**Group ID: 13**

**Topic: Stock Market Trading using Machine Learning and Deep Learning**

Members:

1814054 - Jash Shah  
1814055 - Jill Shah  
1814056 - Vrutik Shah

GitHub Link: <https://github.com/vrutik2906/Stock-Market-Trading-using-Machine-Learning.git>

**Introduction**

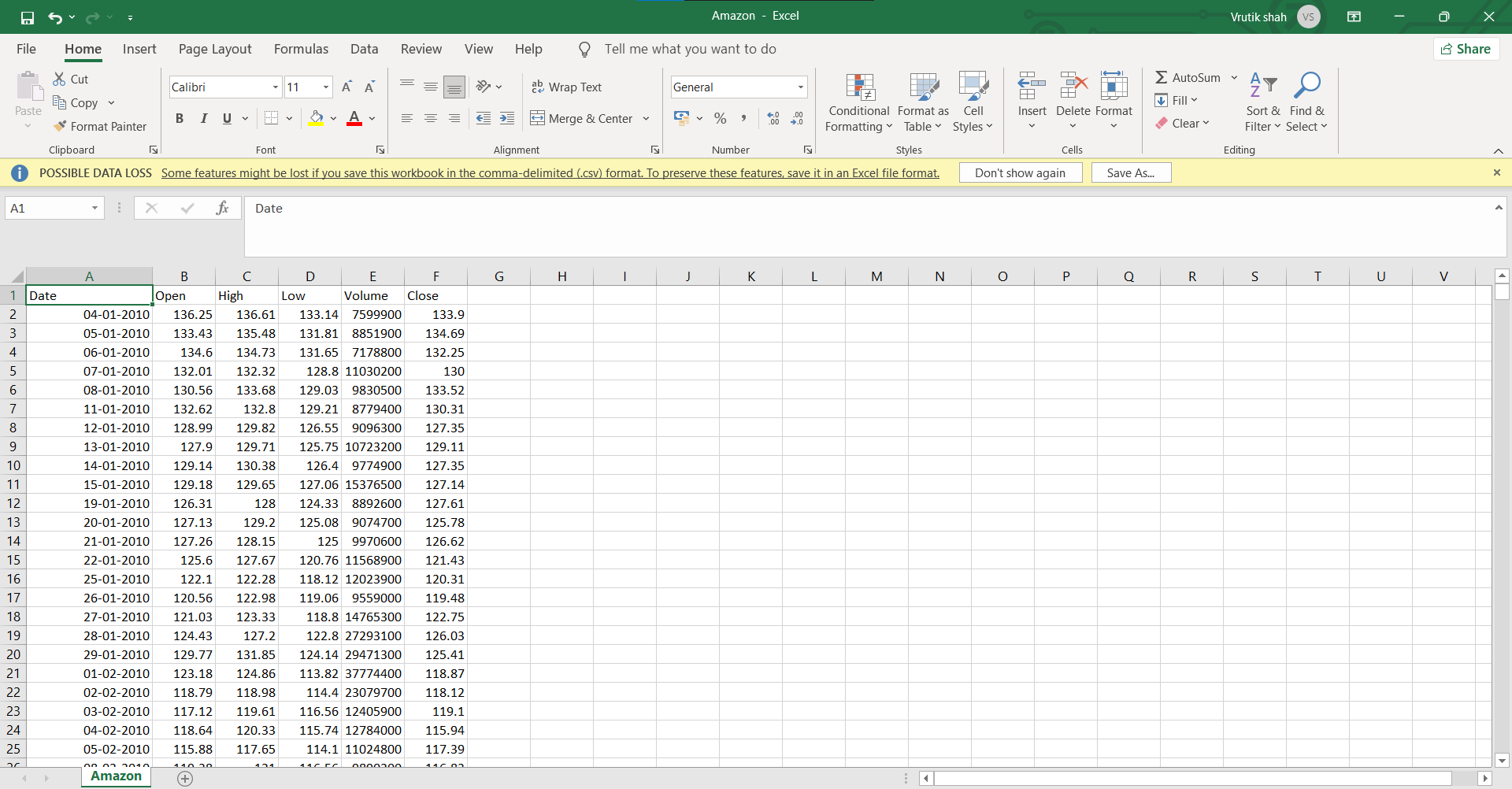
* Stock market forecasting is a technique for predicting the direction of the stock market and estimating the value of a stock or other financial asset in the future.
* 90 percent of the world’s data has been generated in the last few years as a result of the daily generation of 2.5 quintillion bytes of data. The financial market generates a significant amount of data. Recognizing a pattern and then devising an ideal method for making judgments is extremely tough for a trader.
* One of the most difficult things to accomplish is to predict how the stock market will perform. It is critical to develop a system that will function with optimum precision and take into account all key elements that may impact the outcome.

**Problem Definition**

* Every day, the stock market is mentioned in the news. Every time it achieves a new high or low, you hear about it.
* If an effective algorithm could be established to anticipate the short-term price of an individual stock, the rate of investment and business prospects in the stock market may increase.
* So, in this project we will do a comparative study of various machine learning algorithms like Linear Regression, Support Vector Machine, Decision Tree and LSTM Neural Network for efficient stock market prediction.

