Stock Price Prediction using Machine Learning Models

Shivani Sheth [1]

Abstract

Machine learning models can be useful tools in predicting stock prices of a given company. Accurate predictions of a company's future stock price could yield significant profits for an entity. In this project, we use 17 year's of IBM stock price data to predict and evaluate its stock price in the last 3 years. The prices are predicted using machine learning models such as Linear Regression, Long Short-Term Memory Networks, and Autoregressive integrated moving average(ARIMA) models, and a comparison is made between the working and results of each model. The results show that the prices were most accurately predicted by the ARIMA model with a root mean squared error of 0.05, followed by LSTM with a root mean squared error of 1. Linear Regression performs the worst among all three models with a root mean squared error of 51.

1. Introduction

1.1. Stock Market and Stock Price Prediction

The stock market is a collection of markets and exchanges where the buying and selling of shares take place for publicly listed companies. Stock markets are important as they help a given company to raise capital and enable an individual to gain personal wealth. They also serve as an indicator of the economy state.

Stock price prediction refers to the act of predicting a future value of a company's stock price. Successful stock price predictions could help individuals make informed decisions, and increase their profit margin. However, accurately predicting the stock market prices is a significantly difficult task due to the dynamic and volatile nature of the stock market. While some believe in the *efficient market hypothesis* which suggests that the stock prices cannot be predicted accurately, there are other formal methods proposed in the domain of machine learning and deep learning that predict the stock price movement with high accuracy.

In this project, we discuss and implement three models of different domains, namely Linear Regression, Long Short-Term Memory (LSTM) Networks, and Autoregressive integrated moving average (ARIMA) models, that can be used to analyze the non-linearity in the stock prices and predict the future stock prices of a given company with high accuracies.

1.2. Linear Regression

Linear Regression is one of the most basic machine learning algorithm that can be implemented on any data to predict its future values. It is a linear model that assumes a linear relationship between the input and output variables. The model takes a form of

$$y_i = \beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip} + \varepsilon_i = \mathbf{x}_i^\mathsf{T} \boldsymbol{\beta} + \varepsilon_i$$

for all $i = 1, \ldots, n$, where x_i is the input value, y_i is the output value, ϵ_i is the error term, and β_i is the regression coefficients.

1.3. Long Short-Term Memory (LSTM) Networks

As compared to Linear Regression, Long Short-Term Memory Networks (or LSTMs for short) are widely used for sequence prediction tasks and are proven to be highly effective. LSTMs are a type of recurrent neural network that are capable of learning order decencies by storing the important past information. To store this information, a LSTM has three gates, namely the input, forget, and output gates. The input gate adds information to the cell state, the forget gate removes the information that is less important to the model, and the output gate renders the output of the model. Hence, the three gates basically regulate the flow of information that goes in and out of the cell.

1.4. Autoregressive integrated moving average (ARIMA)

The Autoregressive integrated moving average model or ARIMA model is a statistical analysis model that analyzes and forecasts/ predicts time series data based on the series'

[1]Department of Electrical Engineering and Computer Science, York University, Toronto, Canada. Correspondence to: Shivani Sheth <shivs29@yorku.ca>.

Project Report for EECS5327 (Winter 2021), Department of Electrical Engineering and Computer Science, York University. Copyright 2021 by the author(s).

past values. The **AR** in ARIMA which stands for Autoregression shows that the data regresses on its own lagged/ prior values. In other words, it indicates that the model uses the dependent relationship between the series' current data and some number of the past data values. The **I** in ARIMA which stands for Integrated which indicates that the model uses the differencing of raw data observations to make the time series stationary. Stationary data means that the data values are replace by the difference between its current value and its previous value at a given time step. The **MA** in ARIMA stands for Moving Average that indicates that the model uses the dependency between an observation and residual errors from a moving average model applied to lagged observations. It also means that the predictions of the model linearly depend on the past values.

