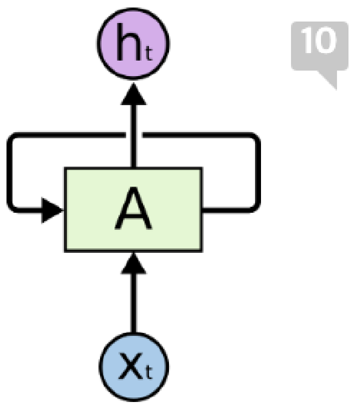


Figure 4. The architecture of the RNN [11]

Technical Indicators & Input Features

This section explains the technical indicators used in the finance filed for analysing stock data. Technical indicators examine historical and current price movements of stocks by utilising market data such as volatility, trading volume, and stock prices. These indicators are used to evaluate momentum, trends, and whether stocks are overbought or oversold. These indicators include, for instance, Bollinger Bands, RSI, and MACD. The fact that these tools can help investors anticipate short-term price movements and determine the best time to enter the market motivated me to consider all of these data as the major and important training features for the CNN-LSTM model.

Simple Moving Average (SMA) is a fundamental type of moving average that calculates the mean stock price over a predetermined period and plots these averages to create a trendline. This method helps smooth out price fluctuations, simplifying the process to detect clear trends. For example, to calculate a 20-day SMA, add up the closing prices of the stock for the previous 20 days, then divide the total by 20. Sequentially plotting these averages helps in showing medium-term pricing trends.

Exponential Moving Average (EMA) is a type of moving average that puts a greater weight on recent price data than on the older data. EMAs are especially helpful for monitoring short-term price trends in financial markets because of this weighting, which delivers a more responsive measure of the average price over aspecific period.

$$Smoothing\ Factor = \frac{2}{Period + 1}$$

$$EMA_{today} = (Price_{today} - EMA_{yesterday}) * Smoothing\ Factor + EMA_{yesterday}$$

Moving Average Convergence Divergence (MACD) is a trend-following indicator used in technical analysis that measures the price movement trends of stocks or other financial assets by utilising the relationship between two moving averages. MACD was implemented by Gerald Appel in the 1970s [16].

Relative Strength Index (RSI) is a widely used momentum oscillator in financial markets that is employed to measure the overbought and oversold conditions of an asset. This indicator was created by J. Welles Wilder Jr. in 1978 [17]. It evaluates the relative strength and velocity of price movements to help predict potential reversal points in the market.

$$RSI = 100 - \left(\frac{100}{1 + RS}\right)$$

$$RS = \frac{Average\ Loss\ over\ n\ periods}{Average\ Gain\ over\ n\ periods}$$

Average True Range (ATR) is a technical analysis tool used in financial markets such as stocks, foreign exchange, and commodities to measure the volatility of an asset's price. This indicator was developed by Welles Wilder Jr. and was first introduced in his 1978 book "New Concepts in Technical Trading Systems" [17]. ATR is particularly useful for identifying changes in market volatility and is commonly used for setting trade ranges and stopping losses.

$$ATR = \frac{1}{n} \sum_{i=1}^{n} TR_i$$

Bollinger Bands are a technical analysis tool for financial markets developed by John Bollinger in the early 1980s [18]. This indicator is used to measure the volatility of prices for various financial assets such as stocks, foreign exchange, and commodities, and to identify overbought or oversold conditions. Bollinger Bands are useful for evaluating price fluctuations and the power of trends, and they indicate relatively high or low price levels in the market.

Commodity Channel Index (CCI) is a technical analysis tool primarily used in financial markets. It was developed by Donald Lambert in 1987 to identify cyclical price movements in futures contracts [19]. This indicator measures the relationship between various asset prices, such as stocks, commodities, and foreign exchange, and their average prices to determine if an asset is in an overbought or oversold state. CCI is particularly useful for detecting extreme price variations and predicting possible changes in trends.

$$CCI = \frac{(TypicalPrice - MovingAverage)}{0.015 * MeanDeviation}$$

Momentum Indicator (MOM) is a technical analysis tool used in financial markets to measure the speed or strength of price movements of stocks or other assets. This indicator tracks the price fluctuations of stocks or other financial assets over time, quantifying the rise or fall in prices during a specific period. The Momentum Indicator is primarily used to predict if market trends will continue or reverse.

$$Momentum = Current\ Closing\ Price - Closing\ Price\ n\ periods\ ago$$

Rate of Change (ROC) is a technical analysis tool used to measure the rate of price fluctuations of stocks or other financial assets. This indicator expresses the change in price over a specific period as a percentage and is used to indicate the momentum or speed of an asset's price. ROC is primarily useful for assessing the strength of market trends and identifying potential trend reversal points.

$$ROC = \frac{Current\ Closing\ Price - Closing\ Price\ n\ periods\ ago}{Closing\ Price\ n\ periods\ ago} * 100$$

Williams %R (WILLR) is a momentum-based technical analysis indicator developed by Larry Williams [20]. This indicator expresses the current position of a stock or other financial asset's price relative to the high and low points over a given period as a percentage. It is used to assess whether an asset is in an overbought or oversold state.

$$Williams\ \%R = \frac{Highest\ High - Current\ Close}{Highest\ High - Lowest\ Low} * -100$$

Raw Stochastic Value (RSV) is an indicator used as a component of the Stochastic Oscillator. It is commonly used in various financial markets such as stocks, commodities, and forex. RSV evaluates the position of an asset's price relative to its price range over recent n periods, aiding in the assessment of overbought or oversold conditions.

$$RSV = \frac{Current\ Close - Lowest\ Low\ of\ n\ periods}{Highest\ High\ of\ n\ periods - Lowest\ Low\ of\ n\ periods} * 100$$

Relative Strength Accumulation/Distribution (RSAD) is a less common technical analysis tool that serves as an accumulation or distribution indicator based on the relative strength of a specific asset. This indicator is primarily used to predict whether an asset's price movements will continue along the existing trend or reverse. RSAD assesses the balance between buying and selling power in the market and is particularly effective in analysing accumulation and distribution conditions on high-volume trading days.

Fourier Transform is a mathematical tool that decomposes a signal varying over time or space into components with various frequencies for analysis. In signal processing, applied mathematics, engineering, and physics, this transformation is essential. Signals can be converted from the time domain to the frequency domain using the Fourier Transform. This feature makes it useful for analysing the frequency components of stock prices, which gives analysts and traders insight into the market's cyclical tendencies. This can be especially helpful in finding and taking advantage of patterns that might not be immediately visible in the time domain data by itself [21][22]. Any continuous signal (function) can be represented using the

Fourier Transform fundamentally as the sum of trigonometric functions (the sine and cosine functions), which have the fundamental frequency and integer multiples. The Fourier Transform can be used to determine which features of a signal correspond to particular frequencies.

$$F(\omega) = \int_{-\infty}^{\infty} f(t) \cdot e^{-i\omega t} dt$$

f(t) is the original signal in the time domain. ω represents the angular frequency of each frequency component. $e^{-i\omega t}$ is a complex exponential function, which encompasses both sine and cosine functions.

Each of these indicators has been coded using Python's built-in libraries, including numpy. In addition to this, data such as percent change or return values are also included, but since they are not considered necessary to explain from an economic or mathematical perspective, those variables will be discussed in the part describing data preprocessing.

Experiment

This section introduces the experimental process for a comparative analysis between the CNN-LSTM model and the ARIMA model. Part 1 explains how to crawl data from FAANG stocks, while Parts 2 and 3 introduce the preprocessing steps for the training datasets for the CNN-LSTM and ARIMA models, respectively. Parts 4 and 5 then present the construction and training processes for each model based on the preprocessed datasets.

1. Data collection for models

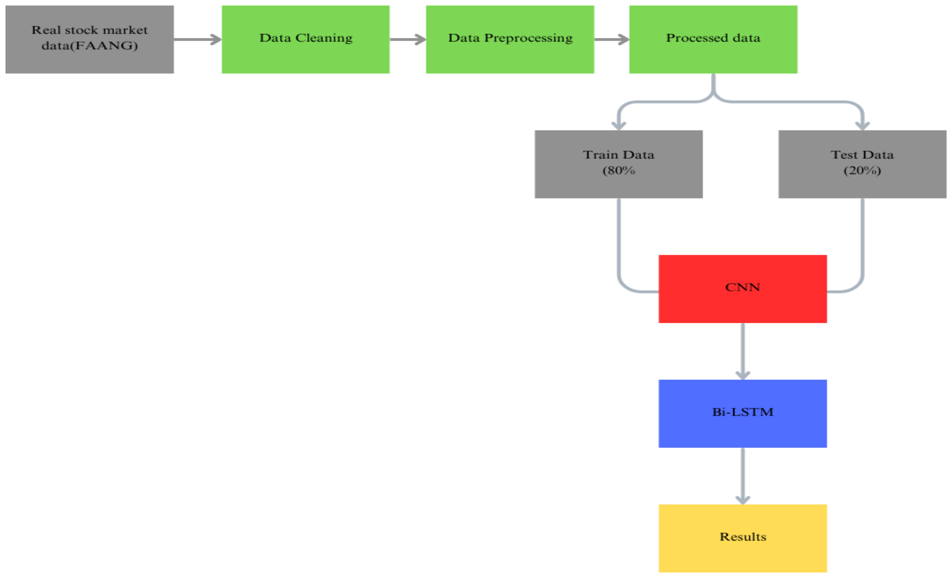


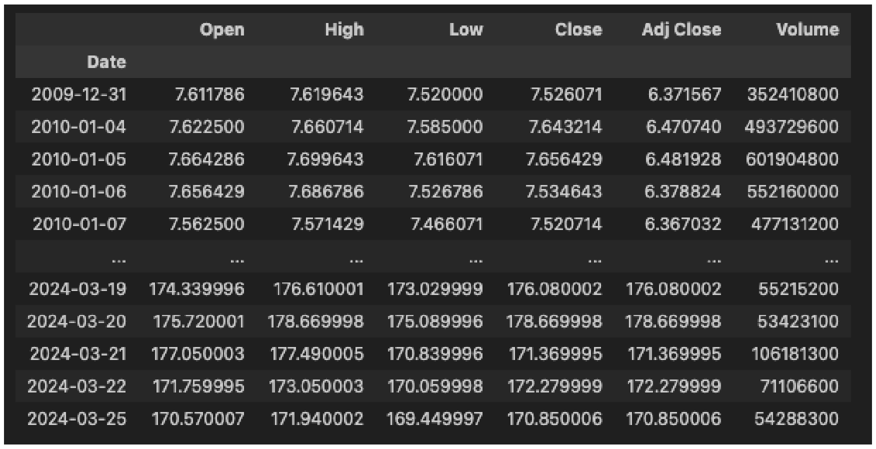
Figure 5. The architecture for the CNN-LSTM model

Before starting to build the models, it is required to collect the stock price data as the first step. For the data collection process, the Yahoo Finance API was used. The Python script was developed based on the API to crawl and retrieve data for the FAANG(Facebook, Apple, Amazon, Netflix, and Google) using their stock tickers. For the whole experiment, Apple stock data was manipulated for measurement equivalence. The data was collected for the period from December 31, 2009, to March 26, 2024. The collected data was stored in a database for processing, and the database used was MySQL. As shown in Table 1, the elements of the collected data are as indicated in Table 1. Figure 6 shows the results of retrieving data for Apple stock, one of the FAANG companies, for approximately 15 years.

Columns' name	***Contents***
Date	Date of trading
Open	Open Price
High	High Price
Low	Low Price
Close	Close Price
Adj Close	Adjusted Close Price
Volume	Trading Volume

Table 1. The table of crawled stock price data

The structure of the crawled dataset can be described below:



```
                  Open        High         Low       Close   Adj Close     Volume
      Date
2009-12-31    7.611786    7.619643    7.520000    7.526071    6.371567  352410800
2010-01-04    7.622500    7.660714    7.585000    7.643214    6.470740  493729600
2010-01-05    7.664286    7.699643    7.616071    7.656429    6.481928  601904800
2010-01-06    7.656429    7.686786    7.526786    7.534643    6.378824  552160000
2010-01-07    7.562500    7.571429    7.466071    7.520714    6.367032  477131200
       ...         ...         ...         ...         ...         ...        ...
2024-03-19  174.339996  176.610001  173.029999  176.080002  176.080002   55215200
2024-03-20  175.720001  178.669998  175.089996  178.669998  178.669998   53423100
2024-03-21  177.050003  177.490005  170.839996  171.369995  171.369995  106181300
2024-03-22  171.759995  173.050003  170.059998  172.279999  172.279999   71106600
2024-03-25  170.570007  171.940002  169.449997  170.850006  170.850006   54288300
```

Figure 6. Example of a stock market dataset of Apple

2. Data preprocessing for the CNN-LSTM model

Constructing a dataset for training the CNN-LSTM model is more important than simply configuring the model architecture. It involves not just using stock price data but 1) building a more sophisticated dataset and 2) conducting effective regression analysis on close data. As previously explained in the Technical Indicators part, various indicators used in financial portfolios analysing numerous stock prices were mathematically implemented using the numpy library and Python’s built-in libraries. The final dataset includes columns reflecting technical indicators and return values used as benchmarks for comparing the performance of the models, totalling 45 columns.

