

**Stock Market Prediction using Time Series Analysis and Neural Networks**

Misati Nyambane - 670145

Bethlehem Kebede - 670549

Hoziyana Rachael - 670448

Group 5

MTH1060: Analytical Computational Foundation

Dr. Francis Oketch

July 29, 2024

# Abstract

Stock market prediction has captivated the interest of investors, financial analysts, and researchers for decades. The rapid development of machine learning and advanced statistical techniques has introduced new methods for forecasting stock prices. This study investigates the efficacy of two prominent predictive models: the Autoregressive Integrated Moving Average (ARIMA) and Feedforward Neural Networks (FNNs), in forecasting stock prices for 20 different companies.

ARIMA models, which combine Autoregression (AR), Integration (I), and Moving Average (MA) components, are well-regarded for their ability to analyze and forecast time series data. The AR component leverages past values to predict future trends, while the I component ensures stationarity through differencing, and the MA component addresses the dependency between observations and residual errors (Box et al., 2015).

In contrast, Feedforward Neural Networks (FNNs) are a class of artificial neural networks inspired by the human brain's structure and function. These networks process information unidirectionally, from the input layer through hidden layers to the output layer, without any feedback loops (Goodfellow et al., 2016).

This paper employs both ARIMA and FNN models to predict stock prices, comparing their performance using key metrics such as Mean Squared Error (MSE), Mean Absolute Error (MAE), and Mean Absolute Percentage Error (MAPE). The analysis reveals insights into the strengths and limitations of each model in capturing the complexities of stock price movements, ultimately providing recommendations for their practical application in financial forecasting.

Table of Contents

[Abstract 2](#_Toc173180926)

[Introduction 4](#_Toc173180927)

[Methodology 6](#_Toc173180928)

[Data Acquisition 6](#_Toc173180929)

[Data Preprocessing 8](#_Toc173180930)

[Machine Learning Models 9](#_Toc173180931)

[Times Series Analysis 9](#_Toc173180932)

[Neural Networks 10](#_Toc173180933)

[Results and Discussion 13](#_Toc173180934)

[Time Series Analysis 13](#_Toc173180935)

[Results 13](#_Toc173180936)

[Discussion 14](#_Toc173180937)

[Neural Networks 19](#_Toc173180938)

[Training 21](#_Toc173180939)

[Testing 22](#_Toc173180940)

[Validation 23](#_Toc173180941)

[Conclusion and Recommendations 24](#_Toc173180942)

[Time Series Analysis 24](#_Toc173180943)

[Neural Networks 25](#_Toc173180944)

[References 26](#_Toc173180945)

[Appendices 27](#_Toc173180946)

# Introduction

Stock market prediction has long been a topic of interest for investors, financial analysts, and researchers. With the advent of machine learning and advanced statistical techniques, new approaches to forecasting stock prices have emerged. This study explores the application of Autoregressive Integrated Moving Average (ARIMA) and Feedforward Neural Network models for predicting stock prices of 20 different companies.

ARIMA models are a class of statistical models used for analyzing and forecasting time series data. They combine three components: Autoregression (AR), which uses past values to predict future ones; Integration (I), which makes the time series stationary by differencing; and Moving Average (MA), which incorporates the dependency between an observation and a residual error from a moving average model applied to lagged observations (Box et al., 2015).

Feedforward Neural Networks (FNNs) are a type of artificial neural network, which is a machine learning model inspired by the structure and function of the human brain. FNNs are called "feedforward" because information flows in one direction, from the input layer, through any hidden layers, to the output layer, without any feedback loops or connections (Goodfellow et al., 2016)

Key terms:

* **Time Series:** A sequence of data points indexed in time order.
* **ARIMA:** Autoregressive Integrated Moving Average, a statistical analysis model that uses time series data to predict future trends.
* **MSE:** Mean Squared Error, a measure of the average squared difference between the estimated values and the actual value.
* **MAE:** Mean Absolute Error, a measure of the average magnitude of the errors in a set of predictions.
* **MAPE:** Mean Absolute Percentage Error, a measure of prediction accuracy in statistics.
* **AR (AutoRegressive) Value:** Represents the number of lagged observations included in the model. It indicates how the current value of the time series is influenced by its past values. The AR parameter is usually denoted by p in the ARIMA(p, d, q) model, where p is the order of the autoregressive part. For example, an AR(1) model uses only one lagged term.
* **MA (Moving Average) Value:** Represents the number of lagged forecast errors included in the model. It captures the influence of past forecast errors on the current value of the time series. The MA parameter is usually denoted by q in the ARIMA(p, d, q) model, where q is the order of the moving average part. For example, an MA(1) model uses only one lagged forecast error term.
* **Input Layer:** The layer that receives the input data
* **Hidden Layer:** The intermediate layers between the input and output layers, where the data is transformed and weighted
* **Output Layer:** The final layer that produces the predicted output
* **Neuron:** An interconnected node in the neural network that performs a simple computation
* **Weight:** A value that determines the strength of the connection between neurons
* **Bias:** An additional parameter that shifts the activation function of a neuron
* **Activation Function:** A function that determines the output of a neuron based on its input
* **Loss Function:** A function that measures the difference between the predicted output and the true output, which is minimized during training
* **Gradient Descent:** An optimization algorithm that adjusts the weights and biases of the network to minimize the loss function
* **Backpropagation:** An algorithm used to compute the gradient of the loss function with respect to the weights and biases of the network, which is used in gradient descent

# Methodology

## Data Acquisition

Data was obtained from Yahoo Finance. The data spans from 1st July 2014 to 28th June 2024. The data is in the general form:

*Date,Open,High,Low,Close,Adj Close,Volume*

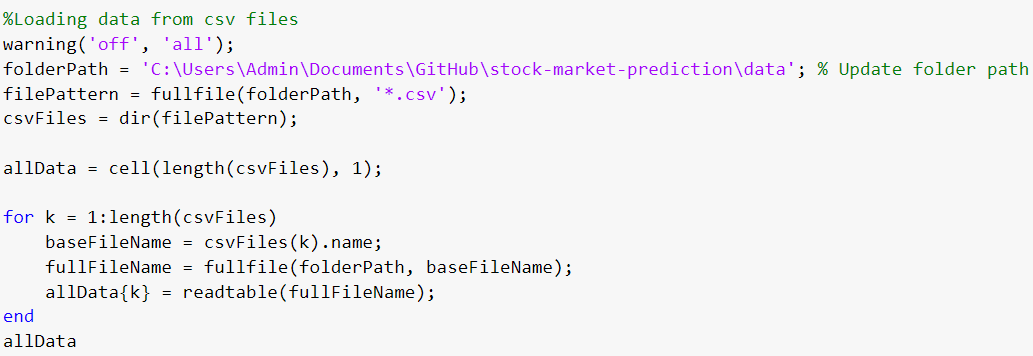
*2014-07-01,23.3799991607666,23.517499923706055,23.282499313354492,23.3799991607666,20.68042755126953,152892000*