Each of these components in the model (namely AR, I, and MA) are explicitly mentioned as parameters of the model. These are given by *ARIMA(p,d,q)*, where *p* is the number of lag observations included in the model, *d* is the number of times that the raw observations are differenced, and *q* is the size of the moving average window. The *p, d, q* parameters are also known as the lag order, degree of differencing, and the order of moving average respectively.

2. Literature Review

Stock Price Prediction is an active area of research by many researches in several domains including tech and finance. Current work in the field can be broadly divided into different clusters based on their approaches. The first cluster of models measure the relationship between the data values mainly using Regression ((Jaffe & R., 1989), (Fama & French, 1995)). The second cluster of models include statistical models that use the concept of time series and measure the relationship between the data using hypotheses((Ariyo et al., 2014)). The final cluster include other deep learning and natural language processing models such as LSTM, CNN, RNNs, and Reinforcement Learning ((Schöneburg, 1990), (KOHARA et al., 1997), (Lee, 2001), (Selvin et al., 2017), (Junqué de Fortuny et al., 2014), (Deng et al., 2011)).

In this project, we compare the different machine learning and deep learning approaches to predict the Closing Stock Prices of IBM and compare their results, working, and shortcomings.

3. Methodology

3.1. Dataset

The IBM dataset from *1999-11-01* to *2021-03-04* (approximately 20 years) is used to model the stock prices of the company using the above given architectures. The dataset consists of the *Open*, *High*, *Low*, *Close*, and *Volume* values of the company on each day in the above mentioned time period. However, since the stock market is closed on weekends and public holidays, the dataset excludes those dates. Thus, the total number of records sum up to 5369.

Data pre-processing:
First, we sort the dataframe values according to the *Date* index. Then, out of all the *Open*, *High*, *Low*, *Close*, values in the dataset, we only choose the Closing values of the stock price on each day. Thus, we take a single column of Closing values consisting of 5369 rows. Further, we divide the dataset into train and test sets so that we can train the models using the train set and evaluate the models using the test sets. A ratio of $8 : 2$ is chosen for train : test data and thus we get a total of $0.8 \times 5369 \approx 4296$ train records and a total of $0.2 \times 5369 \approx 1073$ test records. As per the dates given in the dataset, we notice that the train records are from *1999-11-01* to *2016-09-01* and the test data is from *2016-09-02* to *2021-03-04*. The dataset is initially extracted in a Pandas dataframe from a json file but both train and test values are converted into a Numpy array after pre-processing.

3.2. Architectural Details

Now, let us look at the architectural details of the models implemented in the project.

3.2.1. Linear Regression

For IBM stock price prediction, we take a Simple Linear Regression model that takes the input values as the past records of IBM's closing stock prices, and outputs the predicted values of IBM's future stock prices. We then compare these predicted values against the actual values of the data and compute the model error. Since Linear Regression is a basic machine learning model, it does not require any hyperparameters and computes the error, coefficients, and the intercept of the model based on the training data.

3.2.2. Long Short-Term Memory Networks

For IBM stock price prediction, we implement a LSTM model using the *Keras* library. The model architecture consists of four sequential LSTM layers of 50 units each, where the first LSTM layer takes an additional parameter of the training data (input) shape. Each LSTM layer is followed by a Dropout layer of value 0.2 which helps in regularization of the model. The dropout layer functions by probabilistically removing or 'dropping out' the input/ recurrent connections to a layer so that they do not impact the activations and weight updates while training the network. Thus, the dropout layer prevents overfitting in the model and enforces generalization. The final layer of a LSTM model is a Dense layer of unit 1 that outputs the predicted value computed by the model.

Through experiments on various combinations of LSTM and Dropout layers, this architectural setting of 4 LSTM layers followed by 4 Drouput layers proved to be optimal for the given IBM dataset. Further, we compile the model using the Adam optimizer and the Mean Squared Error loss function.