However, in the CNN-LSTM model, the primary column for analysis is the closing price data. To prevent overfitting and improve the model's performance, the feature selection process in the dataset was considered essential. Exploratory Data Analysis (EDA) was conducted to analyse the correlations between the data. The EDA results, showing the correlation coefficients for all 45 columns, are presented in Figure 7.

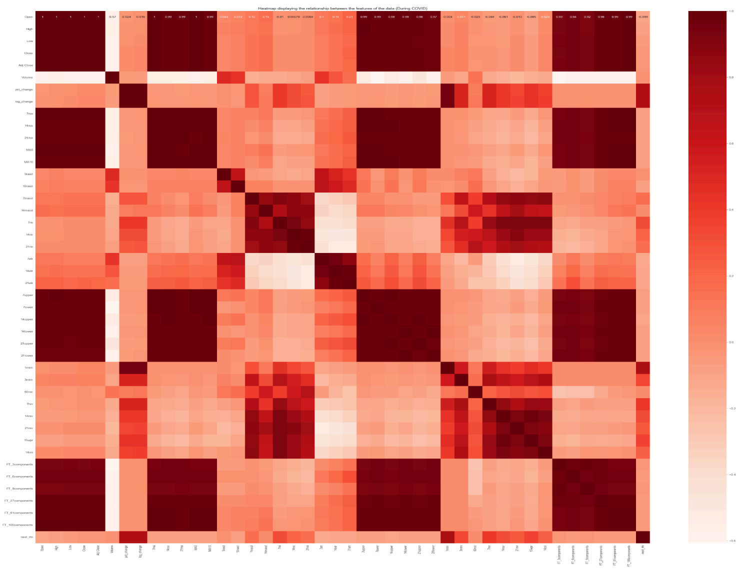


Figure 7. The EDA result of the technical indicators of the stock data

In the deep learning process, analysing and using correlation coefficients is primarily used to understand the interrelationships between input features or to comprehend activation patterns within the network. Input features with very low correlation coefficients may be independent or have very weak relationships. Excluding these features from the model can be beneficial for several reasons: 1) When dealing with high-dimensional data, features with low correlation may provide little or no useful information. Removing these features can reduce model complexity and shorten training time. 2) Removing irrelevant input features can help prevent the model from overfitting the training data, aiding in better generalisation. 3) Features with low correlation coefficients can sometimes represent noise. By removing these features, the model can focus on more important signals, thereby improving performance [23]. Therefore, an EDA centred around close data was conducted, and all data with correlation coefficients less than 0.5 were removed. The remaining data consisted of 28 columns. The EDA results focused on close data are presented in Figure 8.

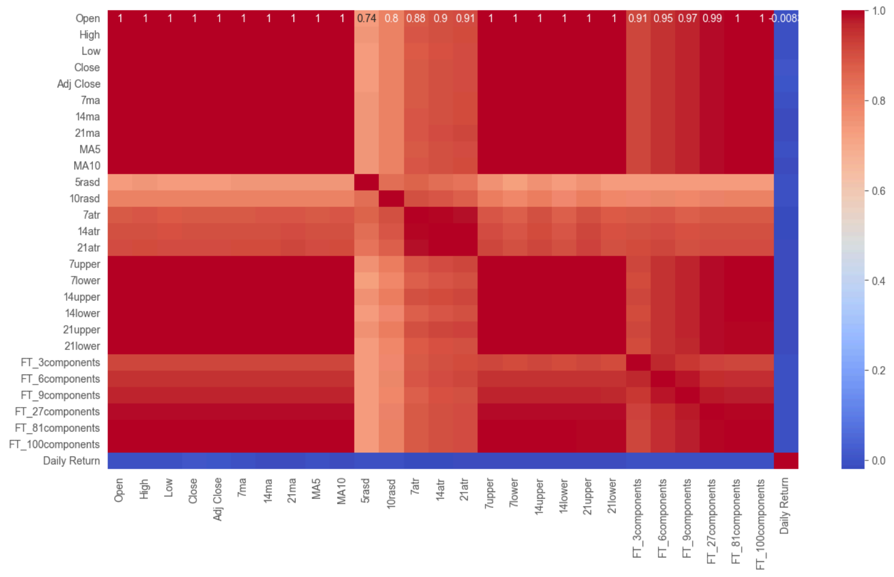


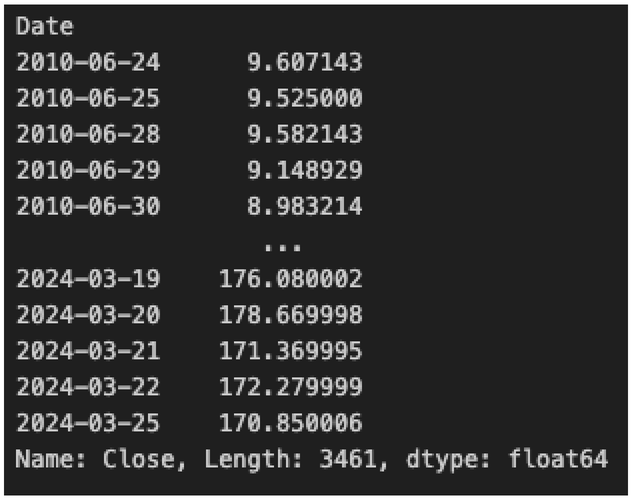
Figure 8. The EDA result after the data cleaning process

After the EDA process, the data cleaning process was conducted for the CNN-LSTM model construction. The cleaning part was considered important since it dealt with the NaN values or the outliers in the crawled data which have the potential to pollute the data analysis process. The crawled data was cleansed by using the Pandas library to remove any NULL values present. All of them are based on the stock price data and are relevant. However, since the analysis focused on the stock's Close data, only indicators related to Close data were selected to be used.

It was discovered during the data preprocessing part that the CNN model always takes 2-dimensional and 3-dimensional arrays into account while training and selecting the required features. However, for the project, the data was formed of 1-dimensional arrays and was used as the training data. It was determined to create a function that converts 1-D arrays to [100,1] tensors to appropriately pare the dataset to the CNN model. A type of data structure called Tensors is used to represent a multilinear relationship between a group of objects in a vector space. [24] Therefore, every 100 rows were selected to convert a 1-D array to a tensor. The whole dataset of each FAANG's data was processed in this manner. The "Close" column was converted for the project because it is the main column that would be examined and used to forecast the stock data. Tensors for the CNN side of the model to train could be obtained after this process. Subsequently, 80% of the entire data set was assigned for training and 20% for testing. To send the data to the training models' phase, I eventually reshaped and organised it.

3. Data preprocessing for the ARIMA model

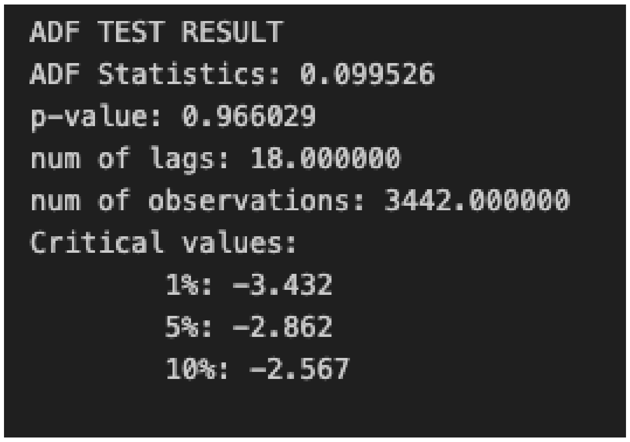
The ARIMA model's data preprocessing and loading processes were technically similar to CNN-LSTM, but unlike the CNN-LSTM model, no additional technical indicators were incorporated. The regression analysis within the model focuses on the stock's closing date, and both training and testing models were split in an 8:2 ratio from the same data period. Figure 9 shows the amended dataset for the ARIMA model.



```
Date
2010-06-24      9.607143
2010-06-25      9.525000
2010-06-28      9.582143
2010-06-29      9.148929
2010-06-30      8.983214
                  ...
2024-03-19    176.080002
2024-03-20    178.669998
2024-03-21    171.369995
2024-03-22    172.279999
2024-03-25    170.850006
Name: Close, Length: 3461, dtype: float64
```

Figure 9. The Dataset for the ARIMA model

ARIMA considers both Autoregressive and Moving Average components and uses differencing between observations to transform non-stationary data. Therefore, the ADF test was required to employ for checking the stationarity of the time series data and determine the differencing order “d”. By using the adfuller library, it was successful in conducting the ADF test. The following Figure 10 shows that the ADF test of the stock data is impossible to reject the null hypothesis since the p-value is more than 0.05. So, it was required to set the input dataset as stationary. The conduct was done with Python’s built-in functions.



```
ADF TEST RESULT
ADF Statistics: 0.099526
p-value: 0.966029
num of lags: 18.000000
num of observations: 3442.000000
Critical values:
        1%: -3.432
        5%: -2.862
        10%: -2.567
```

Figure 10: Augmented Dickey-Fuller(ADF) test result of the Apple data

4. Model Construction & Training for CNN-LSTM model

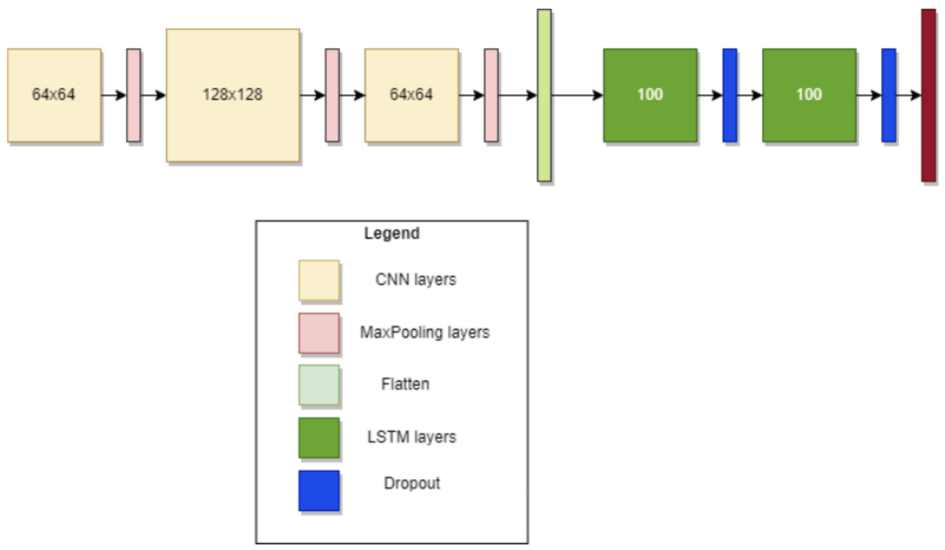
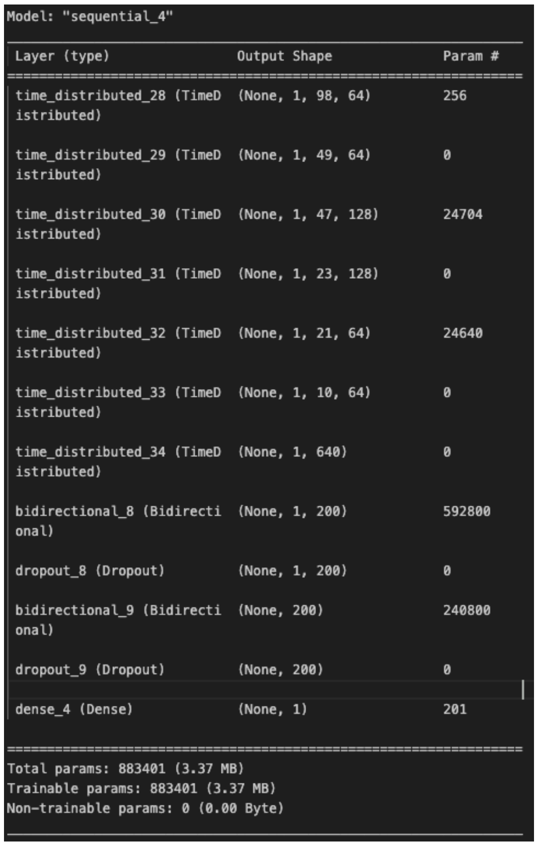