**Dataset:**

Amazon Stock Prices from date 04-01-2010 to 31-12-2020.



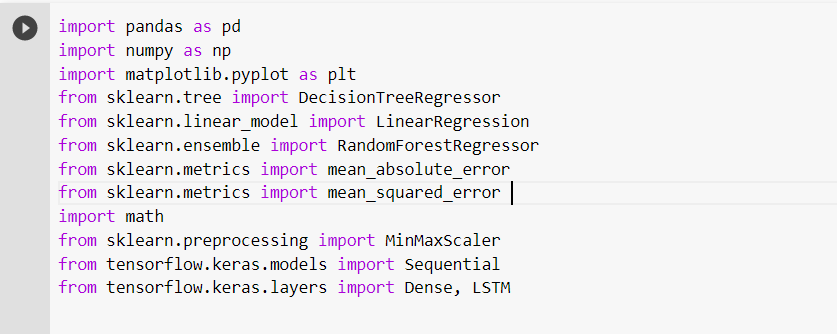
Rows: 2770

Columns: 6

## ****Implementation:****

**Importing the required libraries for our implementation:**

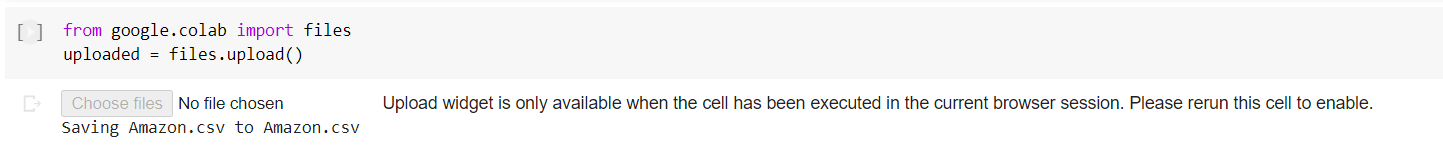
* Pandas - For Data Related Tasks
* Matplotlib – For Data Visualization
* Decision Tree Regressor from Sklearn – Decision Tree Classifier
* Linear Regression from Sklearn
* Random Forest Regressor from Sklearn
* Sklearn Metrics like Mean Absolute Error, Mean Squared Error, Root Mean Squared Error.
* Min Max Normalization from Sklearn
* TensorFlow models and layers – Sequential, LSTM, Dense



**Importing the CSV ‘Amazon’ using files module from Google Colab and uploading the dataset:**

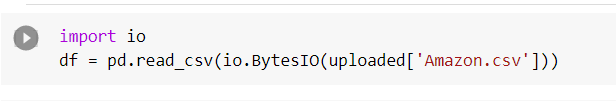
We chose a dataset from Amazon dated 04-01-2010 to 31-12-2020.  
This dataset attribute is open, high, low, close and volume but we selected close as a label data and the rest of to extract the features that will help to predict the result.

Source: <https://finance.yahoo.com/quote/AMZN/history?p=AMZN>

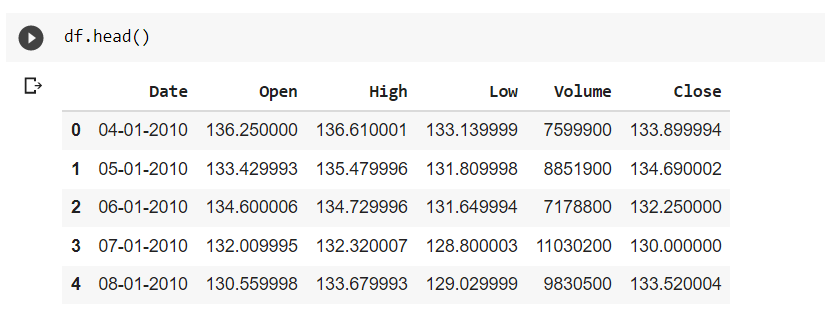


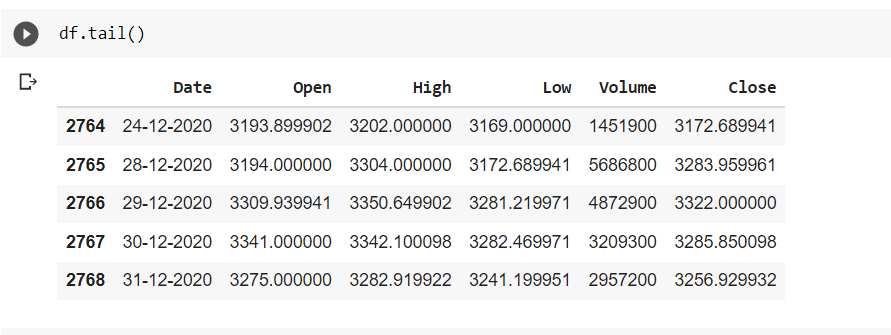
**Converting the above uploaded file into a dataset:**

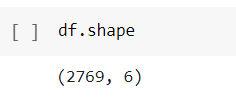
Using IO library which converts the values into a data frame.