3.2.3. Autoregressive integrated moving average (ARIMA) models

Lastly, to implement an ARIMA model for IBM stock price prediction, we start by finding the degree of differencing or the d parameter. We run the Augmented Dickey Fuller (ADF) test to check if the data is stationary with a given d value. The null hypothesis of the ADF test states that the series is non-stationary. Hence, if the p-value of the test is below 0.05 (the significance level), then we reject the null hypothesis and conclude that the series or data is in fact stationary. We run this test for multiple values of d starting from zero, and select the minimum d value that has a p-value < 0.05, so that we do not over-difference the series.

Next, to choose the optimal values of the p and q parameters of the ARIMA model, we plot the Partial Autocorrelation function (PACF) and Autocorrelation function (ACF) graphs, as well as verify with the (Bayesian Information Criterion) BIC values.

The PACF plot measures the correlations for observations with lagged values, that is not accounted for, by prior lagged values. The partial autocorrelation of lag (k) of a series is given by the coefficient of that lag in the autoregression equation of y. For example, if the autoregression equation of y is given by:

$$y_t = \alpha_0 + \alpha_1 y_{t-1} + \alpha_2 y_{t-2} + \alpha_3 y_{t-3}$$

and y_{t-1} is lag 1 of y, then the partial autocorrelation of lag 3 is the coefficient α_3 of y_{t-3} in the above given equation.

Similarly, the ACF plot measures the correlation of the observations with their lagged values. The y-axis for this plot gives the correlation coefficient between -1 and 1 for negative and positive correlation, whereas the x-axis gives the lag. The plot tells us the number of MA terms (or the q value) required to remove any correlations in the stationarized data.

We also verify the p and q parameters obtained by the ACF and PACF plots throught the Bayesian Information Criterion (BIC) score. Under a given Bayesian setup, the BIC is an estimate of a function of the posterior probability of a model being true. Thus, a lower BIC score means that the given model is more likely to be the true model. We plot the BIC scores with different values of p and q parameters for a given model and verify that the values match the results as obtained from the ACF and PACF plots.

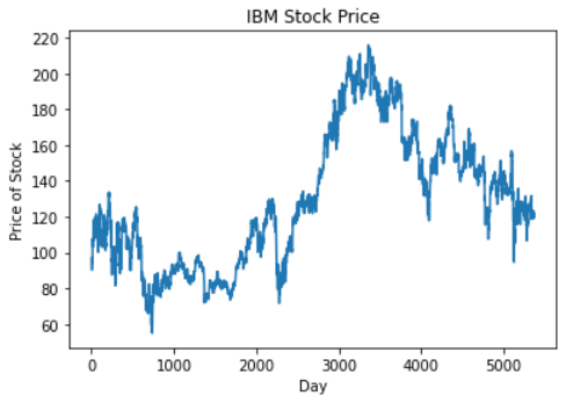


Figure 1. IBM Closing Stock Price

After obtaining the optimal values of *p, q, d*, we build the ARIMA model with the training set of IBM Closing value data and the optimal model parameters. Once, the model is fitted to the time series data, we forecast the IBM Closing stock prices for a given number of days (here, the size of the testing set). Since ARIMA follows a linear approach to forecast future values, a different approach is followed here as compared to the direct forecast method of the model. To include non-linearity of the data in this method, we forecast only one value at a time to predict the stock price of the next day based on the values fitted by the model. Next, we update the training set and add the actual value of the next day, fit the model again, and forecast the next day's value. This process runs in a loop for each data point until the model forecasts the last value in the test set. Note that here, the actual values from the test set are fit to the model, and not the predicted values of the model, due to which at each iteration, the model predict a highly accurate closing price for the next day. Finally, we store all the predicted values in an array and compare it with the actual values of the test set.

3.3. Evaluation Metric

The predicted results of each model are evaluated against the original values of the test set on the basis of Root Mean Squared Error (RMSE) metric. This metric is calculated for each model by *np.sqrt(np.mean(predicted - test)**2)*. In addition, both the predicted and test values are plotted against each other on a graph to visualize the difference between each data point.