Figure 11. The brief architecture of the CNN-LSTM model



```
Model: "sequential_4"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 time_distributed_28 (TimeD  (None, 1, 98, 64)         256
 istributed)

 time_distributed_29 (TimeD  (None, 1, 49, 64)         0
 istributed)

 time_distributed_30 (TimeD  (None, 1, 47, 128)        24704
 istributed)

 time_distributed_31 (TimeD  (None, 1, 23, 128)        0
 istributed)

 time_distributed_32 (TimeD  (None, 1, 21, 64)         24640
 istributed)

 time_distributed_33 (TimeD  (None, 1, 10, 64)         0
 istributed)

 time_distributed_34 (TimeD  (None, 1, 640)            0
 istributed)

 bidirectional_8 (Bidirecti  (None, 1, 200)            592800
 onal)

 dropout_8 (Dropout)         (None, 1, 200)            0

 bidirectional_9 (Bidirecti  (None, 200)               240800
 onal)

 dropout_9 (Dropout)         (None, 200)               0

 dense_4 (Dense)             (None, 1)                 201

=================================================================
Total params: 883401 (3.37 MB)
Trainable params: 883401 (3.37 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Figure 12. The summarised architecture of the CNN-LSTM model

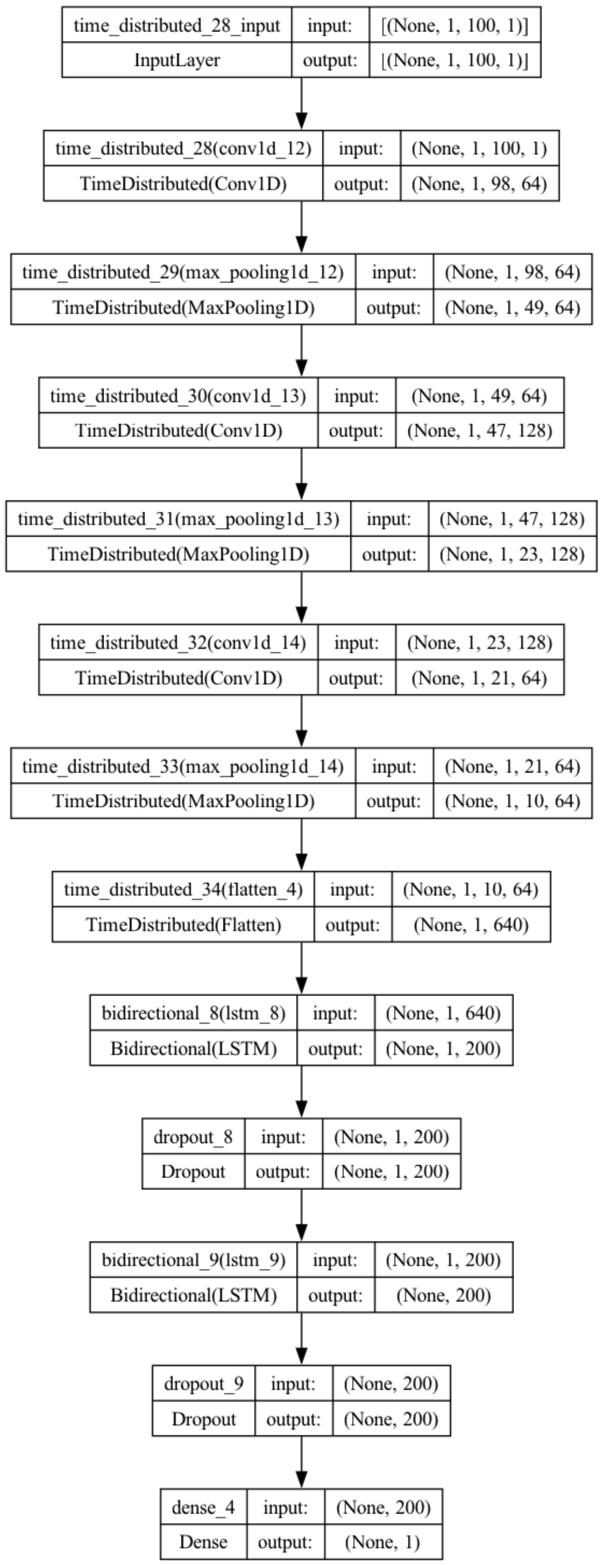

time_distributed_28_input | input: | [(None, 1, 100, 1)]
InputLayer | output: | [(None, 1, 100, 1)]
time_distributed_28(conv1d_12) | input: | (None, 1, 100, 1)
TimeDistributed(Conv1D) | output: | (None, 1, 98, 64)
time_distributed_29(max_pooling1d_12) | input: | (None, 1, 98, 64)
TimeDistributed(MaxPooling1D) | output: | (None, 1, 49, 64)
time_distributed_30(conv1d_13) | input: | (None, 1, 49, 64)
TimeDistributed(Conv1D) | output: | (None, 1, 47, 128)
time_distributed_31(max_pooling1d_13) | input: | (None, 1, 47, 128)
TimeDistributed(MaxPooling1D) | output: | (None, 1, 23, 128)
time_distributed_32(conv1d_14) | input: | (None, 1, 23, 128)
TimeDistributed(Conv1D) | output: | (None, 1, 21, 64)
time_distributed_33(max_pooling1d_14) | input: | (None, 1, 21, 64)
TimeDistributed(MaxPooling1D) | output: | (None, 1, 10, 64)
time_distributed_34(flatten_4) | input: | (None, 1, 10, 64)
TimeDistributed(Flatten) | output: | (None, 1, 640)
bidirectional_8(lstm_8) | input: | (None, 1, 640)
Bidirectional(LSTM) | output: | (None, 1, 200)
dropout_8 | input: | (None, 1, 200)
Dropout | output: | (None, 1, 200)
bidirectional_9(lstm_9) | input: | (None, 1, 200)
Bidirectional(LSTM) | output: | (None, 200)
dropout_9 | input: | (None, 200)
Dropout | output: | (None, 200)
dense_4 | input: | (None, 200)
Dense | output: | (None, 1)

Figure 13. The architecture of the CNN-LSTM model

As Figure 11 shows, the architecture of the CNN-LSTM model consists of the CNN layers as the very first layers. For the CNN part of the model, instead of using an ascending method for the layer sizes, I used a custom method. And, I created three layers of neurons with sizes 64, 128, and 64 with the kernel size of 3 with MaxPooling layers positioned between them. To eventually convert the tensors back to a 1-D array, I added a Flatten layer after the CNN section. Since this is a TSF problem, all CNN layers are added with the TimeDistributed function to train each temporal slice of input. The data is then transmitted to the LSTM layers after processing.

For the LSTM part, I created two Bi-LSTM layers to recognise the features and train the features both forward and backwards. The size of a neuron in each layer is 100. Furthermore, dropout layers are added between some features for stability, with a value of 0.5. Lastly, I added a dense layer with the linear activation function. The "adam" optimiser was implemented at the final layer due to the characteristics of regression analysis. Also, Mean Squared Error was adopted as the loss function. As the evaluation metrics, not only MSE, Mean Absolute Error was also additionally adopted. Figures 11, 12, and 13 display the model's architecture. After training the constructed CNN-LSTM model, I plotted the model's loss values for both training and validation to evaluate. The trend of the visualised plot shown in Figures 14 and 15 describes, what seems generally acceptable volatility. The testing dataset was then successfully converted back to arrays using the reshape() function, and the model was successfully trained to predict the dataset. The graph is plotted, and the results are satisfactory. The general construction of the CNN-LSTM model was inspired by the paper: *Predicting Stock Market Time-Series Data using CNN-LSTM Neural Network Model* [25].

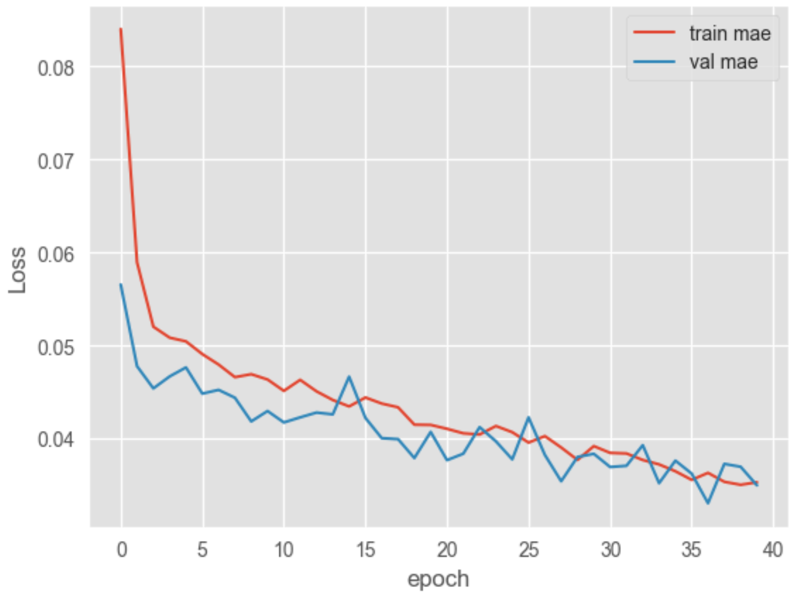


Figure 14. The MAE graph during training

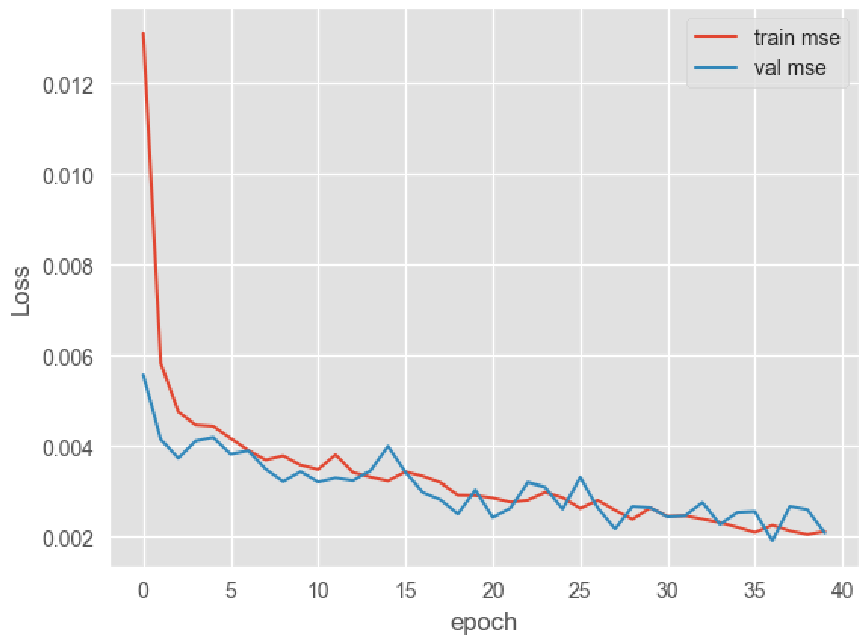
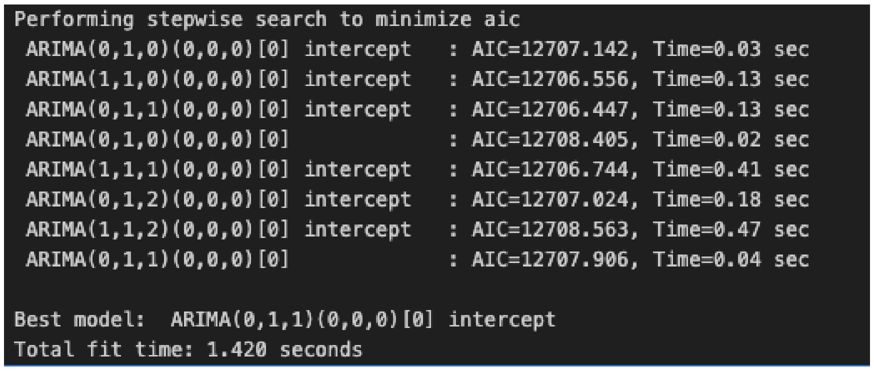


Figure 15. The MSE graph during training

5. Model Construction & Training for ARIMA model

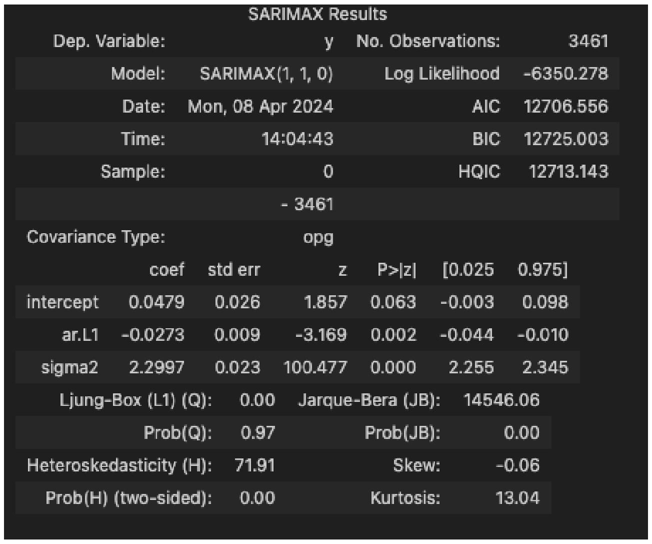
By implementing the pmdarima library to use ndiffs and auto_arima functions, it was possible to identify the ARIMA model's AR order p and MA order q. Figures 16 and 17 show the result of the suggested d, p, and q values of the ARIMA model and its summary with the pmdarima library. The auto_arima function was implemented for the model construction, since depending on the company's dataset, it was required to amend the values of d, p, or q. In ARIMA models, characterised by the parameters AR(p), differences(d), and MA(q), it is common to use values where $p + q < 2$ and $p * q = 0$. It is because either p or q tends to be the "0" value. This is typical because the data tends to exhibit a strong tendency towards either an AR or MA component, so these parameters are chosen accordingly. Therefore, except for setting the parameter of the start and max values of the AR(p) and MA(q), the rest of the model fitting was autonomously conducted based on the input dataset. Consequently, as I did during the construction and training of the CNN-LSTM model, certain visualisations were adopted to assess the ARIMA model and its inserted data (Figure 18). Also, Mean Squared Error and Mean Absolute Error values were adopted to evaluate after the training part of the ARIMA model.