*2014-07-02,23.467500686645508,23.514999389648438,23.272499084472656,23.3700008392334,20.671592712402344,113860000*

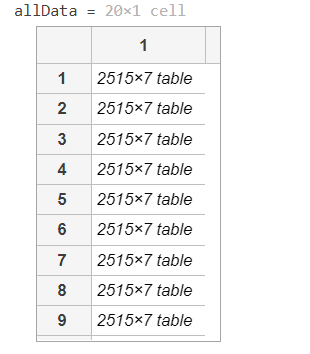
*2014-07-03,23.417499542236328,23.524999618530273,23.299999237060547,23.50749969482422,20.793209075927734,91567200*

The data was obtained using the Yahoo finance package in Python. By use of a dictionary the 20 companies selected were iterated through and their data written to CSV files. Yahoo Finance was preferred since it offers data for each company separately.

To import data the following code was used:



This code imports all CSV files from the specified folder into a cell array allData. Each element of allData contains a table with the stock data for one company.



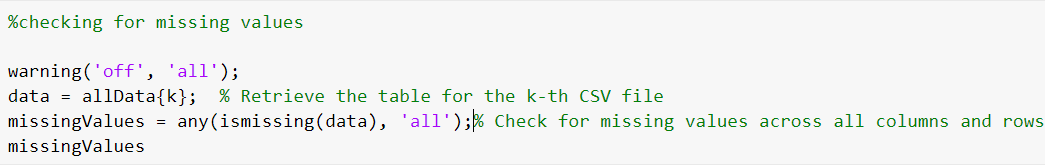
## Data Preprocessing

Data was imported into MATLAB in bulk and stored in an array. This enables the use of iteration and control structures to access the data. Warnings are suppressed for cleaner output and if there are any missing values they are pointed out.

This allows for missing values to be dropped to ensure that operations are not skewed or curtailed by this data.

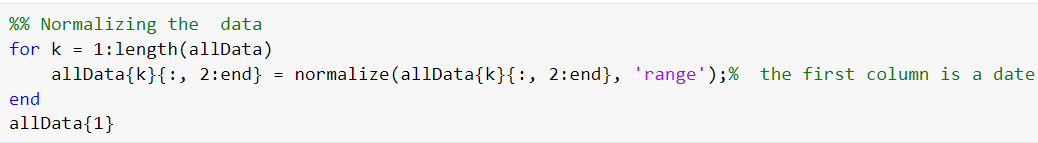
Furthermore, data is normalized between the range of 0 and 1 for our Time Series Analysis and Neural Network models to be able to work with our data. Only numerical columns are normalized. The column *date* is not affected in any way.

Data integrity was checked using:



This code checks for missing values across all columns and rows in each dataset. No missing values were found in the data sets.

Data normalization was performed using:

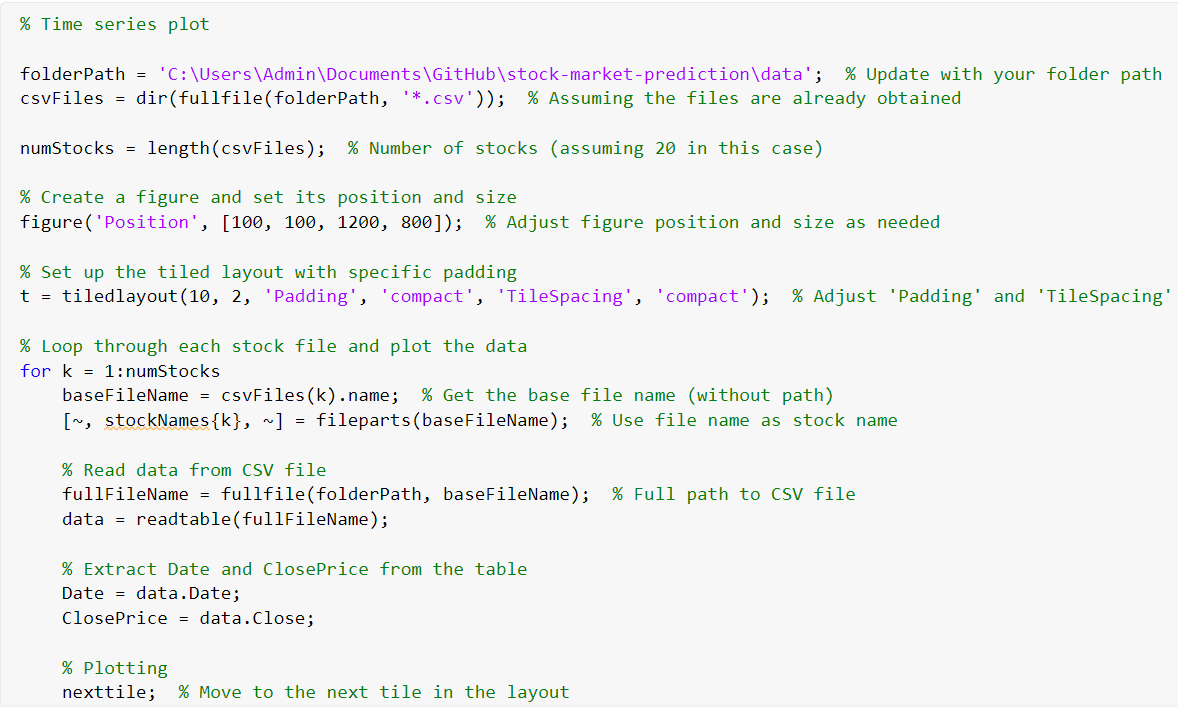


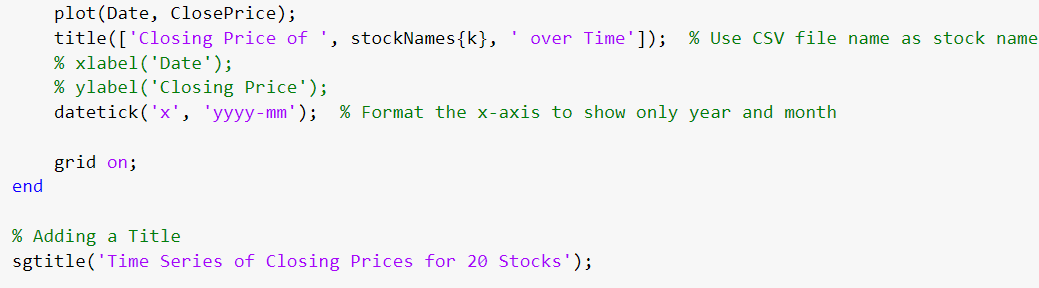
This normalizes all numerical columns to the range [0, 1], facilitating comparison between stocks with different price ranges.