4. Experiments and Discussion

In this section, we evaluate the three different models against the IBM dataset. As mentioned above, we take the Closing stock price values from the dataset. When we visualize all Closing prices of the dataset, we get a graph that is shown in 1.

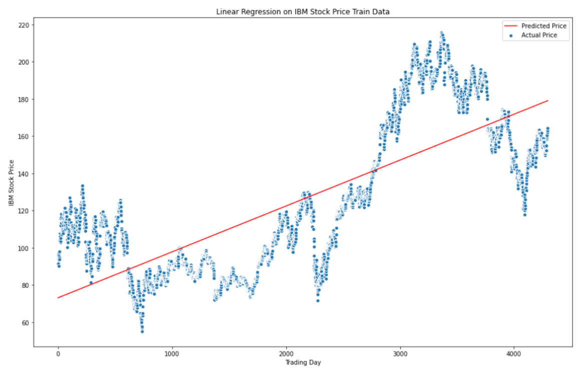


Figure 2. Linear Regression on train data

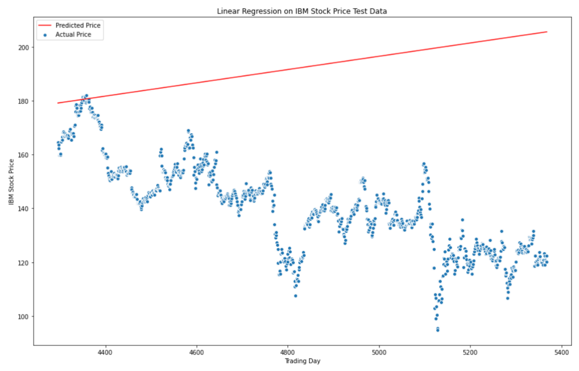


Figure 3. Linear Regression on test data

We then divide the Closing prices into train and test set consisting of 4296 and 1073 records respectively. Now, this data is pre-processed according to the models and each model is trained on the train test. Finally, the predictions of each model are evaluated against the test set. Below given are the results of each model architecture.

4.1. Linear Regression

For Linear Regression, we convert the train set into a 2D array for the *fit()* method of the model. After reshaping, we fit the model on the training data to obtain a slope of ≈ 0.024 and an intercept of ≈ 73.16. We then visualize Linear Regression on the training data which gives a plot as given by 2.

Here, as we see the model captures the linear relationship of the training data quite well since it has been trained on the same data. However, when we visualize the model evaluation on the test data we get a plot shown by 3.

As we see in this plot, the model does not capture the relationship between the test data very well. This is due to several factors, with one of them being the non-linear nature of the data, due to which a linear model would not be able to capture the relationship between the data very accurately.

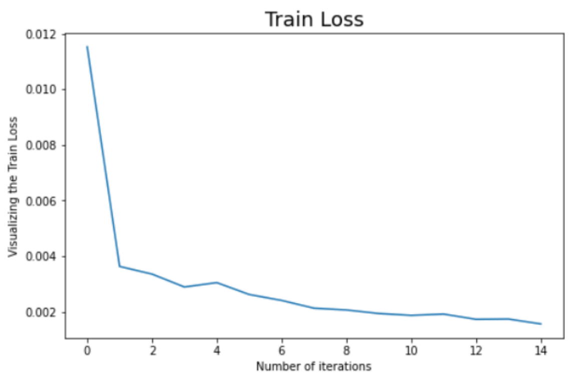


Figure 4. LSTM training error

Another reason could also be in the difference between the trend in the training set and the test set due to which the model predicts an increasing trend on the test set, while the original values of the test set are actually decreasing. Thus, the model does not predict the test data very well and we get a RMSE of 51.80.

4.2. Long Short-Term Memory Networks

For LSTM networks, we normalize both train and test data using *MinMaxScaler* before feeding it into the model. Each data point is normalized to a value between 0 and 1 is then converted into a Numpy array. We further convert the normalized train and test data into a time series data structure with 60 time steps and 1 output, after which the data can be trained on a LSTM network.