```
Performing stepwise search to minimize aic
 ARIMA(0,1,0)(0,0,0)[0] intercept   : AIC=12707.142, Time=0.03 sec
 ARIMA(1,1,0)(0,0,0)[0] intercept   : AIC=12706.556, Time=0.13 sec
 ARIMA(0,1,1)(0,0,0)[0] intercept   : AIC=12706.447, Time=0.13 sec
 ARIMA(0,1,0)(0,0,0)[0]             : AIC=12708.405, Time=0.02 sec
 ARIMA(1,1,1)(0,0,0)[0] intercept   : AIC=12706.744, Time=0.41 sec
 ARIMA(0,1,2)(0,0,0)[0] intercept   : AIC=12707.024, Time=0.18 sec
 ARIMA(1,1,2)(0,0,0)[0] intercept   : AIC=12708.563, Time=0.47 sec
 ARIMA(0,1,1)(0,0,0)[0]             : AIC=12707.906, Time=0.04 sec

Best model:  ARIMA(0,1,1)(0,0,0)[0] intercept
Total fit time: 1.420 seconds
```

Figure 16. The ARIMA model for the stock data of Apple



SARIMAX Results

Dep. Variable:	y	No. Observations:	3461
Model:	SARIMAX(1, 1, 0)	Log Likelihood	-6350.278
Date:	Mon, 08 Apr 2024	AIC	12706.556
Time:	14:04:43	BIC	12725.003
Sample:	0	HQIC	12713.143
	- 3461		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
intercept	0.0479	0.026	1.857	0.063	-0.003	0.098
ar.L1	-0.0273	0.009	-3.169	0.002	-0.044	-0.010
sigma2	2.2997	0.023	100.477	0.000	2.255	2.345

Ljung-Box (L1) (Q):	0.00	Jarque-Bera (JB):	14546.06
Prob(Q):	0.97	Prob(JB):	0.00
Heteroskedasticity (H):	71.91	Skew:	-0.06
Prob(H) (two-sided):	0.00	Kurtosis:	13.04

Figure 17. The summary of the ARIMA model for the stock data

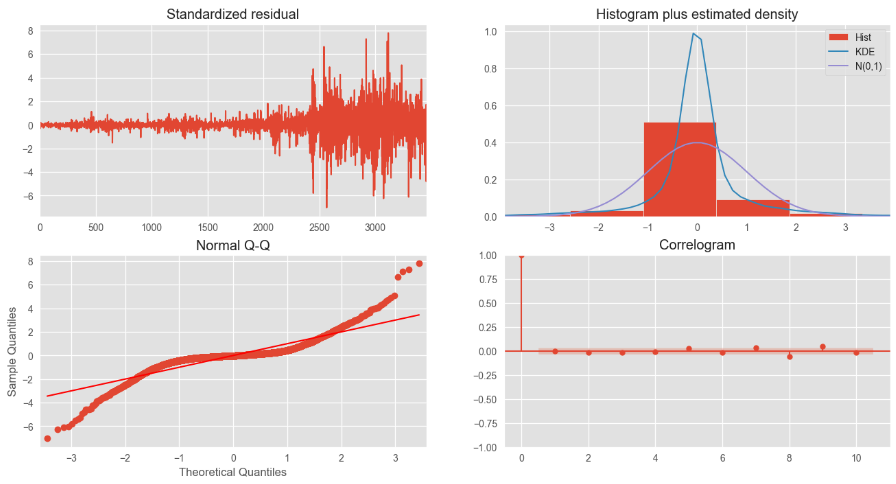


Figure 18. The Diagnostic Plots for the ARIMA and its input data

Evaluation

For the models' evaluation and each model's return value comparison, I utilised Apple's stock data to test, train, and compare. The CNN-LSTM model was trained using various technical indicators in addition to a company's price data, whereas the ARIMA model was assessed using only closing price data. Each model's MSE and MAE values are shown as below:

	ARIMA	CNN-LSTM
MSE	2.3838	0.0021
MAE	5.4084	0.0350

Table 2. Comparison of the MSE and MAE between ARIMA & CNN-LSTM

MSE (Mean Squared Error) and MAE (Mean Absolute Error) are the commonly used metrics in regression analysis problems, where lower values indicate higher model accuracy. However, as Table 2 shows, the ARIMA model's MSE and MAE values are significantly higher compared to those of the CNN-LSTM model.

To determine if the features implemented and calculated from basic price data and included in the CNN-LSTM model's dataset were effectively applied, it was required to check if there was overfitting and whether the model had accurately captured patterns of the stock data. Besides the MSE and MAE evaluation part, analysing Variance, R2 Score, and Max Error values were considered helpful for the evaluation of the CNN-LSTM model's ability and performance. The resulting values are in Table 3.

CNN-LSTM	
Variance	0.942633
R2 Score	0.942633
Max Error	0.275532

Table 3. CNN-LSTM model's Variance, R2 Score, and Max Error values

	ARIMA	CNN-LSTM
Predicted Cumulative Return	1.28738	1.11108
Actual Cumulative Return	1.27786	0.94686

Sharpe Ratio	0.48638	1.68408

Table 4. Comparison of the cumulative returns and Sharpe ratio between ARIMA & CNN-LSTM

Thus, I compared the cumulative returns and Sharpe ratio. According to Table 4, the ARIMA model showed a cumulative return of approximately 129%, but the actual return value without applying the ARIMA strategy was 128%. It shows almost no difference. The Sharpe ratio, an index used in economics to judge investment performance, is considered to appropriate stock to invest in if it is above 0 and attractive if the index is above 1 [26]. The Sharpe ratio based on the ARIMA model is 0.49.

In contrast, the CNN-LSTM model resulted in a cumulative return of 111%, while the strategy of not implementing the model would show a loss of about 5%. This suggests that an actual profit of 6% if one invested in as indicated by the CNN-LSTM model. Moreover, the Sharpe ratio of the CNN-LSTM strategy shows a significant difference of more than 1 compared to the ARIMA model strategy. It can be concluded that CNN-LSTM is indicating a much more accurate judgement.

Not only based on the tables, Figures 19 and 20 which visualise the cumulative returns of both ARIMA and CNN-LSTM models' cumulative returns values show evident insights that the ARIMA model can be fundamentally weak compared to the CNN-LSTM model since it shows little to no difference from the actual values. This result could indicate that the model may not be considered highly accurate, potentially assuming that the price on the current day is equivalent to the previous day's price data.

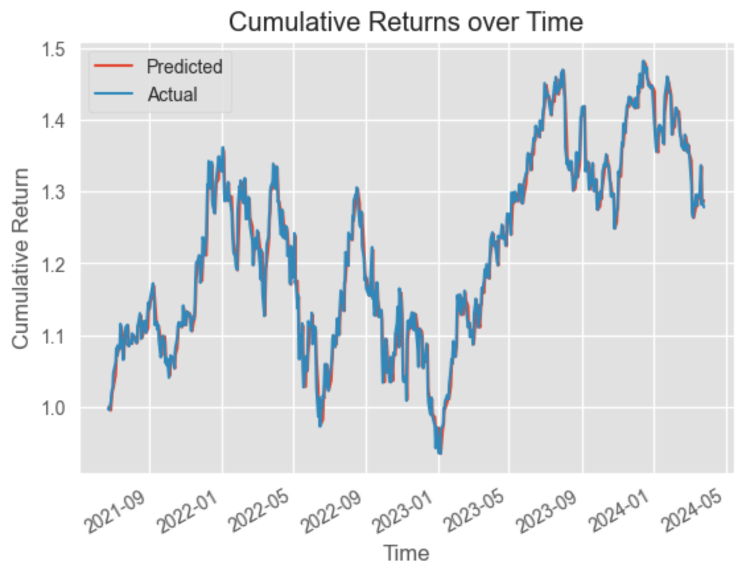


Figure 19. ARIMA model's cumulative returns during the testing period

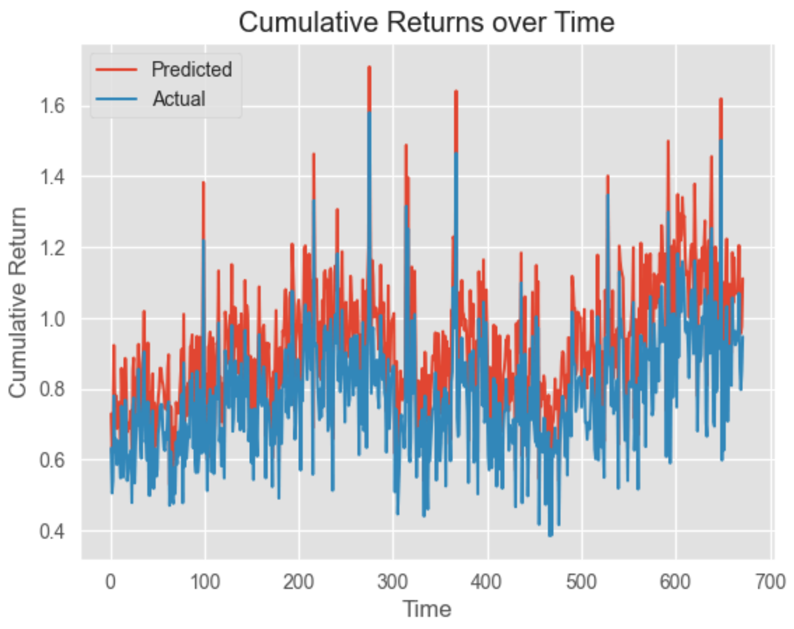


Figure 20. CNN-LSTM model’s cumulative returns during the testing period

System Designs

The application was developed for visualisation purposes based on the resulting academic insights. It consists of three separate parts: a MySQL database server, a frontend based on Streamlit, which enables the visualisation of data analysed in the field of data science, and a backend based on Python's Flask API. The frontend, backend, and database servers are all running concurrently in three different containers on the same environment through Docker Compose. In this project, CNN-LSTM and ARIMA models are used to compare analysis of Time Series Data. The application part of the analysis is highly complicated and well-developed in terms of software engineering, despite having a minimal structure. The developed application's overall architecture is shown in Figure 21. The application, named Won-trading-room, aims to provide insights into analysing stock data. The Won-trading-room application was set up on an AWS EC2 instance after development. The programme may be accessible via the 18.139.100.114 public domain that AWS provides, and because it is running on an Ubuntu operating system, it can use the UNIX-based job scheduler crontab to call scheduler_requests.py on the front end. Docker Compose makes sure that every container is connected, which makes it possible to implement JSON requests and responses. The backend functions as an API server, giving users opportunities to enjoy the effectiveness of RESTful APIs.

There are five pages in the Won-trading room, starting with the main home page. As seen in Figures 22, 23, 24, and 25, the application accessed through the deployed domain is separated into a sidebar and main components. To make the user experiences and interfaces appropriate and effective features such as the colour of the homepage and resizing of the line chart are also included. Also, the general design focuses on an intuitive layout as a method of effectively delivering information to users.

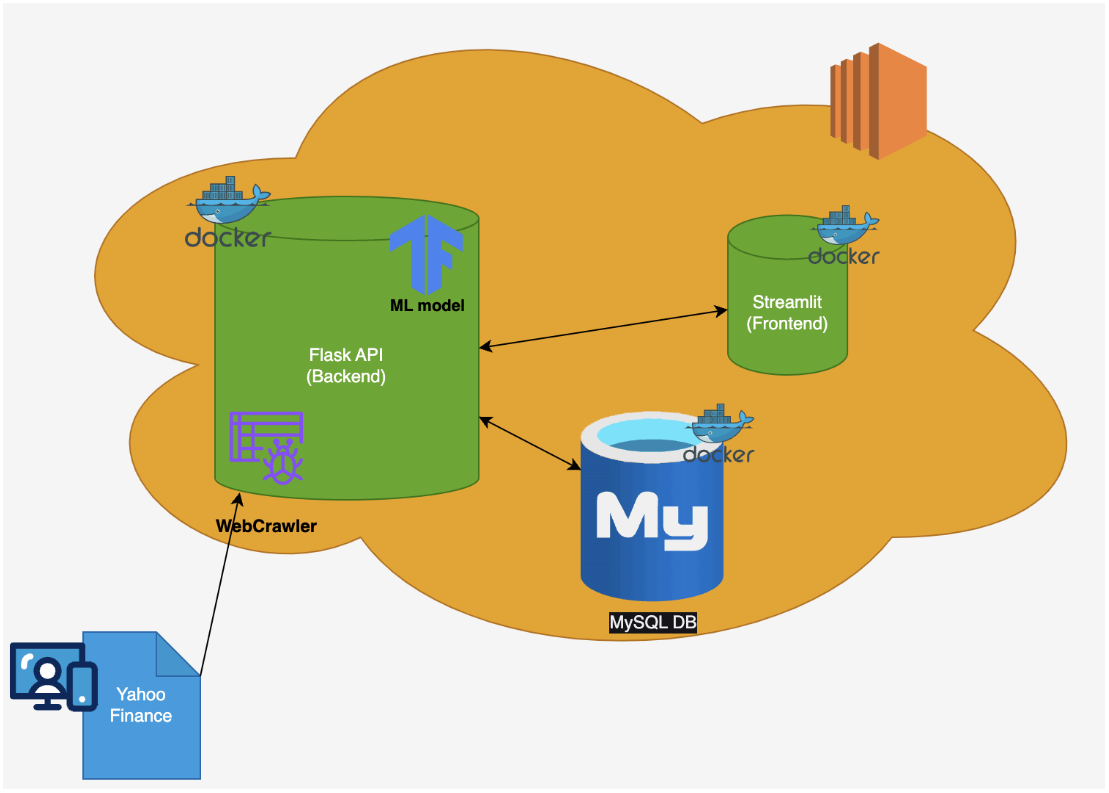
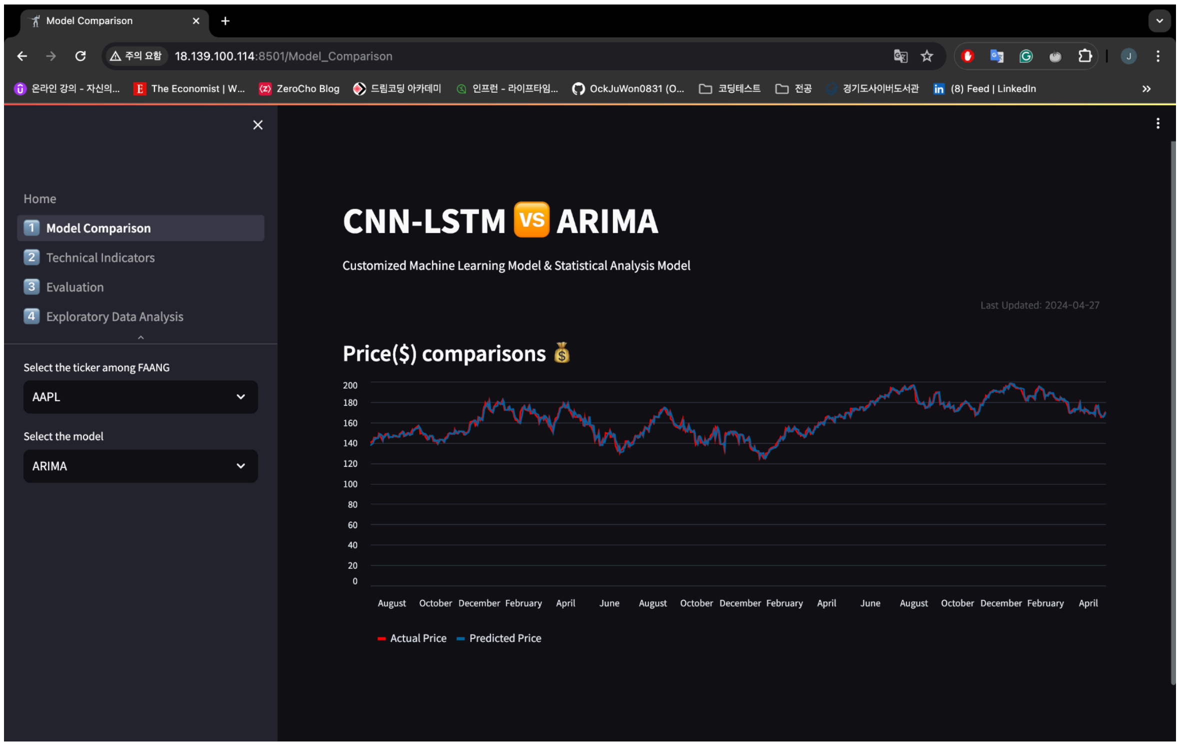


Figure 21. The architecture of Won-trading-room