## Machine Learning Models

### Times Series Analysis

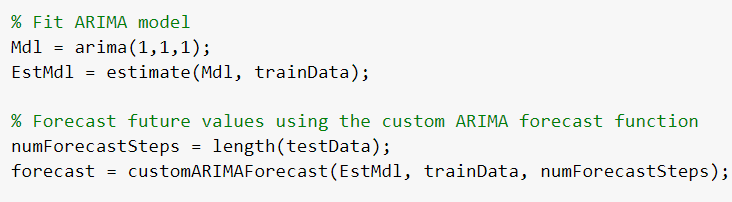
The following code was used to create time series plots for all 20 stocks:





This code creates a 10x2 grid of subplots, each showing the closing price over time for one stock.

The ARIMA(1,1,1) model was implemented for each stock using:



Here, arima(1,1,1) specifies the ARIMA model order. The estimate function fits the model to the training data, and customARIMAForecast (a custom function) generates predictions for the test period.

### Neural Networks

The neural network uses different functions to compute the values. In simplified terms a neural network is a polynomial function whose parameters we do not know. That is, we do not know the degree, variables or coefficients. Nonetheless, what we do know are the weights and biases. These weights and biases are updated using polynomial functions that differentiated and integrated as needed. Furthermore, an activation function and delta variable are employed to control the rate at which the weights and biases are updated. This modification of weights and biases is how the model ‘learns’.

#### Forward Propagation

For forward propagation the functions employed are:

The function above computes a linear combination of the weights and biases then this linear combination is transformed using a non-linear function. This enables complex relationships to be learnt.

The function receives the output from the entry node and then computes the linear combination of weights and biases which are then transformed by the Sigmoid function (defined below).

This function is the Rectified Linear Unit (ReLU) activation function which outputs a value directly if it is positive. Otherwise, it outputs zero.

This function is the Sigmoid function which maps any real-valued number into the range (0, 1).

#### Back Propagation

The function below is the loss function which calculates the Mean Squared Error (MSE) between the actual values and the predicted values. The aim is to minimize the value obtained from the loss function to get a more accurate neural network.

The function below is the derivative of the loss which indicates how much the loss changes with a small change in the predicted output. If  is greater than *y*, the derivative will be positive, suggesting that decreasing   will reduce the loss. Conversely, if   is less than *y*, the derivative will be negative, indicating that increasing  ​ will help reduce the loss. This will inform the network on the way weights and biases should be updated.

This function is the derivative with respect to weights. This term assesses the sensitivity of the output with respect to the input of the activation function. The product with *h* reflects how changes in the weights affect the output based on the current output from the first layer. This derivative is then employed in backpropagation to adjust the weights in the direction that reduces the loss.

The last function is the derivative with respect to bias. Like the derivative with respect to weights function this one is used to adjust the bias in the direction that minimizes the loss.

#### Gradients

The functions are the derivatives with respect to the hidden layer output, weights and bias. These derivatives are then used to update their respective variables. This backpropagation aims to reduce the error between the predicted and the actual values.

#### Weight Update Rules

The weights are then updated using the following functions to minimize the value obtained from the loss function.

#### Bias Update Rules

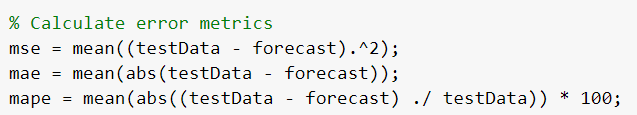
Lastly, the biases are updated.

# Results and Discussion

## Time Series Analysis

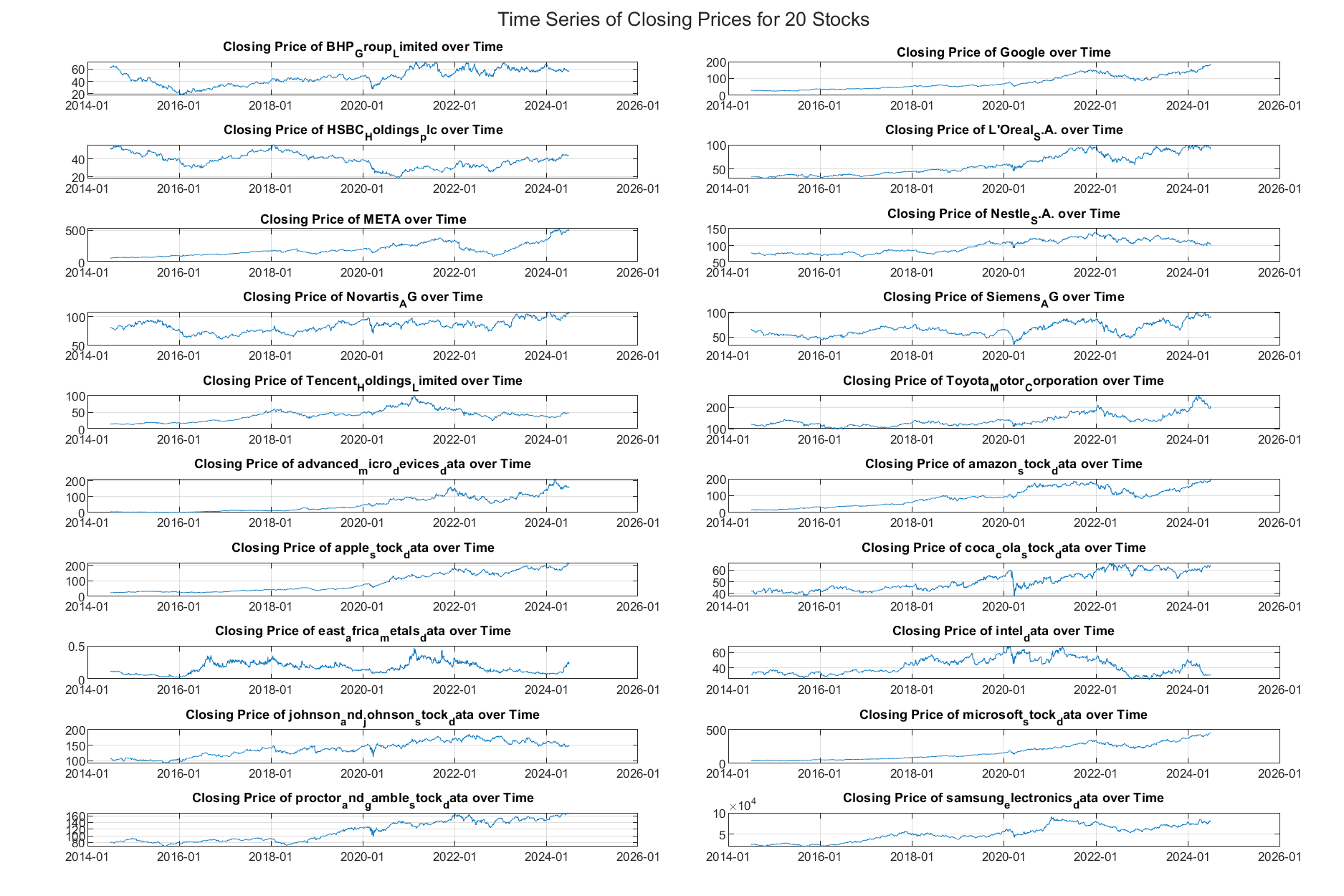
### Results

Performance metrics were calculated using:



These metrics provide different perspectives on the model's prediction accuracy.