The LSTM model is then fit to the processed data with a training loop of 15 epochs and a batch size of 32. These hyperparameters were tuned and optimized manually. Several other combinations of the number of training epochs and batch sizes were also tried on the model architecture with the given dataset, but it was noticed that when we increased the training epochs, the model overfit the training data and had a higher test RMSE. On the contrary, when we decreased the number of training epochs, the model underfit the training data and still gave a higher RMSE. Keeping the batch size constant at 32, the optimal value of the number of training epochs was chosen to be 15, which gave the lowest RMSE on the test data. Similarly, experiments with the batch size, keeping the training epochs constant at 15 were also conducted. However, both decreasing and increasing the batch sizes from 32 gave a higher RMSE. Thus, the optimized values of epochs $= 15$ and batch size $= 32$ were chosen for the model training. The training loss for the model is visualized in 4, where we see a decrease in the train error which means that the model is learning.

The output values were predicted by the model using the *predict()* function, which were further passed through an

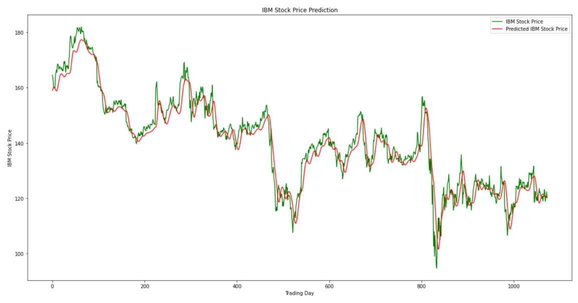


Figure 5. LSTM predictions vs test data

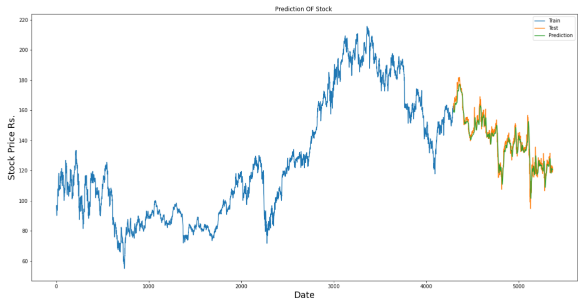


Figure 6. LSTM train, test, predicted data

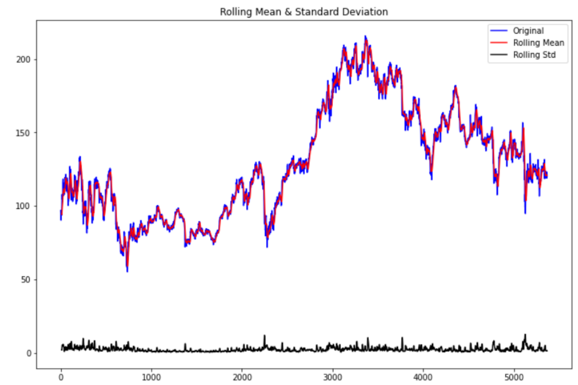
inverse_transform() function to get the unnormalized values of the predictions. These predictions were then compared with the original test data in two plots. In 5, we see that only the predictions and test data are plotted against each other, whereas in 6, we see that the predictions are all values, including the train, test, and predicted values are plotted against each other for comparison with the original values. [1]

As we see from both plots 5 and 6, the model predictions are quite accurate as compared to Linear Regression, with a slight margin error. The RMSE obtained by the model on the test data is calculated to be 1.06, which is a significant increase from the previous linear regression model. This proves that the time series model is able to analyze and predict the non-linearities of the dataset in a better manner.