**Dataset Overview**



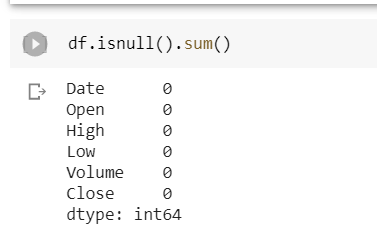




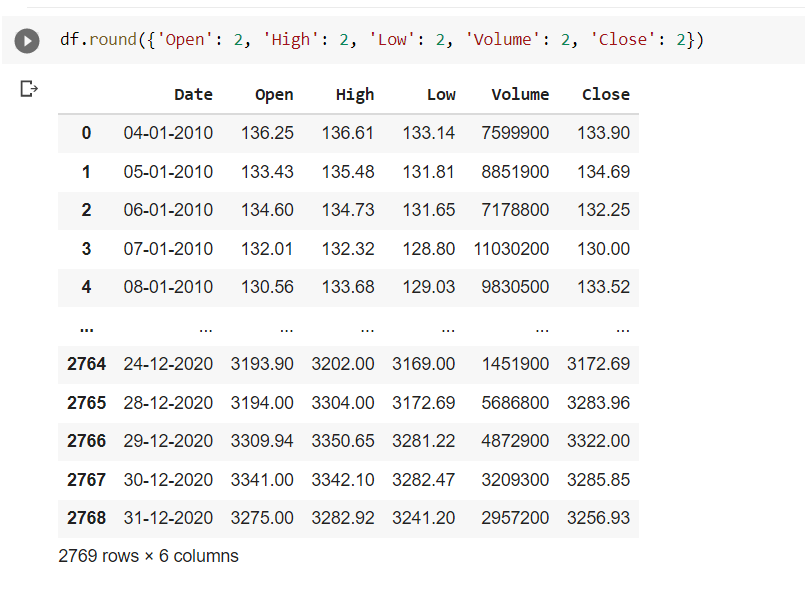
**Dataset Pre-Processing**

1. We will check for null values in the entire dataset

2. Round all the values up to 2 decimal places

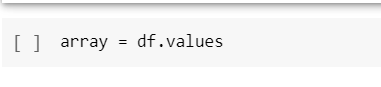


There are no null values present in the dataset. Therefore, no need of any technique for filling missing values.



Rounded all the values up to 2 decimal places.

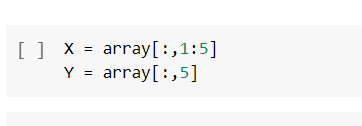
**Converting the dataframe into array.values for easy access to features and target variable and also for training the model.**

