Module Code: CS3AM

Assignment Report Title: Stock Analysis of Apple, with time series forecasting

Name: Lyndon Fernandes

Student Number: 30000205

Date (when work completed): 31/11/2024

Actual hours spent for assignment: 30

Please read the README.md file in the zip file for instructions on file locations.

Contents

[Abstract 2](#_Toc184499964)

[Exploratory data analysis 2](#_Toc184499965)

[Data pre-processing and feature selection 3](#_Toc184499966)

[Machine learning model: ARIMA 4](#_Toc184499967)

[Deep Learning model: LSTM 6](#_Toc184499968)

[Comparisons across the models built 8](#_Toc184499969)

[Conclusion, recommendations, and future work 8](#_Toc184499970)

[References 9](#_Toc184499971)

## Abstract

The stock market is a complex and dynamic system influenced by many factors from economic conditions (Chen, 2024) such as GDP growth, inflation and interest rates to investor sentiment (Bajaj Broking, 2024), which is largely emotional and can have impacts on share prices. Accurately predicting stock prices and market trends has been at the forefront of many investors, financial institutes to optimize investment strategies and manage risk which when done correctly can yield substantial rewards.

Time series analysis has been a powerful technique to uncover hidden patterns and trends in stock market data (Kaniugu, 2023). By analysing historical patterns and identifying recurring patterns such as seasonality it provides valuable insights that can inform us about short-term future price movements. Hence this approach is well suited for capturing the sequential nature of financial data an account for temporal dependencies in the stock market.

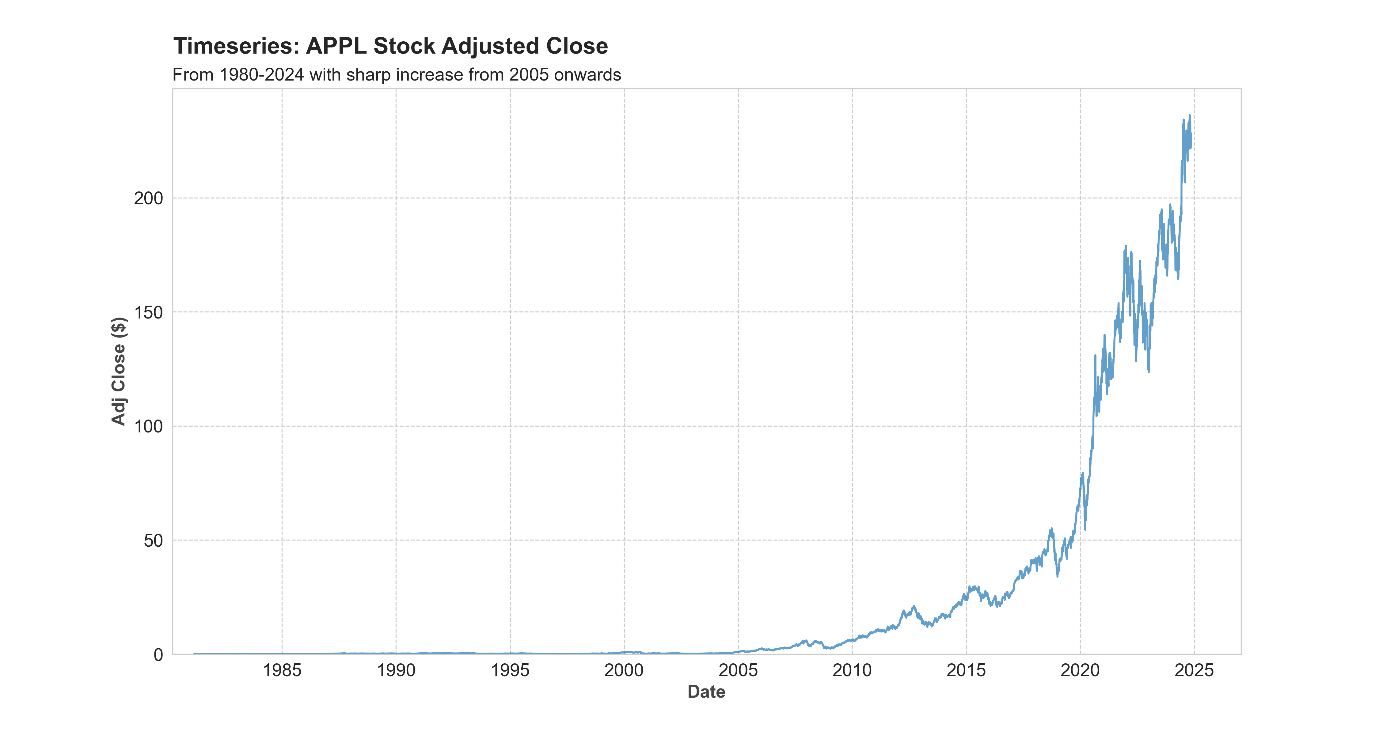
However, stock price forecasting remains a challenge due to the inherent volatility and non-linear nature of the financial markets even among immense asset managers and hedge funds such as BlackRock even with complex and power full systems such as Aladdin. Traditional statistical, machine learning algorithms, and deep learning models have been applied with varying degrees of success.