The time series plots (Image 1) provide valuable insights into the price movements of the 20 stocks over the period from 2014 to 2024:



### Discussion

#### Key Observations

1. Overall Market Trend: Most stocks show a general upward trend from 2014 to 2024, indicating broad market growth over this decade. This aligns with the overall bullish trend in global markets during this period.
2. Tech Stock Performance: Technology stocks like Google, Amazon, Microsoft, and Apple demonstrate particularly strong and consistent growth. This reflects the increasing dominance of the tech sector in global markets.
3. Volatility Differences: Some stocks, such as Intel and Johnson & Johnson, show more pronounced fluctuations, indicating higher volatility. In contrast, stocks like Nestlé and Procter & Gamble display more stable price movements, typical of consumer staples companies.
4. COVID-19 Impact: A sharp decline is visible across most stocks around early 2020, corresponding to the onset of the COVID-19 pandemic. However, the recovery patterns vary:

* Tech stocks like Amazon and Microsoft show rapid recovery and accelerated growth post-dip.
* Traditional industries like Toyota and Johnson & Johnson show a slower, more gradual recovery.

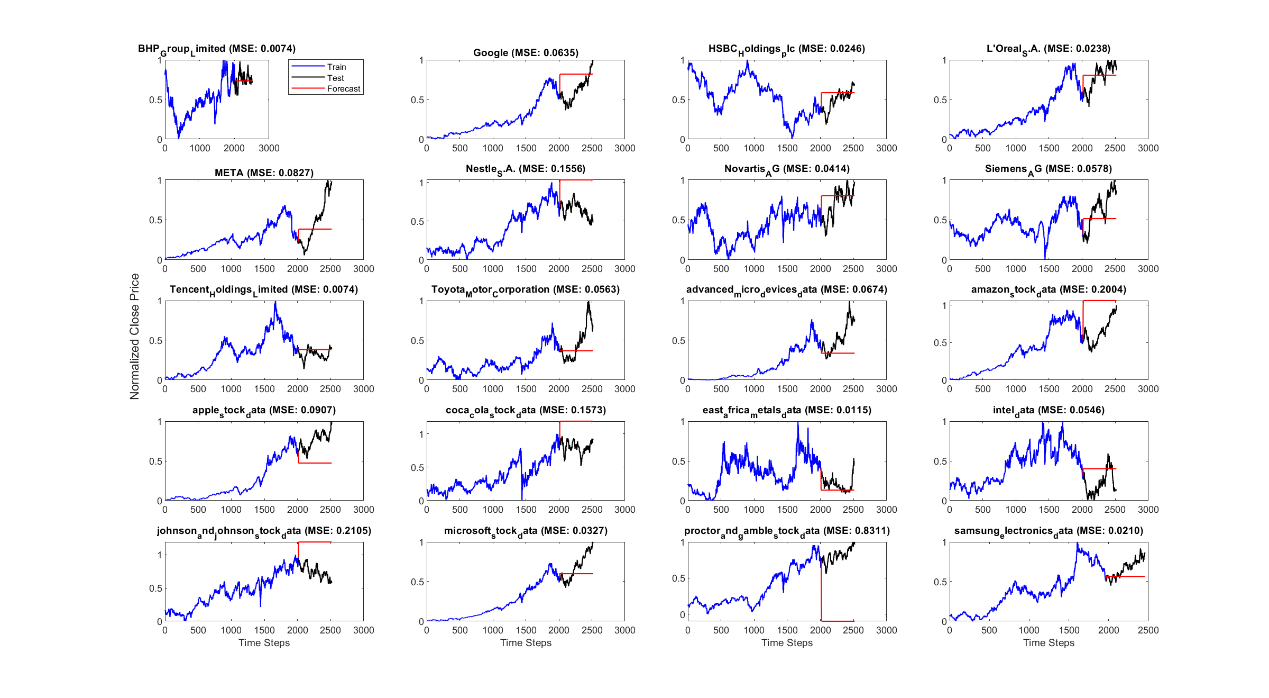
1. Sector-Specific Trends:

* Financial stocks (e.g., HSBC) show more cyclical patterns.
* Consumer goods companies (e.g., Coca-Cola, Procter & Gamble) display relatively stable growth with less pronounced peaks and troughs.
* Pharmaceutical companies (e.g., Novartis) show steady growth, likely influenced by healthcare sector dynamics.

1. Regional Variations: Stocks from different regions (e.g., Tencent from China, Samsung from South Korea) show some unique patterns, possibly reflecting local economic conditions and regulations.

#### Model Performance

The ARIMA(1,1,1) model performance varied significantly across the 20 stocks analysed:



1. Best Performing Stocks:
   * BHP Group Limited (MSE: 0.0074)
   * Tencent Holdings Limited (MSE: 0.0074)
   * L'Oreal S.A. (MSE: 0.0238)

For these stocks, the ARIMA model's predictions closely align with actual price movements. The low MSE values indicate that the model captured the underlying trends and patterns effectively. These stocks likely have more predictable price movements or stronger adherence to historical patterns.

1. Worst Performing Stocks:
   * Proctor & Gamble (MSE: 0.8311)
   * Coca-Cola (MSE: 0.1573)
   * Johnson & Johnson (MSE: 0.2105)

The high MSE values for these stocks suggest that the ARIMA model struggled to accurately predict their price movements. This could be due to higher volatility, more frequent external shocks, or complex underlying patterns that a simple ARIMA(1,1,1) model cannot capture adequately.

The performance distribution exhibited the following characteristics:

* The wide range of MSE values (from 0.0074 to 0.8311) indicates that the ARIMA model's performance is highly stock-specific.
* Stocks with MSE values below 0.05 (e.g., BHP, Tencent, L'Oreal, HSBC) show good model fit.
* Stocks with MSE values above 0.1 (e.g., Proctor & Gamble, Coca-Cola, Johnson & Johnson) indicate poor model fit.

In terms of sector-based performance it is evident that:

* The model seems to perform better for some technology and financial stocks.
* Consumer goods companies (Proctor & Gamble, Coca-Cola) show poorer performance, possibly due to their more stable price movements which might not benefit from the ARIMA model's strength in capturing trends and seasonality.