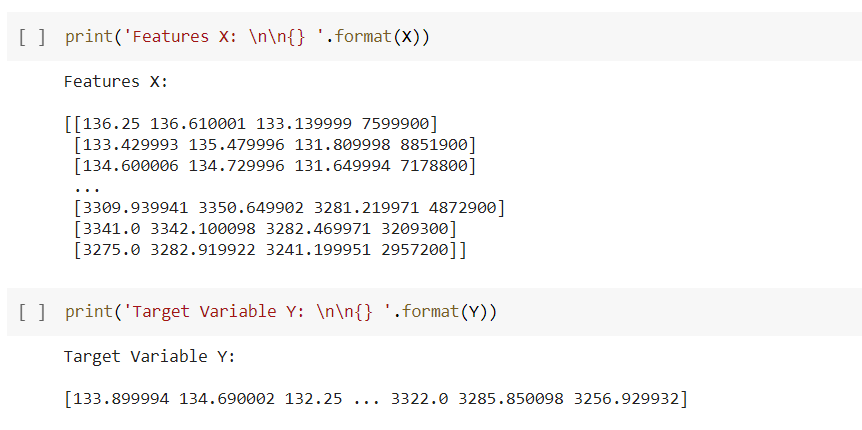
****

**Dividing the attributes into Features and Target Variable.**

## Features X: Attributes: {'Open', 'High', 'Low', 'Volume'}

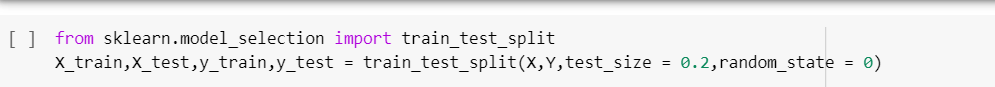
## Target Y: Attribute: {'Close'}

****

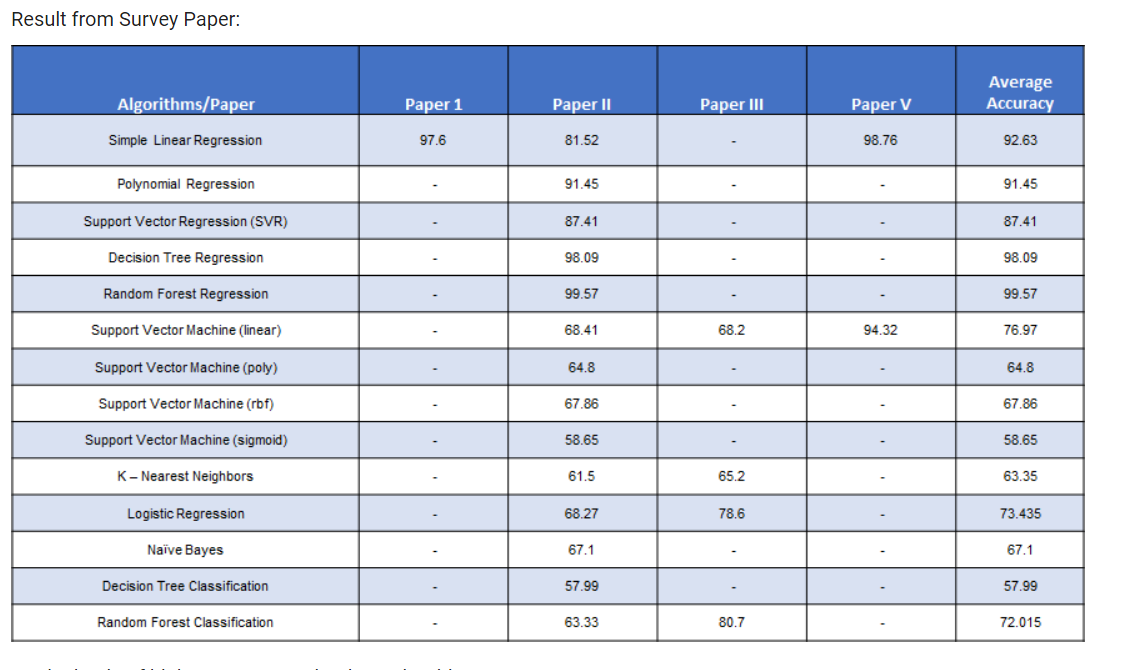
****

**Train-Test Split into 80-20 ratio**

The percentage of trains and tests would impact the accuracy of predicting the result.  
At this stage what ratio you want to choose for the train and test dataset it’s up to you but if you take more train dataset compare to test then accuracy would be better. The general ratio for train and test dataset is 80% and 20% respectively

****

**Selection of Algorithms**

****

**On the basis of highest accuracy, the three algorithms are:**

1. Simple Linear Regression2. Decision Tree Regression3. Random Forest Regression

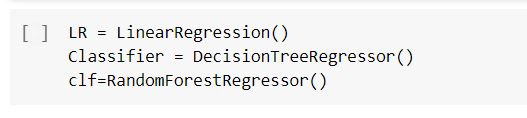
We will also implement one Deep Learning Algorithm after this called

**Long Short-Term Memory** for efficient Stock Market Prediction.

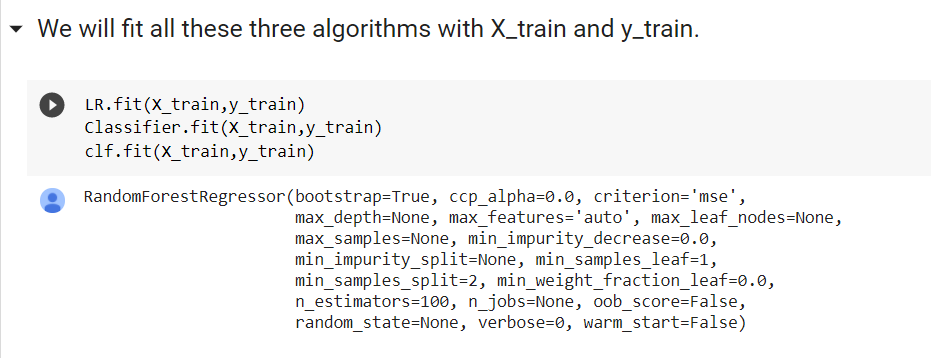
**Linear regression** is perhaps one of the most well-known and well understood algorithms in statistics and machine learning. Simple linear regression is useful for finding relationship between two continuous variables. One is predictor or independent variable and other is response or dependent variable. It looks for statistical relationship but not deterministic relationship. Relationship between two variables is said to be deterministic if one variable can be accurately expressed by the other.

**Decision Tree** is a Supervised learning technique that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome.

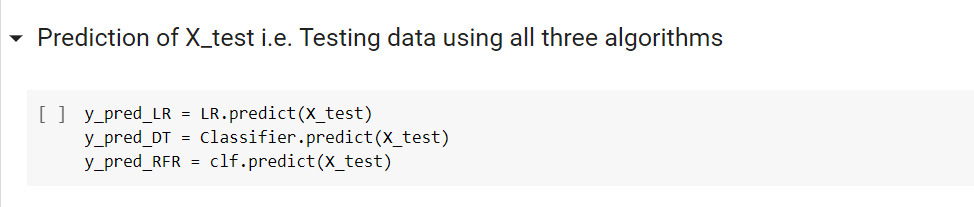
**Random Forest Regression** is a supervised learning algorithm that uses ensemble learning method for regression. Ensemble learning method is a technique that combines predictions from multiple machine learning algorithms to make a more accurate prediction than a single model. A Random Forest operates by constructing several decision trees during training time and outputting the mean of the classes as the prediction of all the trees.



After creating the instance of the algorithms, we call the fit function to fit the algorithm on the training dataset. Fit function is used by every estimator and it accepts the input of the sample data and also the arguments if the learning is supervised. It can also have the weights and other parameters.