## Exploratory data analysis

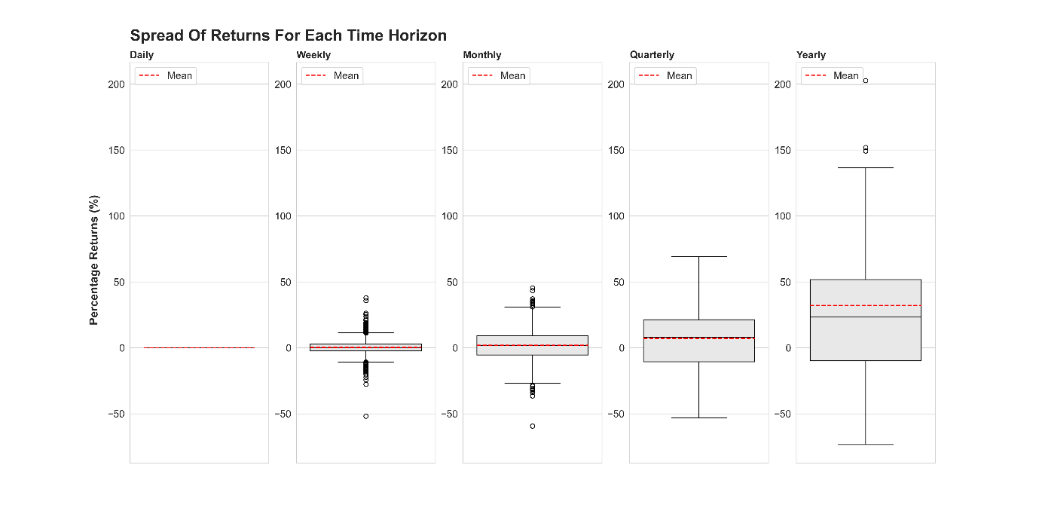
In this section we will analyse and show findings of our stock dataset, we go into details about descriptive statistics as well as an in-dept look at abnormal events with our dataset and what we can infer from them.

Our dataset consists of historical stock price data for Apple (ticker symbol: AAPL), the data was sourced through yfinance (Aroussi, n.d.) which is an open-source tool that uses Yahoo’s publicly available API. It allows you to fetch historical stock data using python by importing it as a module and passing the ticker symbol as a parameter and start and end date. This assignment analyses the daily resolution data from December 1980 to November 2024 excluding bank holidays and weekends as there are no trades during these days. The dataset contains approximately 11,000 rows with features such as Date, Closing price of the stock, Adjusted Close which is the main metric we will be focusing on as it provides a more accurate representation of a stock’s value over time. Unlike the regular closing price, which simply reflects the last traded price of a stock on a given day, the adjusted closing price considers the various corporation actions that can affect a stock price such as dividend or corporate adjustments, other features the opening price the high and low price during the day. Our target variable for this assignment will be the ‘Adjusted Close’ as this is more representative of the stock after actions like stock splits.

The first step was to generate new features from existing ones which would help our stock analysis, from the existing features above we generated percentage daily returns but normalising the day to day adjusted price differences. Furthermore, we also calculate short-term moving averages such as SMA20 and SMA50 which is a widely used technical indicator (Mitchell, June 2024) that smooths out price trends by filtering out noise from short term price fluctuations.