4.3. Autoregressive integrated moving average (ARIMA) models

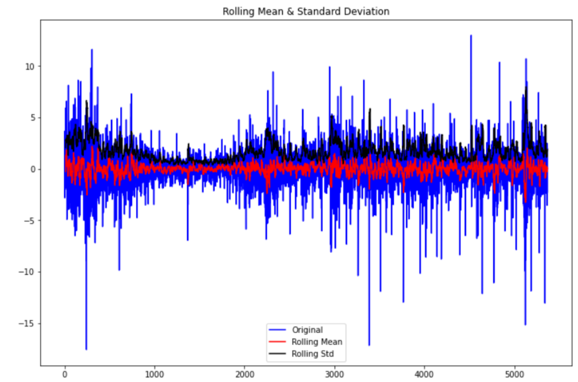
For ARIMA, we take the unnormalized values of the train and test data, similar to Linear Regression. We start by finding the optimal d parameter or the degree of differencing for the model. We perform the Augmented Dickey Fuller(ADF) test using the *adfuller()* function on the original values of the training set. The original values of the data, along with its rolling mean and standard deviation is visualized in 7.

[1]The implementation of all experiments can be found at: Stock Price Prediction by Shivani



```
Results of Dickey-Fuller Test:
p-value = 0.3451279249820689. The series is likely non-stationary.
```

Figure 7. ARIMA original train data



```
Results of Dickey-Fuller Test:
p-value = 5.5895753898431114e-30. The series is likely stationary.
```

Figure 8. ARIMA differenced train data

The p-value obtained from this data is seen to 0.345, which is more than the significance level of 0.05, and thus we can conclude that the data is non-stationary.

We further differentiate the values of the train set once using the *diff()* function and recheck the stationarity of the data using the ADF test. As shown in 8, this time, the p-value of the differenced data is much less than 0.05 which indicates that the null hypothesis is false and concludes that the data is now stationary. Since we choose the minimum value of d as the degree of differencing, we choose the optimal value as $d = 1$.

Now, to choose the optimal values of the p, we plot the PACF. As seen in 9, the PACF trails off after the first lag, and hence we choose the optimal p value as 1. To verify this optimal lag order and to choose the optimal order of moving average, we also plot the values for each parameter p and q respectively against their Bayesian Information Criterion (BIC) scores.

With 10 and 11, we verify that the optimal value of the lag order and order of moving averages is $p = 1$ and $q = 0$

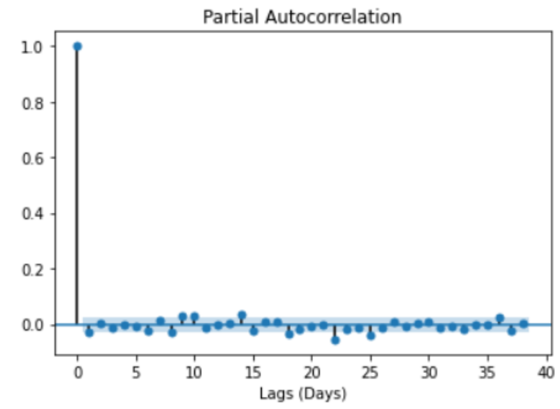


Figure 9. ARIMA PACF

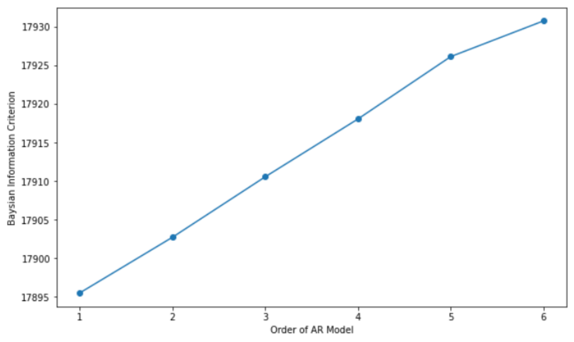


Figure 10. ARIMA lag order BIC score

respectively, since they correspond to the lowest BIC scores. Thus, we get the model parameters for ARIMA as $p = 1$, $d = 1$, and $q = 0$. Now, we train the ARIMA model on the undiffernced (original) values of the training set with the model parameters $(1, 1, 0)$ in the order of *(p,d,q)*. The model summary that we obtain is given in 12.