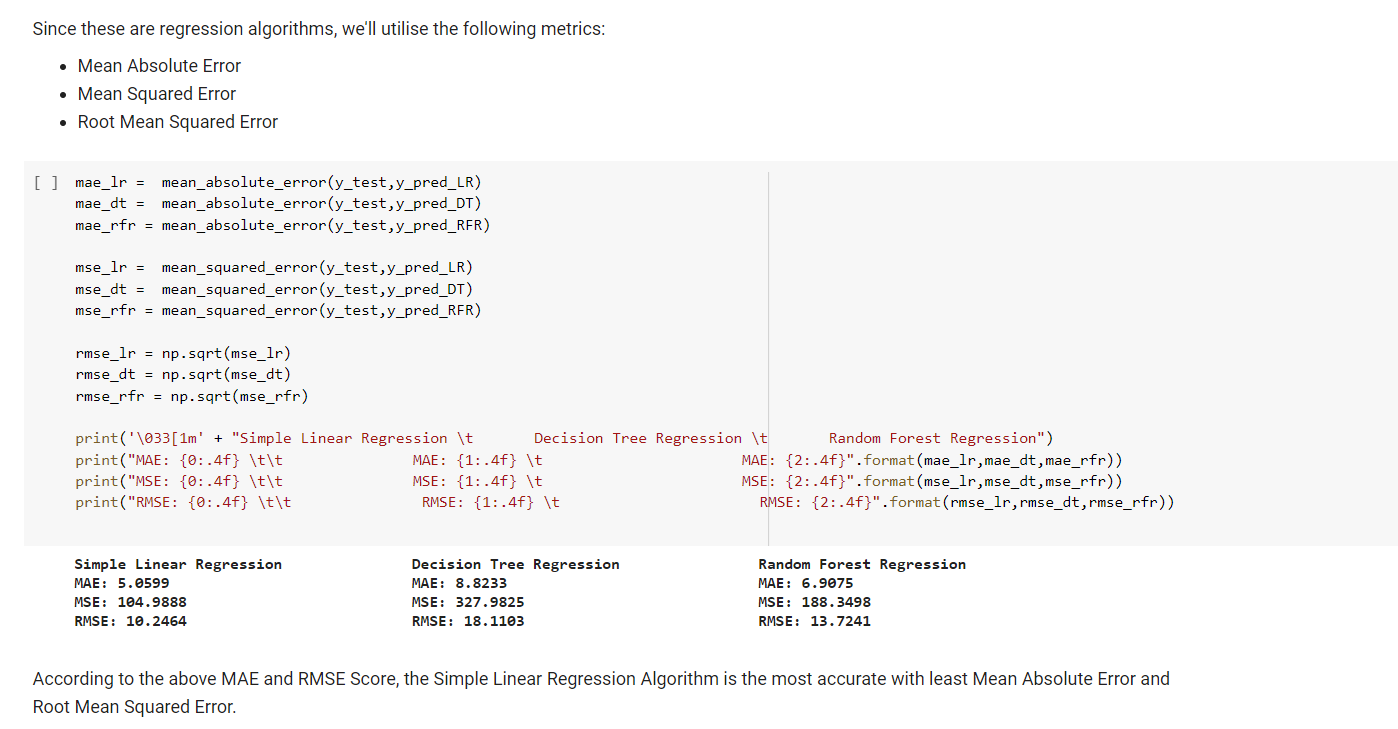


After training is done, we do the prediction of testing data by calling the test function. This function predicts the value or the class based on the regression or classification algorithm. Next, calculation of accuracy or error is done. We can calculate the accuracy for classification algorithm but for regression we can calculate the losses or errors.



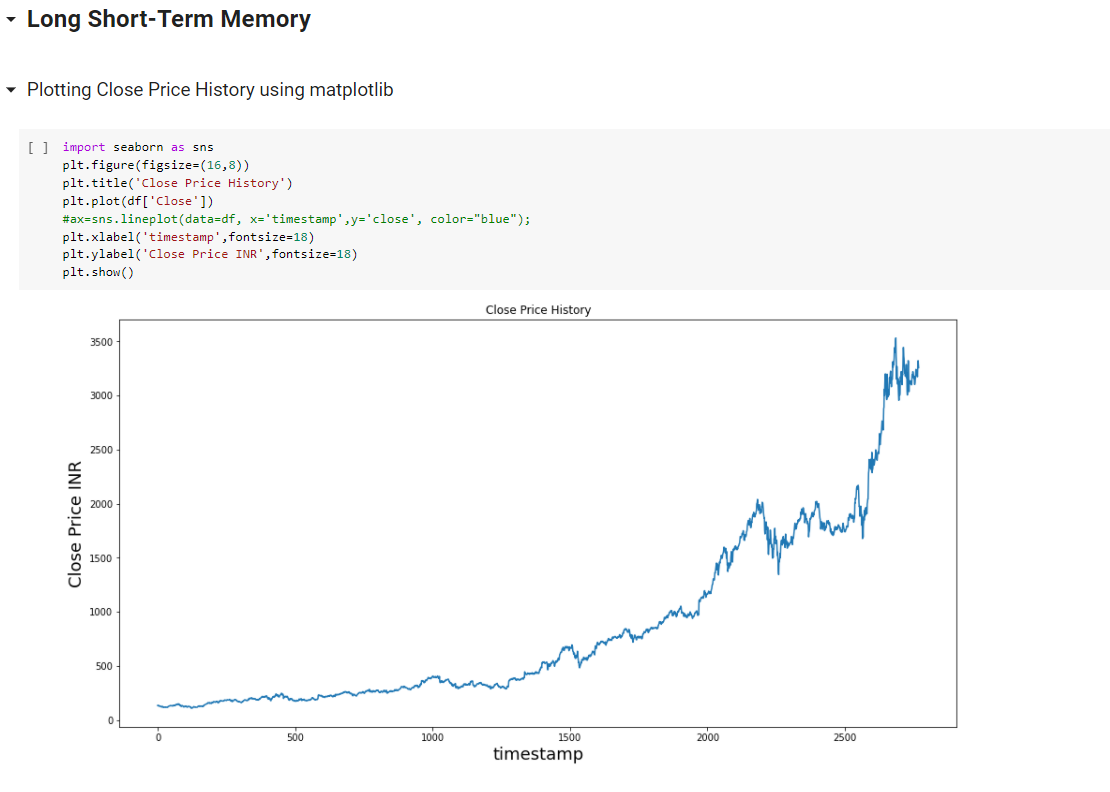
Three errors are used to calculate the best fit algorithm for each machine learning algorithm used. Mean squared error, root mean squared error and mean absolute error are the three errors used. These error functions are already defined in sklearn library. We import them and use them on our algorithms for each of three algorithms.