Select the ticker among FAANG: AAPL

Select the model: ARIMA

CNN-LSTM VS ARIMA

Customized Machine Learning Model & Statistical Analysis Model

Last Updated: 2024-04-27

Price($) comparisons

Figure 22. Model comparison page of the application

The two models' prediction results can be reviewed on the model comparison page. Each ARIMA and CNN-LSTM model's logic has been executed on the backend server, and 20% of the entire crawled data's actual price values and predicted price values are visualised for comparison. Red and blue are the colours used for line comparison in all of the application's line charts as they were found to have the highest contrast. One of the five FAANG companies can be selected, alongside the analytical model for that company, using the sidebar. A full perspective of the information is provided by the main element. Figures 22 and 23 show the model comparison page.

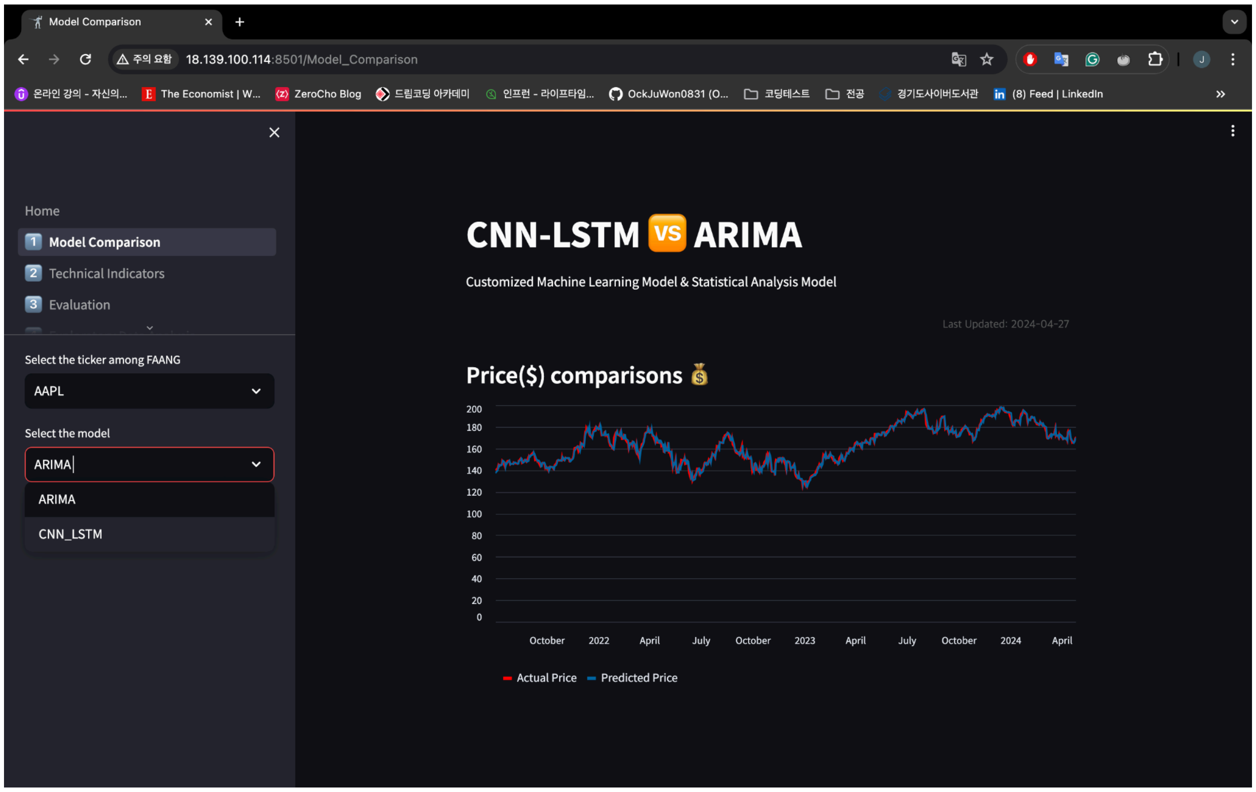
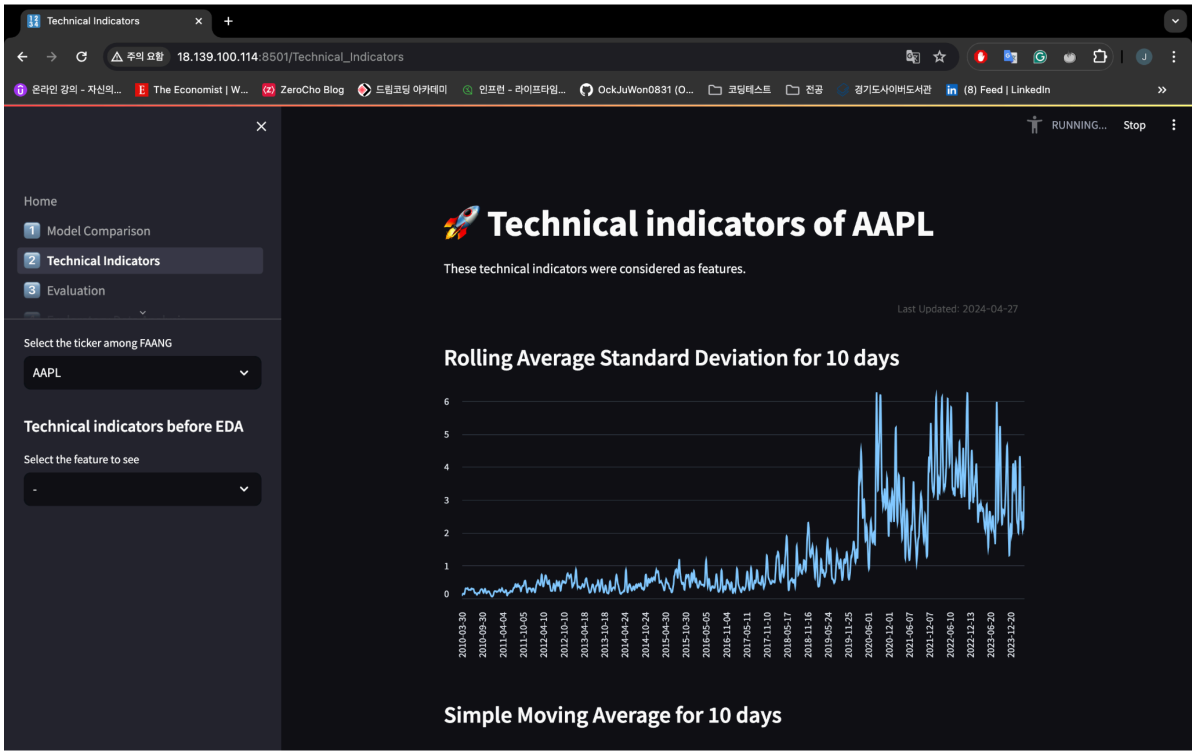


Figure 23. Model comparison page of the application 2.

All the features that were taken into consideration during the training of the CNN-LSTM model for this project are presented on the Technical Indicators page. This was chosen because, despite their lack of relevance to Close data, they are thought to offer distinct insights for stock prediction. Users can load and check the data they are interested in independently using the sidebar. The Close data, a bar chart showing the distribution of daily returns, and a line chart displaying daily returns altogether are all displayed to the users if they choose the Close data from the select bar. The page with the technical indicators is shown in Figure 24.



Select the ticker among FAANG: AAPL

Technical indicators before EDA

Select the feature to see: -

🚀 Technical indicators of AAPL

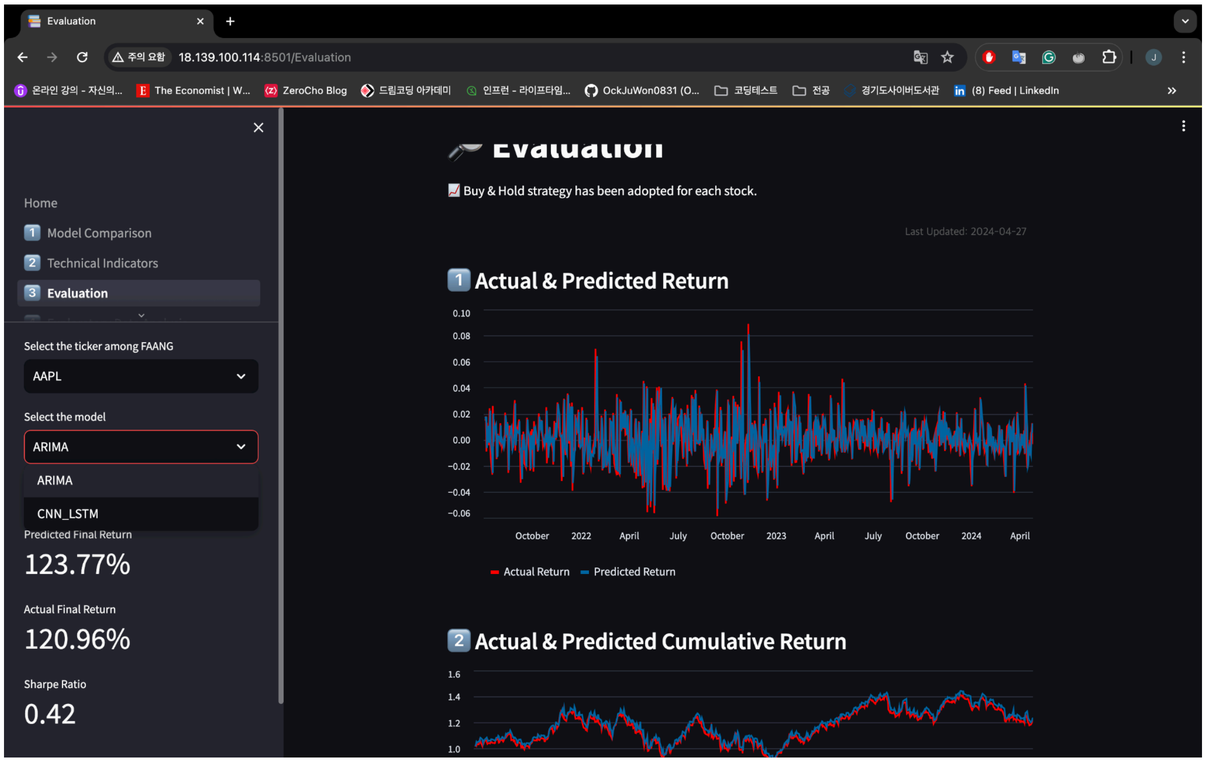
These technical indicators were considered as features.

Last Updated: 2024-04-27

Rolling Average Standard Deviation for 10 days

Simple Moving Average for 10 days

Figure 24. Technical indicators page of the application



Select the ticker among FAANG: AAPL

Select the model: ARIMA

Predicted Final Return: 123.77%

Actual Final Return: 120.96%

Sharpe Ratio: 0.42

Evaluation

📈 Buy & Hold strategy has been adopted for each stock.

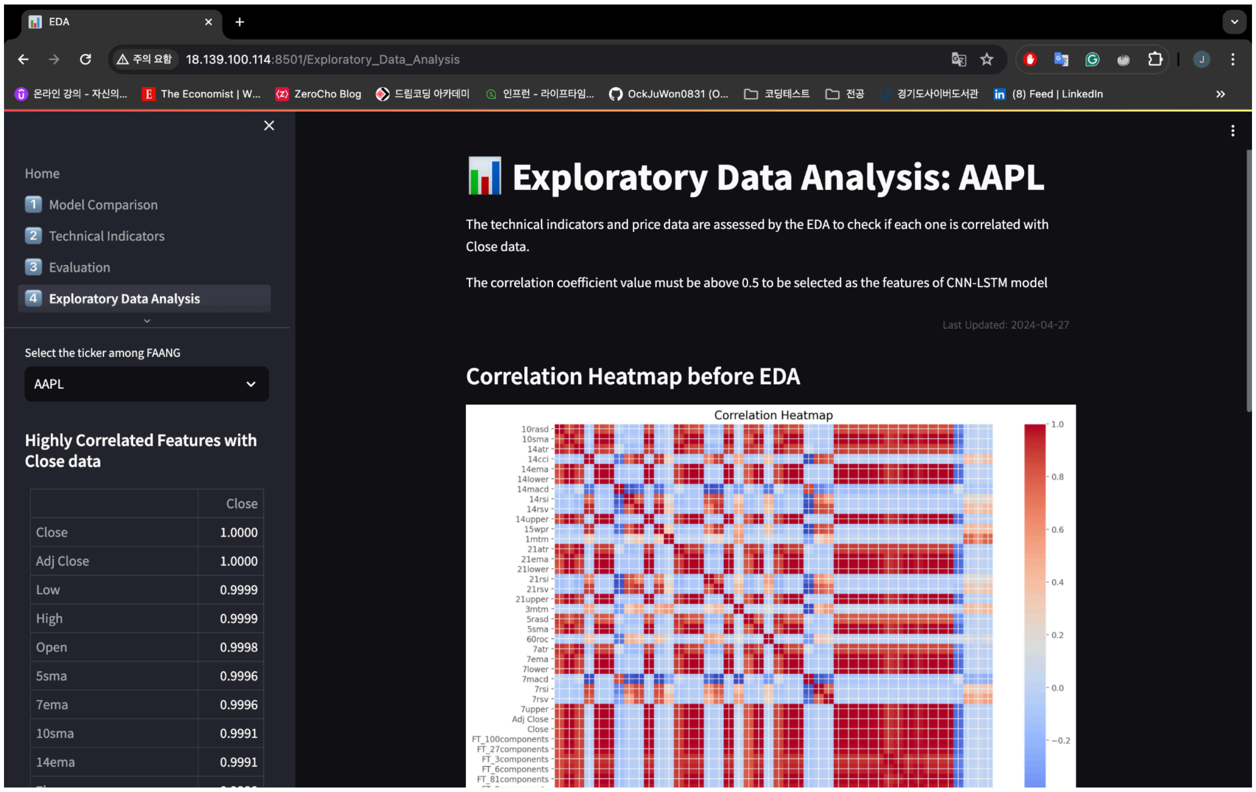
Last Updated: 2024-04-27

1 Actual & Predicted Return

2 Actual & Predicted Cumulative Return

Figure 25. Evaluation page of the application

The return values and cumulative return values from the generated CNN-LSTM and ARIMA models are shown in line charts on the Evaluation page. Users can view the calculated returns through the sidebar. These numbers assist users in determining which model to employ when buying and selling stocks in the future by making the Sharpe Ratio calculation easier. In Figure 25, the evaluation page is displayed.



Exploratory Data Analysis: AAPL

The technical indicators and price data are assessed by the EDA to check if each one is correlated with Close data.

The correlation coefficient value must be above 0.5 to be selected as the features of CNN-LSTM model

Correlation Heatmap before EDA

Highly Correlated Features with Close data

	Close
Close	1.0000
Adj Close	1.0000
Low	0.9999
High	0.9999
Open	0.9998
5sma	0.9996
7ema	0.9996
10sma	0.9991
14ema	0.9991

Figure 26. Exploratory Data Analysis page of the application

The correlation coefficients between the data on the Technical Indicators page and related data are graphically shown in a heatmap on the EDA page. After analysing the correlations with the close data, a heatmap of the dropped values is also included. A table in the sidebar lets users know which values fall on a scale from 0 to 1. Figure 26 shows the page with the data before and after going through the EDA process.

Implementation

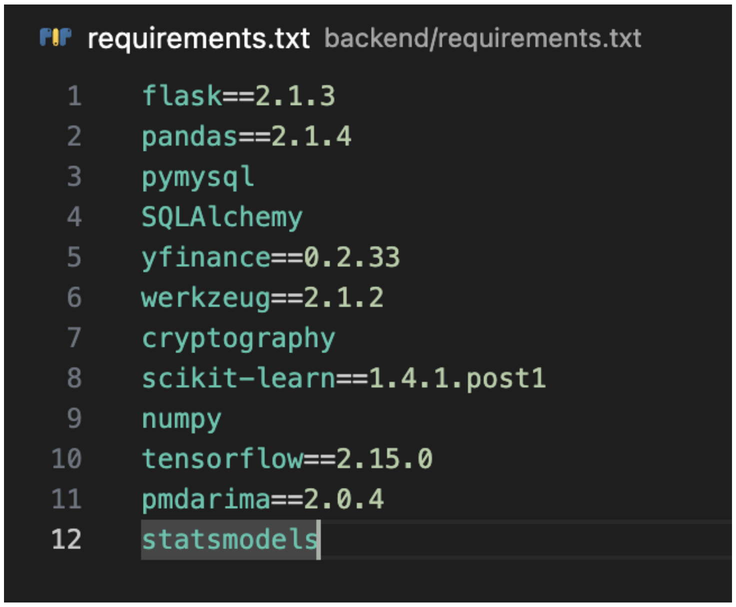
In this part, the technologies and libraries used in developing the Won-trading-room application are explained and introduced. As mentioned in the system designs part earlier, the application consists of three layers as a web application. Additionally, since the application was deployed on an EC2 instance through AWS services, the deployment process and related technologies are subsequently explained.

- Backend

Python's Flask API web application library was used to create the backend part of the application. The decision to utilise Flask over alternative frameworks, such as Python's Django or Java's Spring Boot, was made by several reasons. 1) A Python-based framework from the same ecosystem was thought to be appropriate for an application intended for visualisation; 2) Since the machine learning models were written and analysed using TensorFlow, the backend server constructed in Java Spring Boot was considered too heavy and problematic during the deployment process. 3) Flask API was eventually chosen because it makes building Backend API servers considerably faster and simpler than with Django. The backend server was given port number 5050, and it is set up to send JSON-formatted responses to GET and POST requests from the frontend. The following are the APIs implemented at the backend:

1. @app.route("/api/predict/arima-predict", methods=["POST"])
2. @app.route("/api/predict/cnn-lstm-predict", methods=["POST"])
3. @app.route("/api/data/insert-processed-data", methods=["POST"])
4. @app.route("/api/data/include-technical-indicators", methods=["POST"])
5. @app.route("/", methods=["GET"])

The backend includes logics for user registration and login as following APIs shows: @app.route("/api/register", methods=["POST"]) and @app.route("/api/login", methods=["POST"]). However these were decided not to use due to the project's scope and the limitations of Streamlit library used in the frontend. Also, only GET and POST HTTP methods were used. As the frontend server sends JSON requests, aligning implementations with the POST method was deemed essential to stick to the RESTful API. APIs numbered 1 to 4 are POST APIs. The requests take a format like {"ticker":"AAPL"} to fetch data for one of the five cooperates from the database and send it to the frontend or to execute Python-written analysis models. The fourth POST API computes and JSONifies data for a particular ticker using technical_generator.py and data_preprocessing.py from the tool directory. As the fourth POST API, the third one calculates before inserting it into the MySQL database. Certain logics are used by the fifth GET API to crawl data from the Yahoo Finance API. Python's exception-handling methods control critical logic as the server runs inside the Docker environment, allowing it to handle errors or unexpected situations. Figure 27 shows the libraries used during the backend development.


requirements.txt backend/requirements.txt