The figure below shows the growth of Apple’s daily adjusted closing price from 1980-2024

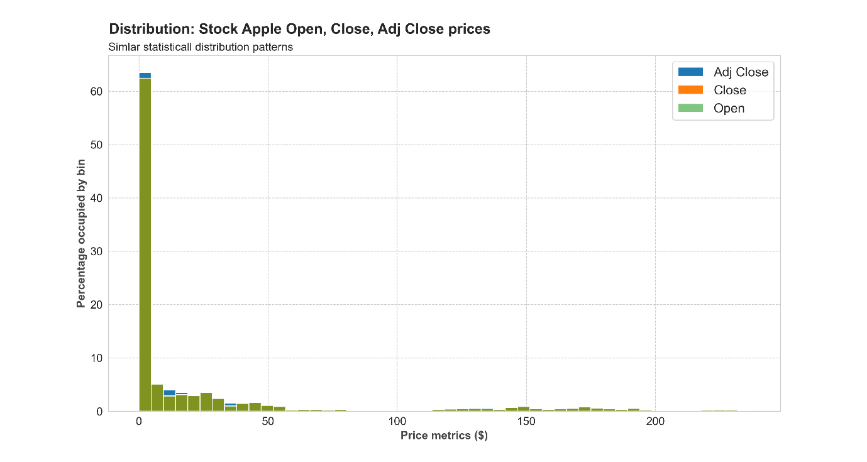
This plot shows that the long-term upward trend especially from 2005 onwards particularly accelerated growth in recent years as well as highlighting periods significant price movements with the release of iPhone and iPad as well as the expansion of the Apple ecosystem and other factors such as COVID-19 influencing the market.

Our temporal analysis of returns through box plot visualisation demonstrates a systematic increase in return dispersion across progressively longer time horizons.

The distribution ranges from relatively contained daily fluctuations to substantially wider annual return spreads (-55% to +200%), with average returns of 30%. With daily and weekly returns of 0% to 5% with more values outside the lower and upper tails, these values should not be considered outliers as these values reflect sudden changes in the stock market.

## Data pre-processing and feature selection

The raw AAPL stock data required several preprocessing steps to ensure data quality and reliability for our analysis. Our initial inspection of the dataset revealed the need for systematic cleaning. We addressed for missing values and duplicates where in this case there were none, the data from yfinance gathers data from yahoo stock API and has some preprocessing built in to handle these null values. Furthermore, to assess data quality further we examined the statistical distribution for each feature, with the focus of finding potential anomalies. These price metrics should exhibit similar statistical patterns and distributional A graph of a graph

Description automatically generatedshapes, serving as a key quality check for our financial time series data.

The distribution analysis of Open, Close and Adjusted Close price reveals strong alignment in their patterns, which reflects the natural continuity of stock trading where each day’s opening price typically corresponds to the previous day’s closing price. While our initial histogram using 50 bins demonstrates this relationship, the granularity of these patterns may vary with different bin selections. The Adjusted Close distribution shows subtle variations from the other metrics, attributable to its incorporation of corporate actions such as stock splits and dividend payments, as evidenced in the distribution above.

Looking deeper at the training volume distribution shows below identified some invalid negative values, upon research stock volumes cannot be negative.

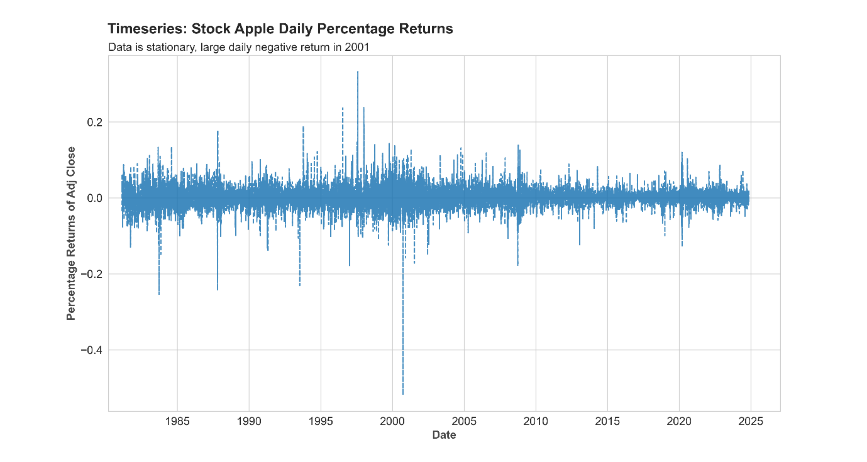
Inspecting the other records containing negative volumes, we cross-referenced our price values against external market data sources (Yahoo Finance, 2024). This validation process confirmed the negative values in the Volume column did not affect values of other features.

Additional preprocessing included, removing null values after calculating SMA20 and SMA50, we decided to drop these values as compared to the whole dataset these were a small number of values. The descriptive statistics table which summarises our dataset can be found in 01\_arima\_modelling.ipynb notebook.