* **Mean squared error (MSE)** is the average or mean of the square of difference between actual and estimated values. Lesser the value of MSE, good is the fit of algorithm on data. The MSE is high value as the difference is squared. Hence, we take the square root and MSE which is then called RMSE i.e., root mean squared error. So, lower the RMSE, better is the algorithm.
* Here we see that the RMSE is lower for the Linear regression algorithm, followed by Random Forest regression and then last by Decision tree algorithm. One more error which is mean absolute error (MAE) can be used. This is the average of the magnitudes of the differences between actual and predicted values.

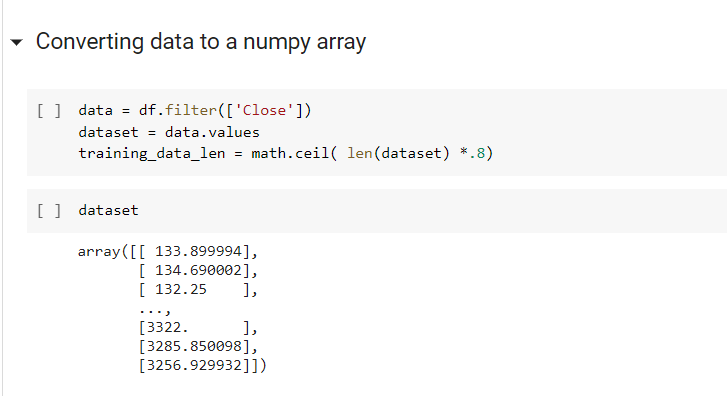


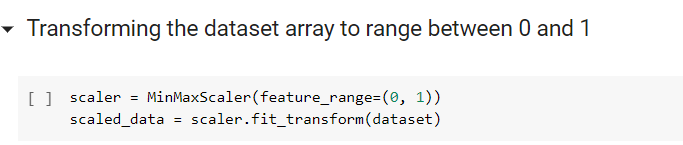
Next, we have also tried to predict the deep learning algorithm, LSTM (long short-term memory). This is a good algorithm for time series data (the price prediction over a period of time).

* **LSTM** is a recurrent neural network with feedback connections. We plot the close price history chart to see the trend of the price. Linear regression and other algorithms usually predict only prices with upward trend whereas the LSTM can predict with any or both upward and downward trend of prices.
* The plot is plotted using close price vs timestamp. Here we use only Close column to predict price further. We extract the column and convert it into NumPy array.

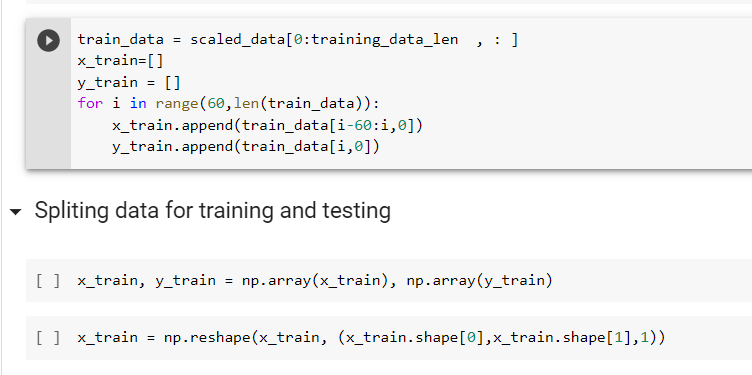


We see that the values are having very long ranges. So, we normalize the data to bring it to one particular range which is 0 to 1. MinMaxScaler is a function which can scale the data and normalize it.