```
flask==2.1.3
pandas==2.1.4
pymysql
SQLAlchemy
yfinance==0.2.33
werkzeug==2.1.2
cryptography
scikit-learn==1.4.1.post1
numpy
tensorflow==2.15.0
pmdarima==2.0.4
statsmodels
```

Figure 27. Libraries used for the backend development

- Frontend

It is effective in comparing the CNN-LSTM model-based results with ARIMA model-based results using Streamlit as a visualisation tool. Streamlit was chosen for some reasons. 1) Its quick setup and ease of understanding, and 2) Its variety of utilities that improve user interface/user experience. When a user interacts with the frontend, the sidebar and selection bar communicate with a backend API server to retrieve data in JSON format. It was determined that employing sessions or cookies to enhance performance was not necessary due to the characteristics of the data amount.

On the front end, it was considered inefficient from a server resource management and user experience perspective to continuously retrieve the results of stock analysis models by directly connecting to the database. Therefore, a scheduler using crontab is configured to run at 4 am in Kuala Lumpur time to conduct the crawling and stock performance analysis models. The frontend server subsequently stores the results in the local data directory.

Scheduler_requests.py is the script that runs every 4 am. The multi-threading method was used to increase productivity. Even though asynchronous I/O management and multiprocessing method could be used, the deployment was done on an EC2 free tier instance environment, which only provides a single CPU with constrained resources. Multithreading was therefore the optimal option for quick and effective I/O operations. As shown in Figures 28 and 29, using multithreading to the backend data pre-processing requests significantly decreased the processing time by approximately 30 seconds (before 598.22s, after 568.02s). The libraries used for the frontend part are shown in Figure 30.

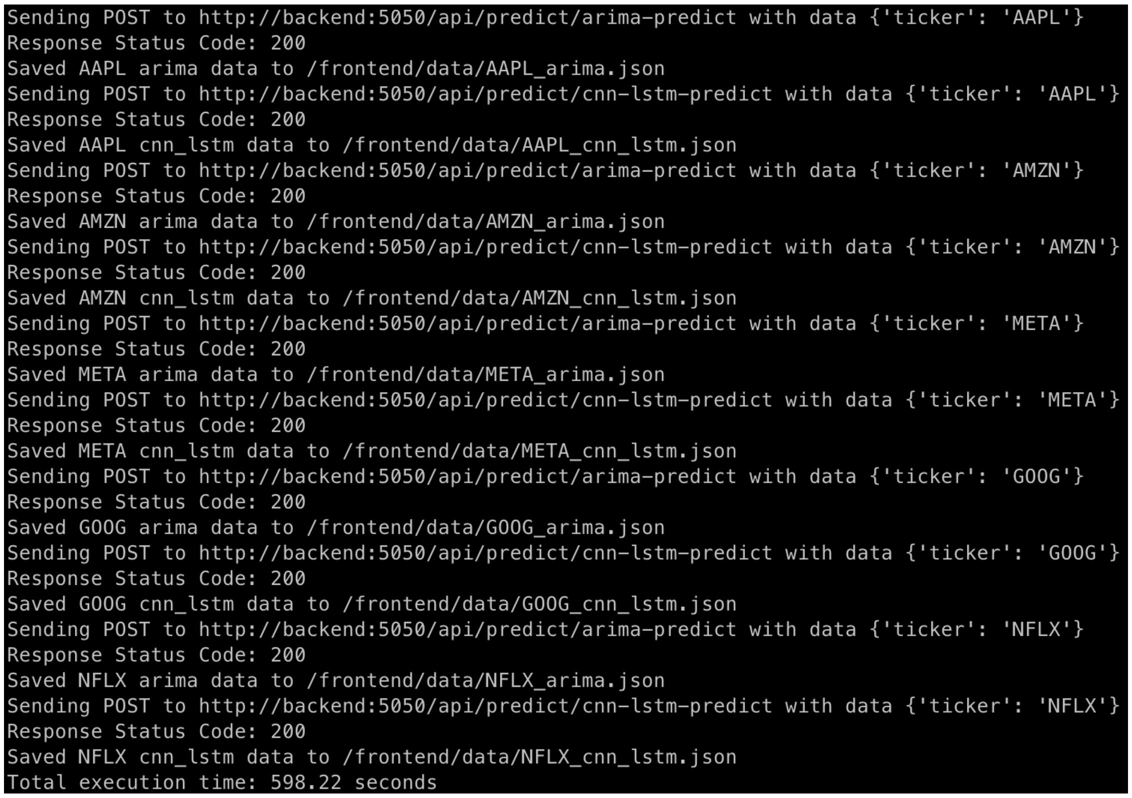