## Machine learning model: ARIMA

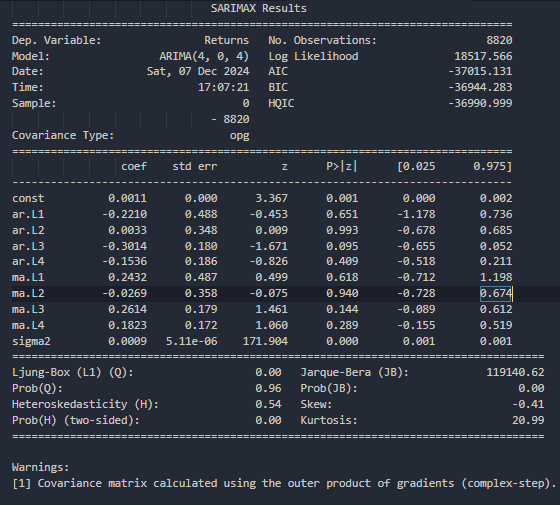
Our first approach utilizes an ARIMA model, selected for its proven effectiveness in time series prediction with its ability to capture trends, seasonality and stationary behaviour. It combines three elements AutoRegressive (AR), Differencing (I), Moving Average (MA) (Paradkar & Thakar, 2023). The AR model predicts future values based on past values in the time series, it takes in the number of lags or previous values so an AR(3) model will predict current values using three time periods before, shown in the equation below (DataCamp, n.d). . Where are constants and is the shock value at time t.

For the MA model we capture the relationships between current values and past forecast residuals, it assumes that each observation is influenced by current and previous random shocks in the data. For a MA(2) model, the equation follows . What makes the ARIMA model different is the ability to make data stationary by passing in an order of differencing meaning that we don’t have to worry about transforming data to be stationary.

A fundamental requirement of an ARIMA model implementation is data stationarity, meaning that the variance must remain constant over time with no visible trend where the data should fluctuate around a mean. Examining our features we would like to forecast the Adjusted Close however the data shows a clear upward trend, indicating non-stationarity. To test if this is statistically significant, we can run the Augmented Dickey-Fuller (ADF) test, which returns p-values and critical values to measure stationarity. A lower p-value indicates stronger evidence for stationarity. Running the ADF test returns a p-value of 1 meaning that we are statistically confident in its non-stationarity.

To create a more suitable for our model, we’ll calculate Returns by measuring the percentage change between consecutive Adjusted Close prices. This transforms our non-stationary price data into stationary return data shown below making it better for forecasting.

Running the ADF test on the Returns gives us a p-value of 0 meaning that we are confident that our data is stationary.

A screenshot of a graph

Description automatically generatedTo evaluate the model’s predictive performance later, we implement a train-test split of the APPL stock price time series, given that out data is a time series we sample chronologically. Moreover, forecasting predictions tend to decrease in accuracy overtime hence we take several test data points separate which results in a training data being 80% of the original dataset. The ARIMA(4, 0, 4) fitted model returns information criteria that indicates a strong model performance. The information criterion values BIC, AIC and HQIC suggest a robust model fit. While both BIC and HQIC penalise model complexity, we prioritise HQIC and AIC for our analysis focusing on predictive accuracy. The next step was to look at the diagnostic plot.

The standardised residual plot shows relatively consistent variance around zero overtime with some notable spikes indicating occasional extreme values which is expected when dealing with stock data. The Q-Q plot indicates that the residuals are not a perfect normal distribution as there are heavy tails on both ends suggesting excess kurtosis which upon research (Wikipedia, 2024) is reflected by the high Jarque-Bera statistic (1.2x105); I believe this is because due to AAPL’s sudden growth post 2000 which had more extreme returns going forward. The histogram of residuals shows heavy tails with a much higher peak than a gaussian normal distribution. The correlogram demonstrates no significant autocorrelation in residuals with values within the confidence bounds as reflected in the Ljung-Box test Q-statistic of 0 (Wikipedia, 2024).

The next step was to optimise the ARIMA model’s performance, we conducted hyperparameter tuning by evaluating different combinations of the autoregressive and the moving average parameters using a grid search approach (found in models/optimal\_arima\_model.py). After evaluating the models, we learned that we already had the optimised parameters which were p=4, d=0, q=4. The model’s forecasting capabilities were measured using metrics and visually as shown below.