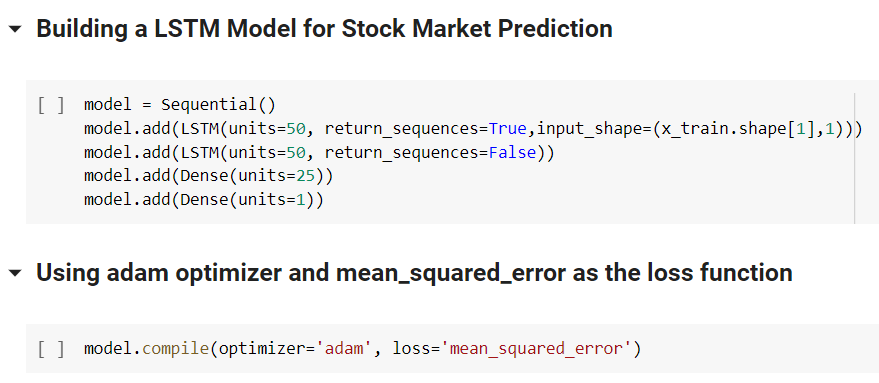




Then we split the data into 60 – 40 training and testing. Splitting is done manually using for loop here and then reshaped.



* Building a LSTM Model: The first layer is the sequential layer for the LSTM Model. After that The LSTM layer is added with the following arguments: 50 units is the dimensionality of the output space, return\_sequences=True is necessary for stacking LSTM layers so the consequent LSTM layer has a three-dimensional sequence input, and input\_shape is the shape of the training dataset. Then again, another layer is added with return\_sequences=False. The next layer added is a Dense layer with 25 units. Another name for dense layer is Fully-connected layer. It's actually the layer where each neuron is connected to all of the neurons from the next layer. It implements the operation output = X \* W + b where X is input to the layer, and W and b are weights and bias of the layer. The output layer is a dense layer with 1 unit.
* Adam optimizer: While compiling the model we have taken the optimizer to be as Adam. Adam optimizer involves a combination of two gradient descent methodologies: Momentum and Root Mean Squared Propagation.

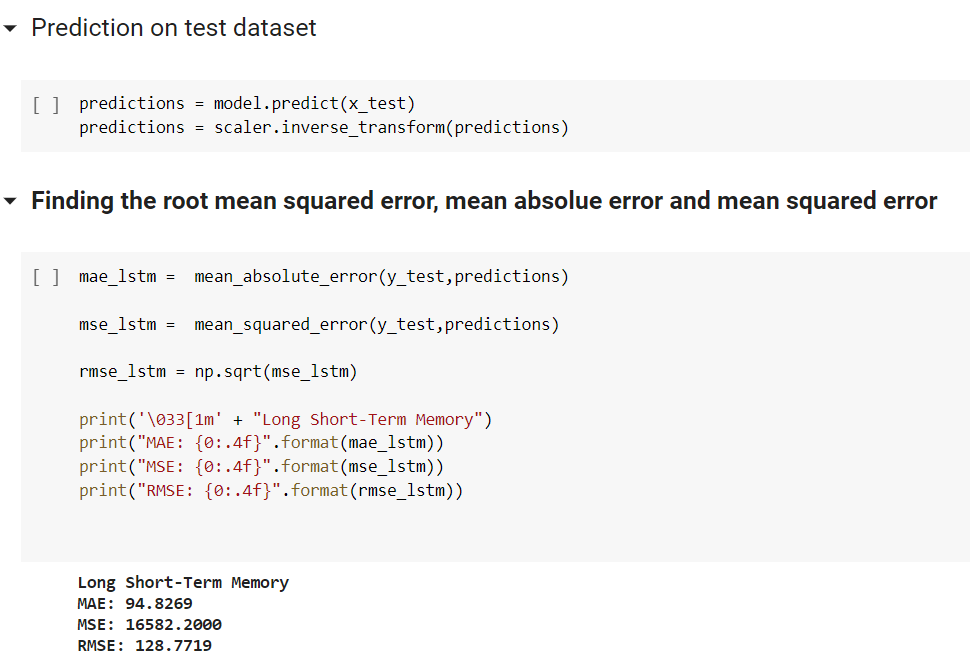


Now we fit the model for a batch size of 64 and for 5 epochs. We can see the loss after every epoch displayed over here. Then, we create the test datasets and convert them into a NumPy array.



Now, we make predictions on the test set. We calculate the performance of the model on various metrics such as:

1. In [statistics](https://en.wikipedia.org/wiki/Statistics), mean absolute error (MAE) is a measure of [errors](https://en.wikipedia.org/wiki/Error_(statistics)) between paired observations expressing the same phenomenon. Here, we have MAE as 94.8269.
2. The mean squared error (MSE) tells you how close a regression line is to a set of points. We have got a MSE of 16582.2.
3. Root Mean Square Error (RMSE) is the [standard deviation](https://www.statisticshowto.com/probability-and-statistics/standard-deviation/) of the [residuals](https://www.statisticshowto.com/residual/) ([prediction errors](https://www.statisticshowto.com/prediction-error-definition/)).  We have got a RMSE of 128.7719.