```
Sending POST to http://backend:5050/api/predict/arima-predict with data {'ticker': 'AAPL'}
Response Status Code: 200
Saved AAPL arima data to /frontend/data/AAPL_arima.json
Sending POST to http://backend:5050/api/predict/cnn-lstm-predict with data {'ticker': 'AAPL'}
Response Status Code: 200
Saved AAPL cnn_lstm data to /frontend/data/AAPL_cnn_lstm.json
Sending POST to http://backend:5050/api/predict/arima-predict with data {'ticker': 'AMZN'}
Response Status Code: 200
Saved AMZN arima data to /frontend/data/AMZN_arima.json
Sending POST to http://backend:5050/api/predict/cnn-lstm-predict with data {'ticker': 'AMZN'}
Response Status Code: 200
Saved AMZN cnn_lstm data to /frontend/data/AMZN_cnn_lstm.json
Sending POST to http://backend:5050/api/predict/arima-predict with data {'ticker': 'META'}
Response Status Code: 200
Saved META arima data to /frontend/data/META_arima.json
Sending POST to http://backend:5050/api/predict/cnn-lstm-predict with data {'ticker': 'META'}
Response Status Code: 200
Saved META cnn_lstm data to /frontend/data/META_cnn_lstm.json
Sending POST to http://backend:5050/api/predict/arima-predict with data {'ticker': 'GOOG'}
Response Status Code: 200
Saved GOOG arima data to /frontend/data/GOOG_arima.json
Sending POST to http://backend:5050/api/predict/cnn-lstm-predict with data {'ticker': 'GOOG'}
Response Status Code: 200
Saved GOOG cnn_lstm data to /frontend/data/GOOG_cnn_lstm.json
Sending POST to http://backend:5050/api/predict/arima-predict with data {'ticker': 'NFLX'}
Response Status Code: 200
Saved NFLX arima data to /frontend/data/NFLX_arima.json
Sending POST to http://backend:5050/api/predict/cnn-lstm-predict with data {'ticker': 'NFLX'}
Response Status Code: 200
Saved NFLX cnn_lstm data to /frontend/data/NFLX_cnn_lstm.json
Total execution time: 598.22 seconds
```

Figure 28. Before applying the multi-threading method

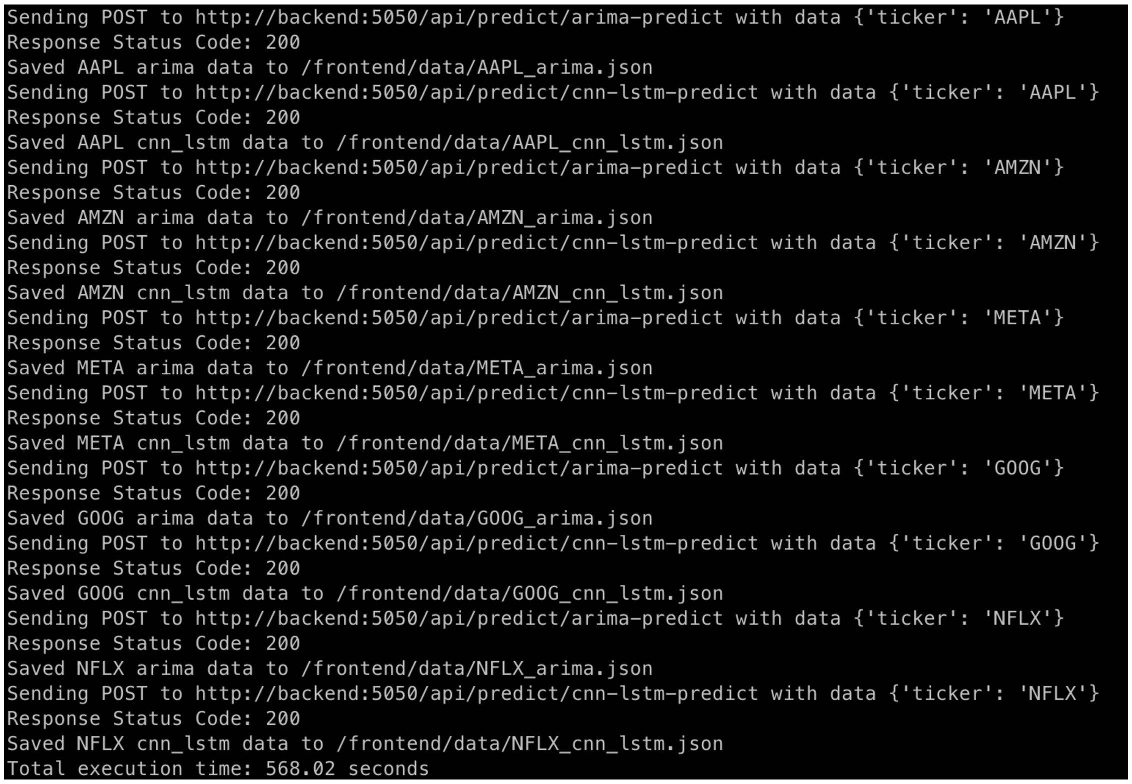
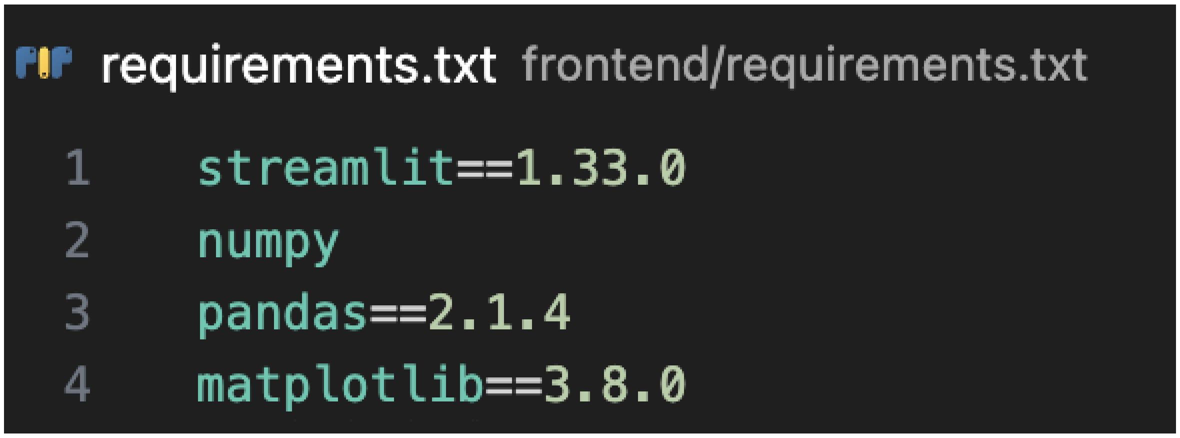

```
Sending POST to http://backend:5050/api/predict/arima-predict with data {'ticker': 'AAPL'}
Response Status Code: 200
Saved AAPL arima data to /frontend/data/AAPL_arima.json
Sending POST to http://backend:5050/api/predict/cnn-lstm-predict with data {'ticker': 'AAPL'}
Response Status Code: 200
Saved AAPL cnn_lstm data to /frontend/data/AAPL_cnn_lstm.json
Sending POST to http://backend:5050/api/predict/arima-predict with data {'ticker': 'AMZN'}
Response Status Code: 200
Saved AMZN arima data to /frontend/data/AMZN_arima.json
Sending POST to http://backend:5050/api/predict/cnn-lstm-predict with data {'ticker': 'AMZN'}
Response Status Code: 200
Saved AMZN cnn_lstm data to /frontend/data/AMZN_cnn_lstm.json
Sending POST to http://backend:5050/api/predict/arima-predict with data {'ticker': 'META'}
Response Status Code: 200
Saved META arima data to /frontend/data/META_arima.json
Sending POST to http://backend:5050/api/predict/cnn-lstm-predict with data {'ticker': 'META'}
Response Status Code: 200
Saved META cnn_lstm data to /frontend/data/META_cnn_lstm.json
Sending POST to http://backend:5050/api/predict/arima-predict with data {'ticker': 'GOOG'}
Response Status Code: 200
Saved GOOG arima data to /frontend/data/GOOG_arima.json
Sending POST to http://backend:5050/api/predict/cnn-lstm-predict with data {'ticker': 'GOOG'}
Response Status Code: 200
Saved GOOG cnn_lstm data to /frontend/data/GOOG_cnn_lstm.json
Sending POST to http://backend:5050/api/predict/arima-predict with data {'ticker': 'NFLX'}
Response Status Code: 200
Saved NFLX arima data to /frontend/data/NFLX_arima.json
Sending POST to http://backend:5050/api/predict/cnn-lstm-predict with data {'ticker': 'NFLX'}
Response Status Code: 200
Saved NFLX cnn_lstm data to /frontend/data/NFLX_cnn_lstm.json
Total execution time: 568.02 seconds
```

Figure 29. After applying the multi-threading method



requirements.txt frontend/requirements.txt