By visual analysis:

* For well-performing stocks, the forecast line (red) closely follows the actual price line (black) in the test period.
* Poorly performing stocks show significant divergence between forecast and actual prices, often with the model failing to capture rapid price changes or reversals.

#### Model Parameters and Statistical Significance

Examining the ARIMA model outputs for each stock reveals interesting patterns. The following is a sample of the outputs of the ARIMA model.

ARIMA(1,1,1) Model (Gaussian Distribution):

**Value** **StandardError** **TStatistic** **PValue**

**\_\_\_\_\_\_\_\_\_\_\_** **\_\_\_\_\_\_\_\_\_\_\_\_\_** **\_\_\_\_\_\_\_\_\_\_** **\_\_\_\_\_\_\_**

**Constant** -3.6643e-05 0.00092815 -0.039479 0.96851

**AR{1}**  2e-12 22.291 8.9722e-14 1

**MA{1}**  0.00075248 22.291 3.3757e-05 0.99997

**Variance** 0.00028467 5.4987e-06 51.771 0

The key observations are:

1. AR and MA Coefficients:

* AR{1} coefficients range from highly negative (e.g., -0.65488 for L'Oreal) to positive (e.g., 0.63026 for Samsung Electronics).
* MA{1} coefficients show similar variability, with values ranging from -0.71849 (Samsung Electronics) to 0.61574 (L'Oreal).
* The variability in these coefficients suggests different underlying price dynamics across stocks.

1. Statistical Significance:

* Many stocks show statistically significant AR{1} and MA{1} coefficients (p-value < 0.05), indicating that these components contribute meaningfully to the model.
* The constant term is often not statistically significant (p-value > 0.05), suggesting that the differencing component adequately captures the trend in most cases.

1. Variance:

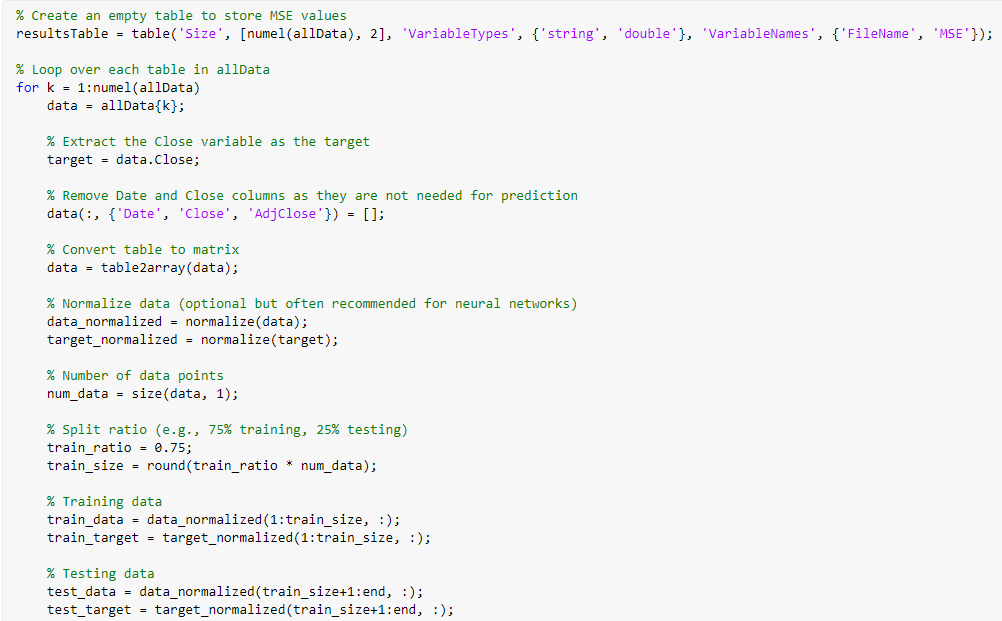
* The variance term is consistently small and statistically significant across all stocks, indicating that the model captures a significant portion of the price variability.

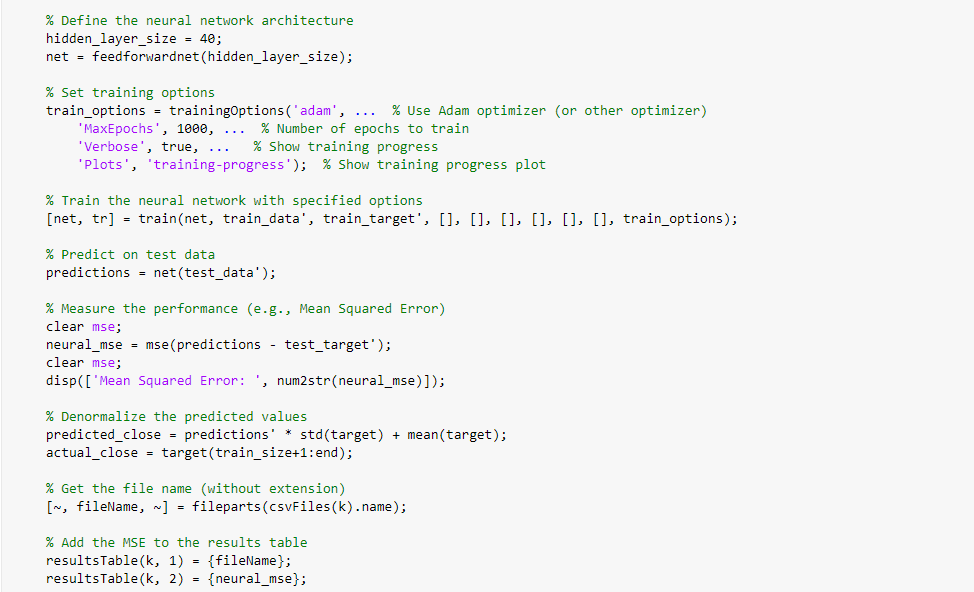
1. Model Interpretation:

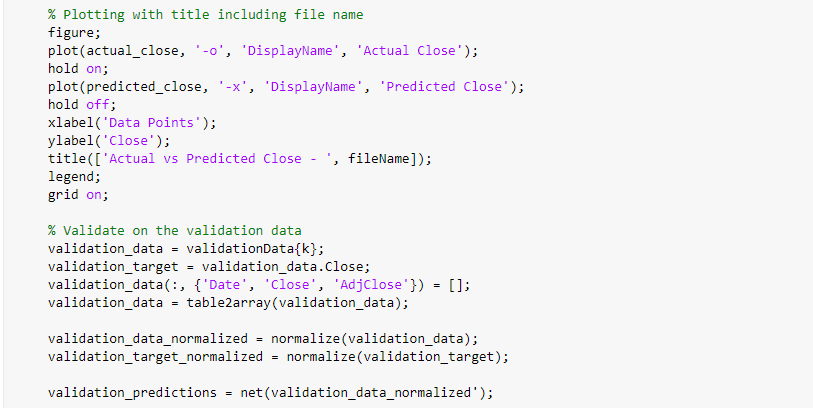
* Negative AR{1} coefficients suggest a tendency for the series to oscillate, while positive coefficients indicate momentum in price movements.
* The MA{1} term captures the impact of past forecast errors, with its sign and magnitude indicating how quickly the model adjusts to these errors.