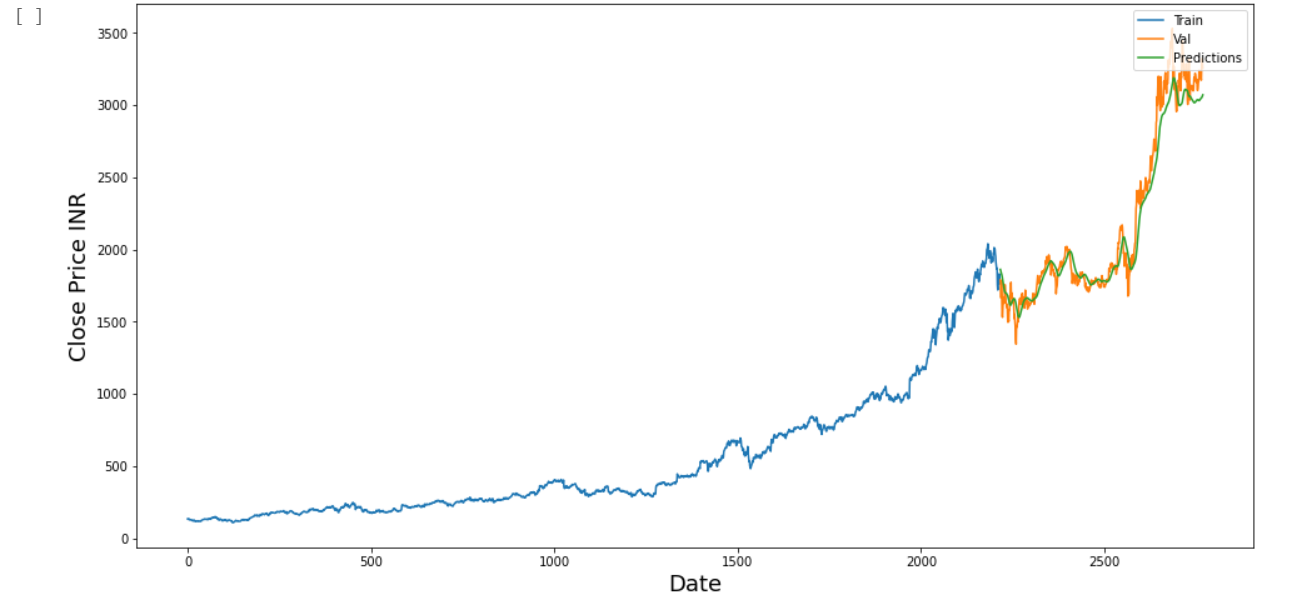


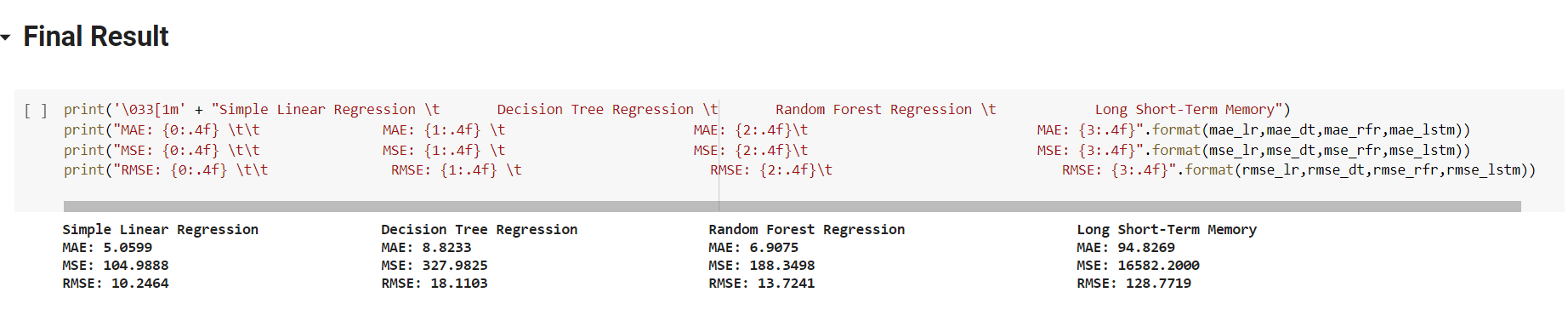
This is the plot Close Price INR versus Date. It has train, validation and prediction plot.

**Blue color - Plot of Training Data**

**Orange color - Plot of Validation Data**

**Green color - Prediction of Validation Data**





**Conclusion:**

We implemented the Simple Linear Regression algorithm, Decision Tree Regression algorithm, Random Forest Regression algorithm and Long Short-Term Memory algorithm on the Stock market data from 4 January 2010 to 31 December 2020 to predict its stock price.

As the results show, the lowest RMSE was found for Simple Linear Regression algorithm. This was the best from all the four algorithms. The next was Random Forest followed by Decision tree algorithm to predict the nearest close value of the stock price. The most RMSE was found for LSTM.

Ranking all the four algorithms from best to least to predict the close value of stock price are:

**I - Simple Linear Regression**

**II - Random Forest Regression**

**III - Decision Tree Regression**

**IV - Long Short-Term Memory**

**Outcomes:**

CO3 Comprehend radial-basis-function (RBF) networks and Kernel learning method Work

**Conclusion: (Conclusion to be based on the objectives and outcomes achieved)**

In this experiment, we created a Machine Learning model for Stock market prediction using Linear regression, Decision Tree and Random Forest Regression. And, we also explored LSTM model under Deep Learning.