```
streamlit==1.33.0
numpy
pandas==2.1.4
matplotlib==3.8.0
```

Figure 30. Libraries used for the frontend development

- **Database**

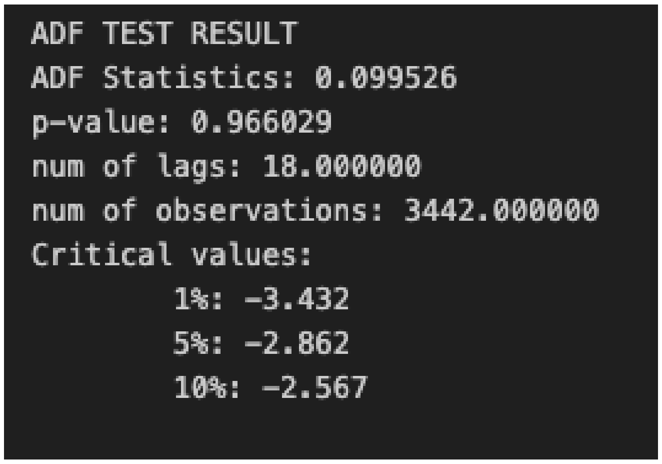
Despite taking into consideration alternatives like Redis, PostgreSQL, or NoSQL databases like MongoDB, MySQL was selected as the database due to its ease of use when managing structured data. The environment variables necessary for setting up a connection with the database server are saved in the .env file for security purposes. To improve security, it was decided to create and configure this .env file on the EC2 server during deployment. In addition, the required databases and tables are created during the server launching through a script called db_init.sql. The approach to development for the data handling logic focused on optimising database transactions and updating data rather than adding new data. PyMySQL and SQLAlchemy were two of the libraries utilised for database-related procedures.

- **Deployment**

During the development process, the initial plan was to deploy the application on a free-tier EC2 instance from AWS services. But when the application was run locally, it consumed around 30GB of storage and about 7GB of RAM, which meant that when the server was deployed on the free tier, it would crash due to a lack of resources. As a result, the application was set up on an Ubuntu-based t3.large instance on AWS, which offers two virtual CPUs and 8GB of RAM to guarantee consistent performance. It was confirmed that additional devices could successfully access the deployed instance's public IP. Although the process of fetching data from Yahoo Finance and calculating stock returns using models took longer compared to the local environment, no additional modifications were made. This decision was based on two considerations: 1) the calculation and crawling logic were scheduled to run at 4 am considered as the least active time in a day, and 2) the server was capable of handling the load without further adjustments.

Discussions & Future Directions

There are two critical characteristics to consider before analysing stock prices. First of all, unlike other time series data, stock prices indicate a high degree of randomness and are possible to be considered unpredictable random walks [27]. Secondly, it is assumed that the next day's data of the stock price reflects the previous day's data, which are: "Open", "High", "Low", and "Close". It is concluded that future information is already reflected in stock prices. Lastly, randomness is found during the construction of the ARIMA model, where the Augmented Dickey-Fuller (ADF) test indicated that the dataset for training is non-stationary. According to the result of the ADF test below (Figure 31), the p-value of the training data is more than 0.05, which cannot reject the null hypothesis. It reflects that the original stock price data is difficult to spot the patterns as time series data. So, stock price analysis using CNN-LSTM and ARIMA models has definitive limitations. However, this project provides academic insights and opportunities for discussions into more accurately analysing stock prices as a TSF project.



```
ADF TEST RESULT
ADF Statistics: 0.099526
p-value: 0.966029
num of lags: 18.000000
num of observations: 3442.000000
Critical values:
        1%: -3.432
        5%: -2.862
        10%: -2.567
```

Figure 31: Augmented Dickey-Fuller(ADF) test result of the Apple data

First of all, the CNN-LSTM model made better precision than the ARIMA model, and it can be defined as a result of its complexities, sophistication, and the implementation of various input data as features. In addition to simply including price data and stock-related technical indicators, expanding the dataset to include additional ETFs such as the S&P 500 or other technical data could improve the results. [28]

Secondly, the data from day "n-1" and day "n" have a close relationship which makes the stock data regression analysis inaccurate. Therefore, modifying the prediction problem from a simple next-day stock price regression to a classification problem could be an alternative option. If the stock price rises from the previous day, then it can be labelled as "1". For the opposite cases, "0" or "-1" can be labelled for the classification. It could give the model to forecast the trend of price movements but the exact stock data, which is potentially impossible due to several factors. [29]

Thirdly, since CNN-LSTM achieves better results, it is possible to get further enhancements by adjusting other deep learning or machine learning models. Even though CNN was utilised in the LSTM model in this study, many more accurate predictions might result from testing with different and cutting-edged architectures, or even implementing other models like the Transformer, which is commonly employed in LLM models like GPT. [30]

Another important discovery during the CNN-LSTM model's development was the massive impact that even minor modifications in the LSTM model's parameters can have on the results. Notably, the two key variables are 1) the window's size, and 2) the amount of training data that it holds [31]. Given that LSTM models have limitations when it comes to interpreting long-term data, experimenting with parameter adjustments can lead to much better outcomes by minimising these disadvantages and maximising the advantages of Time Series Forecasting.

Another strategy is to improve the construction of the dataset for the training models by making it more multi-dimensional [32], which will improve the accuracy of the machine learning model training, as opposed to just adding technical indications as features. Furthermore, using several scaling strategies in the data processing process instead of merely Min-Max scaling can offer a comparative study. To construct machine learning models for time series forecasting, the data preprocessing step must become more comprehensive and reliable.

It is necessary to have a deeper understanding of financial engineering to apply statistical insights. The performance to predict the trend of the stock of the model can be greatly increased by utilising several statistical equations that are frequently used in financial engineering during the data preparation procedure and by utilising the results that have been computed through various techniques [33] [34].

Conclusion

Throughout the project, I compared the customised CNN-LSTM and ARIMA models to determine which model is more suitable and accurate for predicting the "Close" data of stocks. The CNN-LSTM model demonstrated superior numerical performance in analysing stock prices compared to the ARIMA model. Despite this, the inherent unpredictability and complexity of stock price data cast significant challenges and limitations to any modelling approach. As previously mentioned, alternate approaches like changing the problem into a classification task or improving the dataset by adding more variables and using different preprocessing methods can lead to better outcomes. Adjusting model parameters and implementing advanced machine learning architectures may potentially increase forecasting accuracy. In conclusion, even though the CNN-LSTM model shows significant results, continuous exploration and adaptation of different strategies and technologies are important in advancing the capabilities of stock price prediction as Time Series Forecasting task.

References

[1] Shi, J., Jain, M., & Narasimhan, G. (2022, April 23). Time Series Forecasting (TSF) Using Various Deep Learning Models. ArXiv.org. https://doi.org/10.48550/arXiv.2204.11115

[2] Chen, G., Chen, Y., & Fushimi, T. (n.d.). Application of Deep Learning to Algorithmic Trading. Retrieved November 19, 2023, from https://cs229.stanford.edu/proj2017/final-reports/5241098.pdf

[3] Pang, X. W., Zhou, Y., Wang, P., Lin, W., & Chang, V. (2018). An innovative neural network approach for stock market prediction. The Journal of Supercomputing, 76(3), 2098–2118. https://doi.org/10.1007/s11227-017-2228-y

[4] Liu, S., Liao, G., & Ding, Y. (2018). Stock transaction prediction modelling and analysis based on LSTM. 2018 13th IEEE Conference on Industrial Electronics and Applications (ICIEA). https://doi.org/10.1109/iciea.2018.8398183

[5] Selvin, S., Vinayakumar, R., Gopalakrishnan, E. A., Menon, V. K., & Soman, K. P. (2017). Stock price prediction using LSTM, RNN and CNN-sliding window model. 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI). https://doi.org/10.1109/icacci.2017.8126078

[6] LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., & Jackel, L. D. (1989). Backpropagation Applied to Handwritten Zip Code Recognition. Neural Computation, 1(4), 541–551. https://doi.org/10.1162/neco.1989.1.4.541

[7] Yani, M., Budhi Irawan, S., Si., M.T., & Casi Setiningsih, S. T., M.T. (2019). Application of Transfer Learning Using Convolutional Neural Network Method for Early Detection of Terry's Nail. Journal of Physics: Conference Series, 1201, 012052. https://doi.org/10.1088/1742-6596/1201/1/012052

[8] Sepp Hochreiter, & Jürgen Schmidhuber. (1995). Long short term memory. München Inst. Für Informatik.

[9] Akita, R., Yoshihara, A., Matsubara, T., & Uehara, K. (2016). Deep learning for stock prediction using numerical and textual information. IEEE Xplore. https://doi.org/10.1109/ICIS.2016.7550882

[10] Nabipour, M., Nayyeri, P., Jabani, H., Mosavi, A., Salwana, E., & S., S. (2020). Deep Learning for Stock Market Prediction. Entropy, 22(8), 840. https://doi.org/10.3390/e22080840

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

[12] Lu, W., Li, J., Li, Y., Sun, A., & Wang, J. (2020). A CNN-LSTM-Based Model to Forecast Stock Prices. Complexity, 2020, 1–10. https://doi.org/10.1155/2020/6622927

[13] Box, G. E. P., & Al, E. (2015). Time series analysis : forecasting and control. John Wiley & Sons.

[14] Elman, J. L. (1990). Finding Structure in Time. Cognitive Science, 14(2), 179–211. https://doi.org/10.1207/s15516709cog1402_1

[15] Saad, E. W., Prokhorov, D. V., & Wunsch, D. C. (1998). Comparative study of stock trend prediction using time delay, recurrent and probabilistic neural networks. IEEE Transactions on Neural Networks, 9(6), 1456–1470. https://doi.org/10.1109/72.728395

[16] Larson, M. (2012). Moving Average Convergence/Divergence (MACD). , 43-50. https://doi.org/10.1002/9781119204428.CH4.

[17] J Welles Wilder Jr. (1978). New concepts in technical trading systems. Trend Research.

[18] Bollinger, J. (2001). Bollinger on Bollinger bands. Mcgraw-Hill.

[19] Aby, C., & Fusilier, M. (1997). Applying Futures Contract Evaluations to Small Business Stock Portfolios and Retirement Plans: An Innovative Approach Using Technical Analysis. Academy of Accounting and Financial Studies Journal, 1, 17.

[20] Williams, L. R. (1979). How U Made One Million Dollars Last Year Trading Commodities.

[21] Quant, T. A. (2023, March 14). Predicting Stock Prices using ARIMA, Fourier Transforms, and Technical Indicators with Deep…. Medium. https://pub.towardsai.net/predicting-stock-prices-using-arima-fourier-transforms-and-technical-indicators-with-deep-43a164859683

[22] STÁDNÍK, B., RAUDELIŪNIENĖ, J., & DAVIDAVIČIENĖ, V. (2016). FOURIER ANALYSIS FOR STOCK PRICE FORECASTING: ASSUMPTION AND EVIDENCE. Journal of Business Economics and Management, 17(3), 365–380. https://doi.org/10.3846/16111699.2016.1184180

[23] Hsu, H.-H., & Hsieh, C.-W. (2010). Feature Selection via Correlation Coefficient Clustering. Journal of Software, 5(12). https://doi.org/10.4304/jsw.5.12.1371-1377

[24] Burdick, D. (1995). An introduction to tensor products with applications to multiway data analysis. Polymer. https://doi.org/10.1016/0169-7439(95)80060-M.

[25] A, A., R, R., S, V. R., & Bagde, A. M. (2023, May 21). Predicting Stock Market Time-Series Data using CNN-LSTM Neural Network Model. ArXiv.org. https://doi.org/10.48550/arXiv.2305.14378

[26] Agudo, L., & Marzal, J. (2004). An analysis of Spanish investment fund performance: some considerations concerning Sharpe's ratio. Omega-international Journal of Management Science, 32, 273-284. https://doi.org/10.1016/J.OMEGA.2003.11.006.

[27] Asiri, B. (2008). Testing weak-form efficiency in the Bahrain stock market. International Journal of Emerging Markets, 3(1), 38–53. https://doi.org/10.1108/17468800810849213

[28] Kim, T., & Kim, H. (2019). Forecasting stock prices with a feature fusion LSTM-CNN model using different representations of the same data. PLoS ONE, 14. https://doi.org/10.1371/journal.pone.0212320.

[29] Guo, J. (2023). Predict Stock Price Trend by Using Classification Model. Advances in Economics, Management and Political Sciences. https://doi.org/10.54254/2754-1169/45/20230260.

[30] Xu, T. (2022). Analysis on the Applicability of RNN, LSTM, and GRU Deep Learning Algorithms for Stock Price Prediction. Proceedings of the International Conference on Big Data Economy and Digital Management. https://doi.org/10.5220/0011175000003440.

[31] Taslim, D. G., & Murwantara, I. M. (2022). A Comparative Study of ARIMA and LSTM in Forecasting Time Series Data. 2022 9th International Conference on Information Technology, Computer, and Electrical Engineering (ICITACEE). https://doi.org/10.1109/icitacee55701.2022.9924148

[32] Shang, D., Shang, P., & Liu, L. (2018). Multidimensional scaling method for complex time series feature classification based on generalized complexity-invariant distance. Nonlinear Dynamics, 95(4), 2875–2892. https://doi.org/10.1007/s11071-018-4728-6

[33] Ludkovski, M. (2022). Statistical Machine Learning for Quantitative Finance. Annual Review of Statistics and Its Application, 10(1). https://doi.org/10.1146/annurev-statistics-032921-042409

[34] Bounid, S., Oughanem, M., & Salman Bourkadi. (2022). Advanced Financial Data Processing and Labeling Methods for Machine Learning. https://doi.org/10.1109/iscv54655.2022.9806060