Now, to forecast the values up to the number of days as given in the test set, we follow a different methodology that predicts the IBM Closing price for the next day, given a training set. This method is run in a loop for n times, where n is the number of records in the test set. In each iteration, the predicted closing value is stored in an array and the train set is updated with the closing price value of that day. Hence, in the next iteration, the price is predicted for day 2 and the same procedure is followed till day n. The predicted values that we obtain from this method give very accurate results, as shown in 13. The per day prediction helps us acquire the non-linear predictions in the data as compared to the linear predictions obtained by applying the forecast method once for all test values.

In 13 and 14, we see that the predicted values are almost the same as the actual test values of the data set, which enforces the power of per day predictions, as modelled by ARIMA.

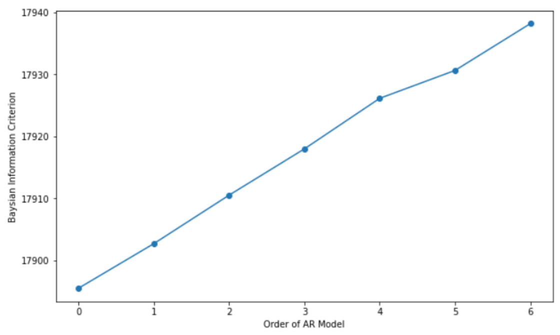
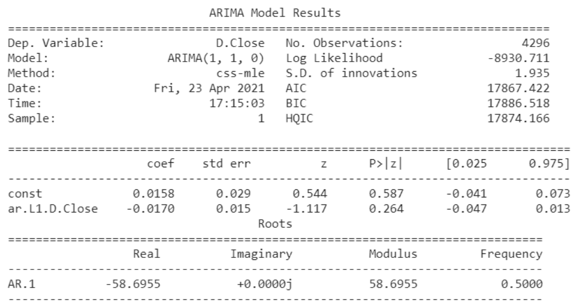


Figure 11. ARIMA order of moving average BIC score



```
                              ARIMA Model Results
==============================================================================
Dep. Variable:                D.Close   No. Observations:                 4296
Model:                 ARIMA(1, 1, 0)   Log Likelihood               -8930.711
Method:                       css-mle   S.D. of innovations              1.935
Date:                Fri, 23 Apr 2021   AIC                          17867.422
Time:                        17:15:03   BIC                          17886.518
Sample:                             1   HQIC                         17874.166

=================================================================================
                    coef    std err          z      P>|z|      [0.025      0.975]
---------------------------------------------------------------------------------
const             0.0158      0.029      0.544      0.587      -0.041       0.073
ar.L1.D.Close    -0.0170      0.015     -1.117      0.264      -0.047       0.013
                                    Roots
=============================================================================
                  Real          Imaginary           Modulus         Frequency
-----------------------------------------------------------------------------
AR.1          -58.6955           +0.0000j           58.6955            0.5000
-----------------------------------------------------------------------------
```

Figure 12. ARIMA model summary

The RMSE obtained by ARIMA under this setting is 0.05, which is significantly low as compared to both the other models. However, one thing to note here is that although the per-day predictions give higher accuracies, it also limits the predictions to only one day at a given time. Hence, it increases the training time taken by the ARIMA model as compared to other models.

Comparing the RMSE values for all models together in a tabular form, we see from 1 that ARIMA outperforms both the other models in terms of the test error by a significant margin. Thus, it can be concluded that with optimal values of *p, d, q* parameters of the ARIMA model, it can analyze the data very well and give highly accurate results.