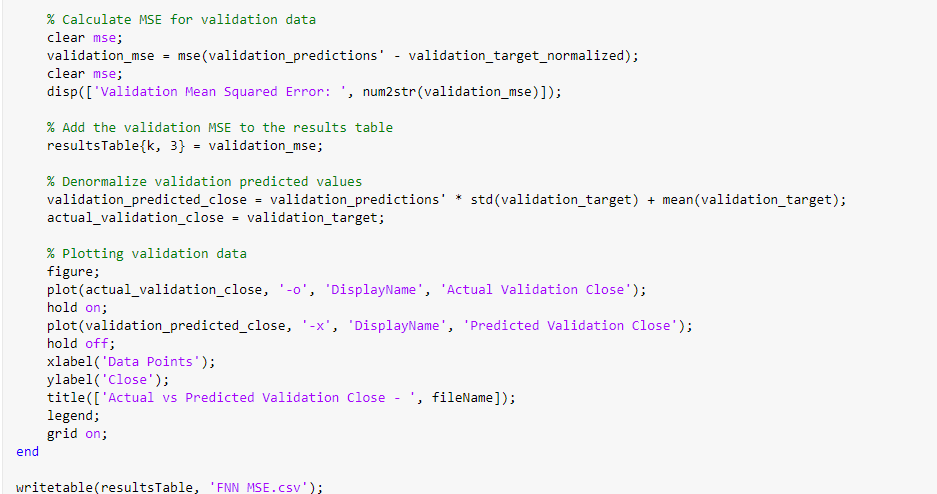
## Neural Networks

The code that accomplished this uses a loop to create a neural network for each stock index in the allData array. The code is as below:









### Training

We decided to use a Feedforward Neural Network that:

* Has 40 hidden nodes
* Trains for a maximum of 1000 epochs
* Uses the adam optimizer
* Utilizes 75% of the data to train
* Utilizes 25% of the data to test
* Uses the Mean Squared Error to test the error of the results
* Utilizes validation data from 1st July 2024 to 26th July 2024

### Testing

In testing the MSE’s obtained were as detailed in the table below.

|  |  |
| --- | --- |
| **Company** | **MSE** |
| HSBC Holdings | 0.000289854 |
| Tencent Holdings Limited | 0.000328486 |
| Samsung Electronics | 0.000384859 |
| Amazon | 0.000831416 |
| Alphabet | 0.000862567 |
| Nestle | 0.000945337 |
| Intel | 0.001157845 |
| Proctor and Gamble | 0.001207186 |
| BHP Group Limited | 0.001295159 |
| East Africa Metals | 0.003050332 |
| AMD | 0.005859265 |
| Coca Cola | 0.006657724 |
| Apple | 0.007479319 |
| Novartis AG | 0.007599726 |
| Siemens AG | 0.021622344 |
| L'Oreal | 0.026000365 |
| Microsoft | 0.041602214 |
| Johnson and Johnson | 0.077928339 |
| Meta | 0.123486883 |
| Toyota Motor Corporation | 0.156136595 |

The smallest MSE is the one from HBSC Holdings at 0.000289854 and the largest is by Toyota Motor Corporation at 0.156136595. An interesting trend that emerges that for companies that:

1. Have a wide range of products for sale.
2. Were not affected by the AI boom.
3. Were not really affected by COVID 19.

The model can use data for the first seven and a half years to predict the remaining two and a half years. This explains why Samsung Electronics or Alphabet despite being technological companies have such low MSEs.

### Validation

On the other hand, when it comes to validation HBSC Holdings maintains its position as the one with the most accurate FNN, yet Intel moves from somewhere in the middle to being the one with the least accurate FNN. Toyota Motor Corporation curiously has a smaller MSE over the brief validation time frame. This means that its stock must be volatile and best assessed over shorter periods. This same trend occurs for Meta, Johnson and Johnson, and Microsoft also.

|  |  |
| --- | --- |
| **Company** | **Validation MSE** |
| HSBC Holdings | 0.011285854 |
| Meta | 0.014930324 |
| Alphabet | 0.015347291 |
| Toyota Motor Corporation | 0.017593062 |
| Microsoft | 0.020737847 |
| Apple | 0.024885528 |
| Siemens AG | 0.028157622 |
| Coca Cola | 0.028276792 |
| Tencent Holdings Limited | 0.03699829 |
| Samsung Electronics | 0.041988428 |
| Amazon | 0.044326173 |
| East Africa Metals | 0.049413927 |
| Novartis AG | 0.049855178 |
| Nestle | 0.051387616 |
| BHP Group Limited | 0.072831488 |
| Proctor and Gamble | 0.102384032 |
| L'Oreal | 0.126287847 |
| Johnson and Johnson | 0.154387357 |
| AMD | 0.174740765 |
| Intel | 0.483313323 |

# Conclusion and Recommendations

## Time Series Analysis

1. Model Simplicity: The ARIMA(1,1,1) model, while effective for some stocks, may be too simplistic for others. More complex ARIMA models (with different p, d, q parameters) might yield better results for some stocks.
2. Stationarity Assumptions: The ARIMA model assumes that the differenced series is stationary. This assumption may not hold for all stocks, particularly those with changing volatility over time.
3. External Factors: The model does not account for external factors (e.g., market sentiment, economic indicators, company-specific news) that can significantly impact stock prices.
4. Short-term vs. Long-term Predictions: The model's performance in short-term predictions may differ from its long-term forecasting ability. This aspect needs further investigation.
5. Data Normalization Impact: While normalization allows for easier comparison across stocks, it may obscure some stock-specific characteristics that could be relevant for prediction.

In conclusion, the ARIMA(1,1,1) model shows varied performance across different stocks, highlighting the complexity of stock price prediction. While it performs exceptionally well for some stocks, it struggles with others, underscoring the need for stock-specific modelling approaches and the potential inclusion of additional predictive factors.

## Neural Networks

The FNN:

1. Has low values for every stock index. The value is either 0.x or 0.0x or even 0.00x and in cases it goes down to 0.000x. This illustrates that when used in a real-world context the model will be precise in predicting the values hence will be generally profitable.
2. Brings to the light the effect of confounding factors on the stock market trends. This makes evident that for certain volatile stock- that is stock with a wide variety of confounding factors in comparison to the diversity of goods sold- when the contextual data is briefer it is much more accurate. For such stock if the data spans a large time frame it underfits.
3. Illustrates that for less volatile stock the neural network does extremely well when the data spans long periods of time.
4. Displays that there exist outliers that have very stable stocks like HBSC Holdings. When trained over long periods of time it performs just as well in short term prediction.