A graph with a line graph

Description automatically generated with medium confidenceThe model seems to do well at predicting the overall trend of long-term growth for a stock but suffers when it comes to reacting to volatility in the stock market.

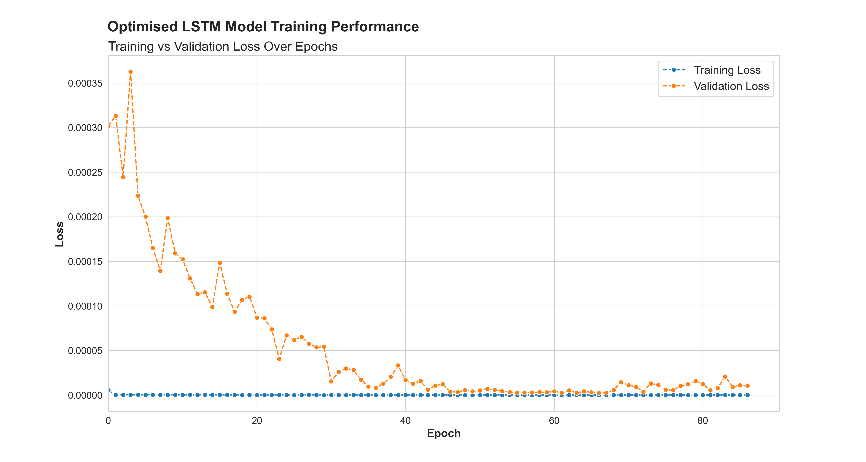
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Model | RMSE | MSE | MAPE | MAE |
| ARIMA(4,0,4) | 24.07 | 579.54 | 14.7 | 16.4 |

From these metrics suggest that out model is not performing well at predicting the test data being on average 15% away from the true value, these results are completely acceptable for an ARIMA model.

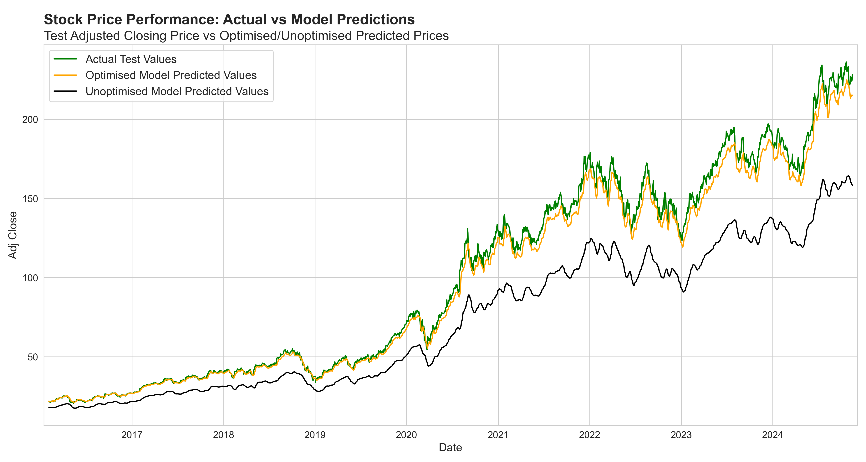
## Deep Learning model: LSTM

For deep learning we implemented a Long Short-Term Memory (LSTM) network a type of RNN, which should capture complex non-linear patterns in the AAPL stock prices. The LSTM architecture is particularly suited for this time series forecasting task due to its ability to remember information over long periods of time which gives them an edge over traditional RNNs which fail due to the vanishing gradient problem. This ability to incorporate mechanisms that allow the network when to remember and when to forget is why I believe the LSTM would be suitable for this task.

Before our model’s training process, we scale the Close values between 0 and 1, we then generate overlapping sequences as the inputs of the model each sequence contains past 10 trading values from time and the next row will contain 10 values from .

To start we create a sequential network with an LSTM layer with 50 nodes, followed by a Dropout layer of 20% repeated one more time then followed by an output layer with one node. The dropout help prevent overfitting as they randomly drop 20% of the nodes every iteration which forces neurons to learn more robust features. For out optimiser we use Adam which combines the main ideas from techniques such as momentum and RMSprop (DataCamp, 2024). Some of the key advantages include adapting the learning rate for each parameter, which can speed up learning in many cases, inclusion of bias correction terms, which help it perform well even in the initial stages of training. Finally, we set our loss function to Mean squared error and use early stopping as our callback function which monitors the model’s performance on validation data during training and stops training when the validation metric stops improving saving the weights of the best performing epoch preventing overfitting (TensorFlow, 2024).