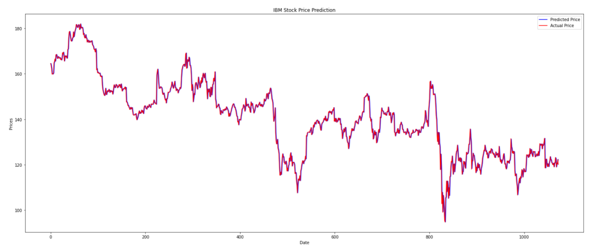


Figure 13. ARIMA predictions vs test data

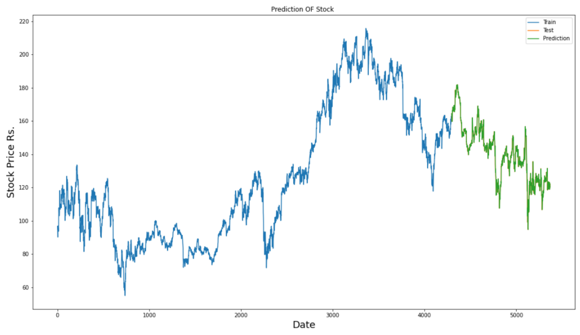


Figure 14. ARIMA train, test, predicted data

Table 1. Root Mean Square Errors (RMSEs) of all models on the test data.

MODEL	RMSE
LINEAR REGRESSION	51.80
LONG SHORT-TERM MEMORY	1.06
ARIMA	0.05

5. Conclusion

By predicting the Closing Stock values of IBM over a time period of about 20 years, we see through various experiments and results that some machine learning models are able to predict the future values of closing stock prices quite accurately. Through a comparison between three different models, we see that some models outperform others in terms of accuracy, and are able to capture the non-linearity in the data. Particularly thorough the series of experiments in this project, we see that the ARIMA model has the best performance with an error value of 0.05 RMSE. These accurate predictions can largely help an individual or a group of individuals gain high profits in the stock market and can help them make better decisions. We also see that other models, such as Linear Regression was not able to capture the non-linearity in the data very well and had a higher test error. Lastly, we see that LSTM also performs competitively with ARIMA and has a slight higher error rate. Thus, for stock price prediction, both LSTM and ARIMA models could be used to make accurate predictions based upon the length of predictions required by an individual.

References

Ariyo, A. A., Adewumi, A. O., and Ayo, C. K. Stock price prediction using the arima model. In *2014 UKSim-AMSS 16th International Conference on Computer Modelling and Simulation*, pp. 106–112, 2014. doi: 10.1109/UKSim.2014.67.

Deng, S., Mitsubuchi, T., Shioda, K., Shimada, T., and Sakurai, A. Combining technical analysis with sentiment analysis for stock price prediction. In *2011 IEEE Ninth International Conference on Dependable, Autonomic and Secure Computing*, pp. 800–807, 2011. doi: 10.1109/DASC.2011.138.

Fama, E. F. and French, K. R. Size and book-to-market factors in earnings and returns. In *Journal of Finance*, pp. 131 – 155, 1995.

Jaffe, J. Keim, D. B. K. and R., W. Earnings yields, market values and stock returns. In *Journal of Finance*, pp. 135 – 148, 1989.

Junqué de Fortuny, E., De Smedt, T., Martens, D., and Daelemans, W. Evaluating and understanding text-based stock price prediction models. *Information Processing & Management*, 50(2):426–441, 2014. ISSN 0306-4573. doi: https://doi.org/10.1016/j.ipm.2013.12.002.

KOHARA, K., ISHIKAWA, T., FUKUHARA, Y., and NAKAMURA, Y. Stock price prediction using prior knowledge and neural networks. *Intelligent Systems in Accounting, Finance and Management*, 6(1):11–22, 1997.

Lee, J. W. Stock price prediction using reinforcement learning. In *ISIE 2001. 2001 IEEE International Symposium on Industrial Electronics Proceedings (Cat. No.01TH8570)*, volume 1, pp. 690–695 vol.1, 2001. doi: 10.1109/ISIE.2001.931880.

Schöneburg, E. Stock price prediction using neural networks: A project report. *Neurocomputing*, 2(1):17–27, 1990. ISSN 0925-2312. doi: https://doi.org/10.1016/0925-2312(90)90013-H.

Selvin, S., Vinayakumar, R., Gopalakrishnan, E. A., Menon, V. K., and Soman, K. P. Stock price prediction using lstm, rnn and cnn-sliding window model. In *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pp. 1643–1647, 2017. doi: 10.1109/ICACCI.2017.8126078.