In conclusion the methodology used to train the model is extremely useful in identifying stable stock indices that would be invaluable in growing one’s money predictably. It also identifies stock indices that are riskier yet will return a higher Return on Investment (ROI). And our model could be an invaluable key to this process of investing.

# References

Box, G. E., Jenkins, G. M., Reinsel, G. C., & Ljung, G. M. (2015). *Time series analysis: forecasting and control*. John Wiley & Sons.

Hyndman, R. J., & Athanasopoulos, G. (2018*). Forecasting: Principles and Practice*. OTexts.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT press.

Chollet, F., & Allaire, J. J. (2021). *Math for deep learning: What you need to know to understand neural networks*. O'Reilly Media.

# Appendices

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Company** | **Testing MSE** | **Validation MSE** | | **Difference** |
| HSBC Holdings | 0.000289854 | 0.011285854 | 0.010996 | |
| Tencent Holdings Limited | 0.000328486 | 0.03699829 | 0.036669804 | |
| Samsung Electronics | 0.000384859 | 0.041988428 | 0.041603569 | |
| Amazon | 0.000831416 | 0.044326173 | 0.043494757 | |
| Alphabet | 0.000862567 | 0.015347291 | 0.014484724 | |
| Nestle | 0.000945337 | 0.051387616 | 0.050442279 | |
| Intel | 0.001157845 | 0.483313323 | 0.482155478 | |
| Proctor and Gamble | 0.001207186 | 0.102384032 | 0.101176846 | |
| BHP Group Limited | 0.001295159 | 0.072831488 | 0.071536329 | |
| East Africa Metals | 0.003050332 | 0.049413927 | 0.046363595 | |
| AMD | 0.005859265 | 0.174740765 | 0.1688815 | |
| Coca Cola | 0.006657724 | 0.028276792 | 0.021619068 | |
| Apple | 0.007479319 | 0.024885528 | 0.017406209 | |
| Novartis AG | 0.007599726 | 0.049855178 | 0.042255452 | |
| Siemens AG | 0.021622344 | 0.028157622 | 0.006535278 | |
| L'Oreal | 0.026000365 | 0.126287847 | 0.100287482 | |
| Microsoft | 0.041602214 | 0.020737847 | -0.020864367 | |
| Johnson and Johnson | 0.077928339 | 0.154387357 | 0.076459018 | |
| Meta | 0.123486883 | 0.014930324 | -0.108556559 | |
| Toyota Motor Corporation | 0.156136595 | 0.017593062 | -0.138543533 | |

*Fig 1: Testing and Validation MSE’s with the Difference*

ARIMA(1,1,1) Model (Gaussian Distribution):

Value StandardError TStatistic PValue

\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_

Constant -3.6643e-05 0.00092815 -0.039479 0.96851

AR{1} 2e-12 22.291 8.9722e-14 1

MA{1} 0.00075248 22.291 3.3757e-05 0.99997

Variance 0.00028467 5.4987e-06 51.771 0

ARIMA(1,1,1) Model (Gaussian Distribution):

Value StandardError TStatistic PValue

\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_

Constant 0.00039164 0.00024744 1.5828 0.11348

AR{1} -0.46309 0.12945 -3.5772 0.00034726

MA{1} 0.39382 0.13335 2.9533 0.0031441

Variance 6.0012e-05 8.4202e-07 71.272 0

ARIMA(1,1,1) Model (Gaussian Distribution):

Value StandardError TStatistic PValue

\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_

Constant -0.0003605 0.00047079 -0.76573 0.44384

AR{1} -0.44141 0.45479 -0.97057 0.33176

MA{1} 0.41146 0.4621 0.89041 0.37324

Variance 0.00020679 4.2404e-06 48.767 0

ARIMA(1,1,1) Model (Gaussian Distribution):

Value StandardError TStatistic PValue

\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_

Constant 0.00036578 0.00036635 0.99845 0.31806

AR{1} -0.41885 0.16408 -2.5527 0.010689

MA{1} 0.35417 0.16881 2.098 0.035902

Variance 0.00014274 2.3214e-06 61.49 0

ARIMA(1,1,1) Model (Gaussian Distribution):

Value StandardError TStatistic PValue

\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_

Constant 0.00017365 0.00036545 0.47517 0.63467

AR{1} -0.56642 0.15401 -3.6779 0.00023517

MA{1} 0.5128 0.16144 3.1765 0.0014907

Variance 9.5612e-05 5.7214e-07 167.11 0

ARIMA(1,1,1) Model (Gaussian Distribution):

Value StandardError TStatistic PValue

\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_

Constant 0.00038733 0.00045561 0.85015 0.39524

AR{1} -0.46608 0.24275 -1.92 0.054859

MA{1} 0.42689 0.2488 1.7158 0.086193

Variance 0.00019259 3.2377e-06 59.483 0

ARIMA(1,1,1) Model (Gaussian Distribution):

Value StandardError TStatistic PValue

\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_

Constant 5.2094e-05 0.00068235 0.076344 0.93915

AR{1} -0.50743 0.15567 -3.2595 0.0011159

MA{1} 0.45553 0.1608 2.8329 0.0046128

Variance 0.00042192 7.0103e-06 60.185 0

ARIMA(1,1,1) Model (Gaussian Distribution):

Value StandardError TStatistic PValue

\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_

Constant -0.00011657 0.00050719 -0.22984 0.81821

AR{1} -0.51496 0.32328 -1.5929 0.11118

MA{1} 0.48085 0.33158 1.4502 0.147

Variance 0.00022999 4.1866e-06 54.935 0

ARIMA(1,1,1) Model (Gaussian Distribution):

Value StandardError TStatistic PValue

\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_

Constant 0.00020324 0.00030648 0.66315 0.50723

AR{1} -0.090296 0.11666 -0.77398 0.43895

MA{1} -0.024245 0.11727 -0.20675 0.8362

Variance 0.0001877 2.3863e-06 78.657 0

ARIMA(1,1,1) Model (Gaussian Distribution):

Value StandardError TStatistic PValue

\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_

Constant 0.00010889 0.00029901 0.36417 0.71573

AR{1} 2e-12 1.1126 1.7976e-12 1

MA{1} -0.012728 1.1126 -0.01144 0.99087

Variance 0.00013663 2.3304e-06 58.632 0

ARIMA(1,1,1) Model (Gaussian Distribution):

Value StandardError TStatistic PValue

\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_

Constant 0.00017431 0.00015871 1.0982 0.2721

AR{1} 0.10152 0.12371 0.82065 0.41185

MA{1} -0.17279 0.12285 -1.4065 0.15957

Variance 7.3553e-05 9.2561e-07 79.464 0

ARIMA(1,1,1) Model (Gaussian Distribution):