The training validation loss plot reveals several key insights about our optimized LSTM model’s learning using Random Search. The training loss is stable and efficient at learning, this could be because the inputs are sequential lags hence the model’s strong and effortless capability to learn patterns. The validation loss shows a clear learning progression starting from higher loss values with fluctuations and steadily decreasing and stabilising around 60 epochs. The convergence pattern suggests that the model has learnt well as there is no significant gap between training and validation loss in later epochs.



The comparison between actual and predicted AAPL stock price reveals significant improvements from the optimised model, both models capture the overall growing trend with varying accuracy. Where predictions get worse overtime due to noise from previous values.

Evaluating metrics between predicted models results and true values:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Model Type | MSE | RMSE | MAPE | MAE |
| Unoptimised | 0.0297 | 0.1459 | 26.0 | 0.119 |
| Optimised | 6.458x10-4 | 0.0254 | 3.60 | 0.019 |

The optimised model achieves a mean absolute percentage error of 3.6% compared to 26%. This suggests that our predictions deviate, on average by 3.6% from the actual value.

## Comparisons across the models built

Over both models we decide to forecast 3206 daily datapoints and compare predictions:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Model Type | MSE | RMSE | MAPE | MAE |
| Optimised ARIMA | 24.070 | 579.54 | 14.7 | 16.37 |
| Optimised LSTM | 6.458x10-4 | 0.0254 | 3.60 | 0.019 |

The LSTM model demonstrates superior performance across all metrics, it captures trends across time well and can handle the inherent volatility reasonably. The ARIMA model produces a smooth forecast that follows the general direction of the price movements but does not capture the volatility of the test data over long periods.

## Conclusion, recommendations, and future work

Comparing LSTM and ARIMA models for AAPL stock price predictions from 2016 reveals distinct strengths and limitations. LSTM significantly outperformed ARIMA across all metrics but was computationally much more expensive to train and tune however the ARIMA model was much simpler to train and tune but only provided the rough direction of growth without any volatility. Based on our analysis I would recommend LSTM for short term price predictions but use ARIMA to get an idea for the direction of growth. To improve our work, we could spend more time tuning our model in addition to integrating more features such as news indicators as Boolean values which will help forecast sharp growth during predictions. While both models demonstrated strong predictive capabilities it is important to note that unpredictable nature of the stock market presents ongoing challenges for any forecasting approach.

## References

Chen, J. (2024, October 10). Forces That Move Stock Prices. Investopedia. <https://www.investopedia.com/articles/basics/04/100804.asp>

Bajaj Broking. (2024, November 28). What are the Factors Affecting Share Prices? <https://www.bajajbroking.in/blog/factors-affecting-share-prices>

Kaniugu, H. (2023, June 24). The power of time series analysis in stock prediction. LinkedIn. <https://www.linkedin.com/pulse/power-time-series-analysis-stock-prediction-heri-kaniugu/>

Aroussi, R. (n.d.). yfinance: Download market data from Yahoo! Finance's API. GitHub. <https://github.com/ranaroussi/yfinance>

Mitchell, C. (2024). How to Use a Moving Average to Buy Stocks. Investopedia. <https://www.investopedia.com/articles/active-trading/052014/how-use-moving-average-buy-stocks.asp>

Yahoo Finance. (2024). Apple Inc. (AAPL) Historical Data. Retrieved from <https://uk.finance.yahoo.com/quote/AAPL/history/>

Paradkar, M., & Thakar, C. (2023). Forecasting Stock Returns Using ARIMA Model. QuantInsti Blog. <https://blog.quantinsti.com/forecasting-stock-returns-using-arima-model/>

DataCamp. (n.d.). ARIMA Models in Python: ARMA Models [Online course]. <https://campus.datacamp.com/courses/arima-models-in-python/arma-models-1>

Wikipedia. (2024). Jarque-Bera test. <https://en.wikipedia.org/wiki/Jarque%E2%80%93Bera_test>

Wikipedia. (2024). Ljung-Box test. <https://en.wikipedia.org/wiki/Ljung%E2%80%93Box_test>

DataCamp. (2024). Adam Optimizer Tutorial. <https://www.datacamp.com/tutorial/adam-optimizer-tutorial>

TensorFlow. (2024). tf.keras.callbacks.EarlyStopping. <https://www.tensorflow.org/api_docs/python/tf/keras/callbacks/EarlyStopping>