Value StandardError TStatistic PValue

\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_

Constant 0.00043083 0.0004029 1.0693 0.28492

AR{1} -0.65488 0.14145 -4.6297 3.6621e-06

MA{1} 0.61574 0.14677 4.1954 2.7235e-05

Variance 0.00012152 1.5574e-06 78.028 0

ARIMA(1,1,1) Model (Gaussian Distribution):

Value StandardError TStatistic PValue

\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_

Constant 0.00025011 0.00016378 1.5271 0.12673

AR{1} 0.17469 0.18059 0.96736 0.33336

MA{1} -0.2336 0.17941 -1.302 0.19291

Variance 6.55e-05 9.0201e-07 72.616 0

ARIMA(1,1,1) Model (Gaussian Distribution):

Value StandardError TStatistic PValue

\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_

Constant 0.00047757 0.00063923 0.7471 0.455

AR{1} -0.33949 0.31312 -1.0842 0.27827

MA{1} 0.31457 0.31799 0.98923 0.32255

Variance 0.00040733 5.2256e-06 77.949 0

ARIMA(1,1,1) Model (Gaussian Distribution):

Value StandardError TStatistic PValue

\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_

Constant 5.9394e-05 0.00026263 0.22615 0.82108

AR{1} 0.43385 0.051651 8.3996 4.4788e-17

MA{1} -0.62812 0.043805 -14.339 1.249e-46

Variance 0.00094052 1.6617e-05 56.602 0

ARIMA(1,1,1) Model (Gaussian Distribution):

Value StandardError TStatistic PValue

\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_

Constant 0.00012732 0.0006632 0.19198 0.84776

AR{1} -0.45949 0.055396 -8.2945 1.09e-16

MA{1} 0.28742 0.058768 4.8908 1.0045e-06

Variance 0.00049473 5.9266e-06 83.477 0

ARIMA(1,1,1) Model (Gaussian Distribution):

Value StandardError TStatistic PValue

\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_

Constant 0.00052917 0.00048107 1.1 0.27134

AR{1} -0.33405 0.095636 -3.4929 0.00047782

MA{1} 0.24417 0.098429 2.4807 0.013112

Variance 0.00026932 3.6333e-06 74.124 0

ARIMA(1,1,1) Model (Gaussian Distribution):

Value StandardError TStatistic PValue

\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_

Constant 0.00034476 0.00017418 1.9794 0.047777

AR{1} -0.2838 0.060033 -4.7274 2.2736e-06

MA{1} 0.11398 0.063001 1.8091 0.07043

Variance 4.4992e-05 6.4988e-07 69.231 0

ARIMA(1,1,1) Model (Gaussian Distribution):

Value StandardError TStatistic PValue

\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_

Constant 0.0001146 9.7907e-05 1.1705 0.24181

AR{1} 0.63026 0.060467 10.423 1.941e-25

MA{1} -0.71849 0.056925 -12.622 1.6023e-36

Variance 0.00016926 2.0906e-06 80.961 0

ARIMA(1,1,1) Model (Gaussian Distribution):

Value StandardError TStatistic PValue

\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_

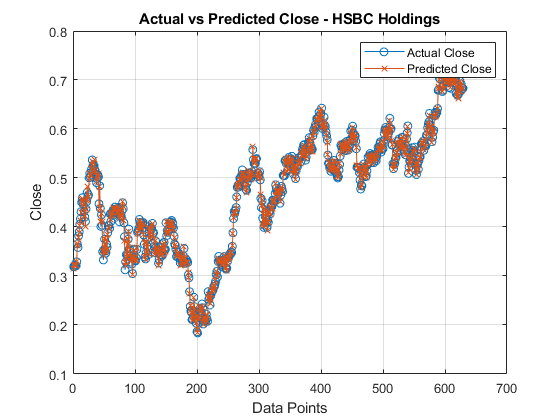
Constant 0.00023972 0.00029747 0.80584 0.42033

AR{1} -0.036543 0.5292 -0.069053 0.94495

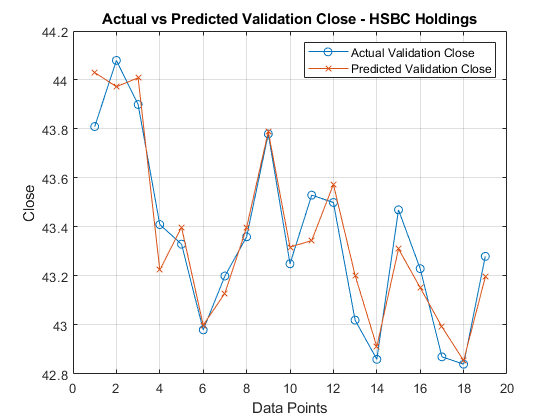
MA{1} 0.07098 0.52546 0.13508 0.89255

Variance 0.0001304 2.4457e-06 53.32 0

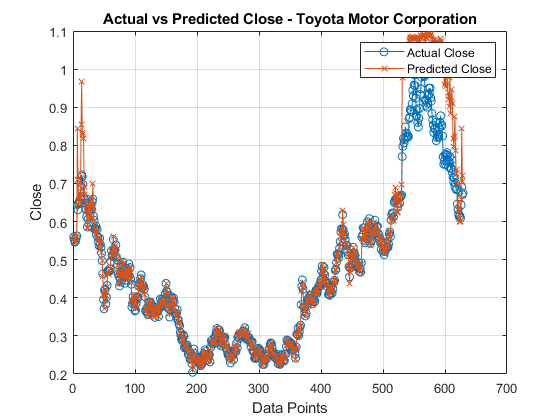
*Fig 2: ARIMA(1,1,1) Model Gaussian Distribution*



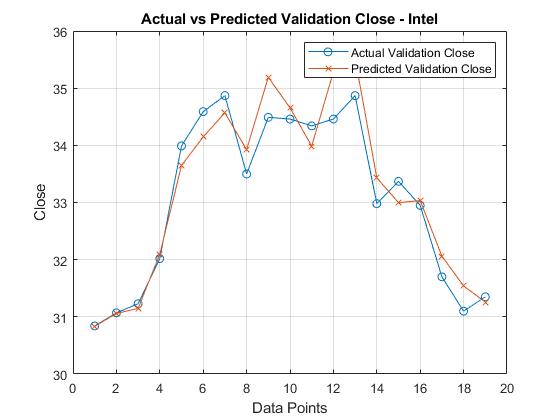
*Fig 3: HBSC Holdings Testing Data*



*Fig 4: HBSC Holdings Validation Data*



*Fig 5: Toyota Motor Corporation Testing Data*



*Fig 6: Intel Validation Data*