Team Number: 35

*Mid-Project Evaluation* **Stock Market Prediction**

## Team Members:

Aanshul Sadaria (20161140)

Aniket Shrimal (20161136)

Ronak Aghera(20161002)

Deep Patel (20161010)

**Dataset:** Vanguard Total Stock Market ETF (VTI) dataset available from yahoo finance.

## Moving Average Method:

Moving Average is the technique to study the trend within a predefined history of a data, also used to create a smoothing effect over a series of data points having a noisy loss to damped the noise. They are used to gauge the direction of current trade.

### Why use Moving Average:

The fundamental assumption of technical analysis holds that past performance can inform future movements. Moving averages play a central role in the determination of past price trends. The direction and slope of moving average lines inform investors about the relationship between historical data values and present data values.

### Dataset Preprocessing:

The major steps taken during data preprocessing are as follows:

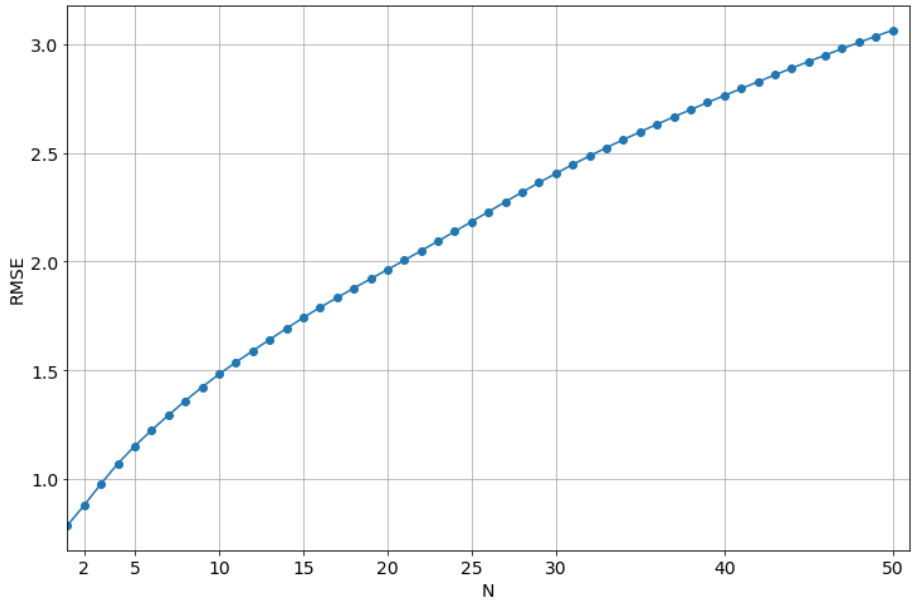
- Training - Validation - Test Split in the ratio of 60% - 20% - 20% respectively.
- During the prediction of validation/test data-points, the averages for last N dates are calculated on the fly.

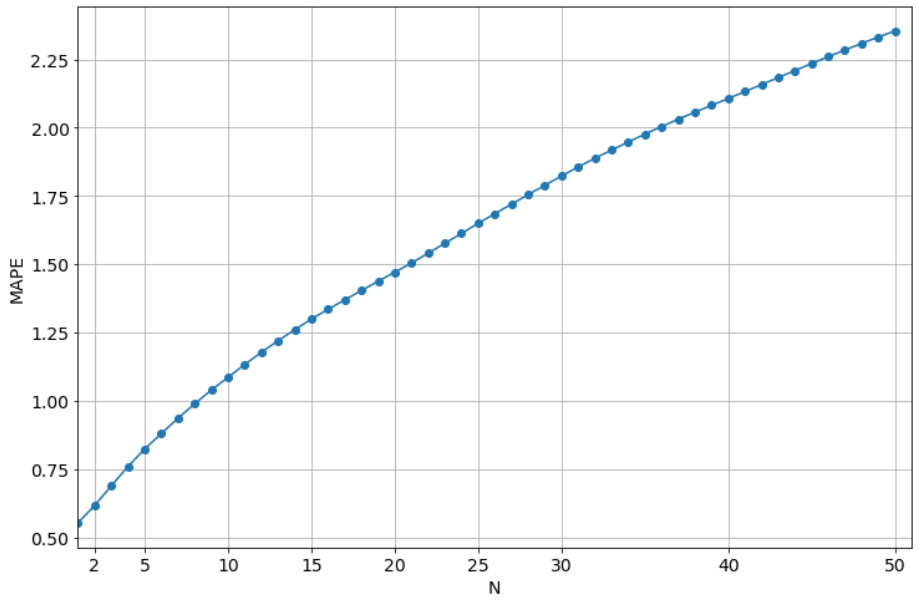
### Performance, Output, and Observations:

The model requires a single hyperparameter which the time-step (number of historical days whose data is used to compute the average).

The function for moving average loops over a rolling window of sizes for 1 to 50 to understand the behavior of performance measures against the hyperparameter.

Below are the graphs of different performance measures used against the size of the rolling window (time-steps).



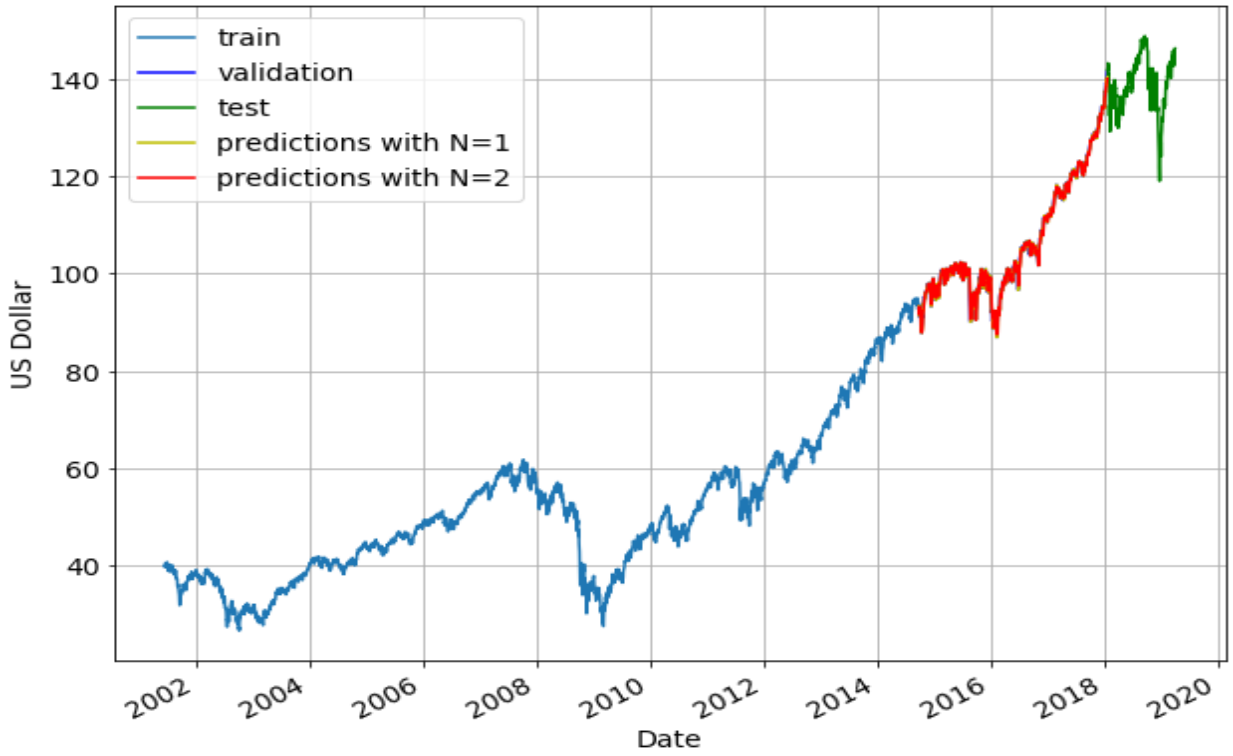




This clearly shows that N=1 is the best value for time-step. In other words, it means that the predicted value for day T is same as day T-1, and so on. Hence, there is no real mathematics or learning involved. Hence, we would choose N=2 as our meaningful optimal time-step value.

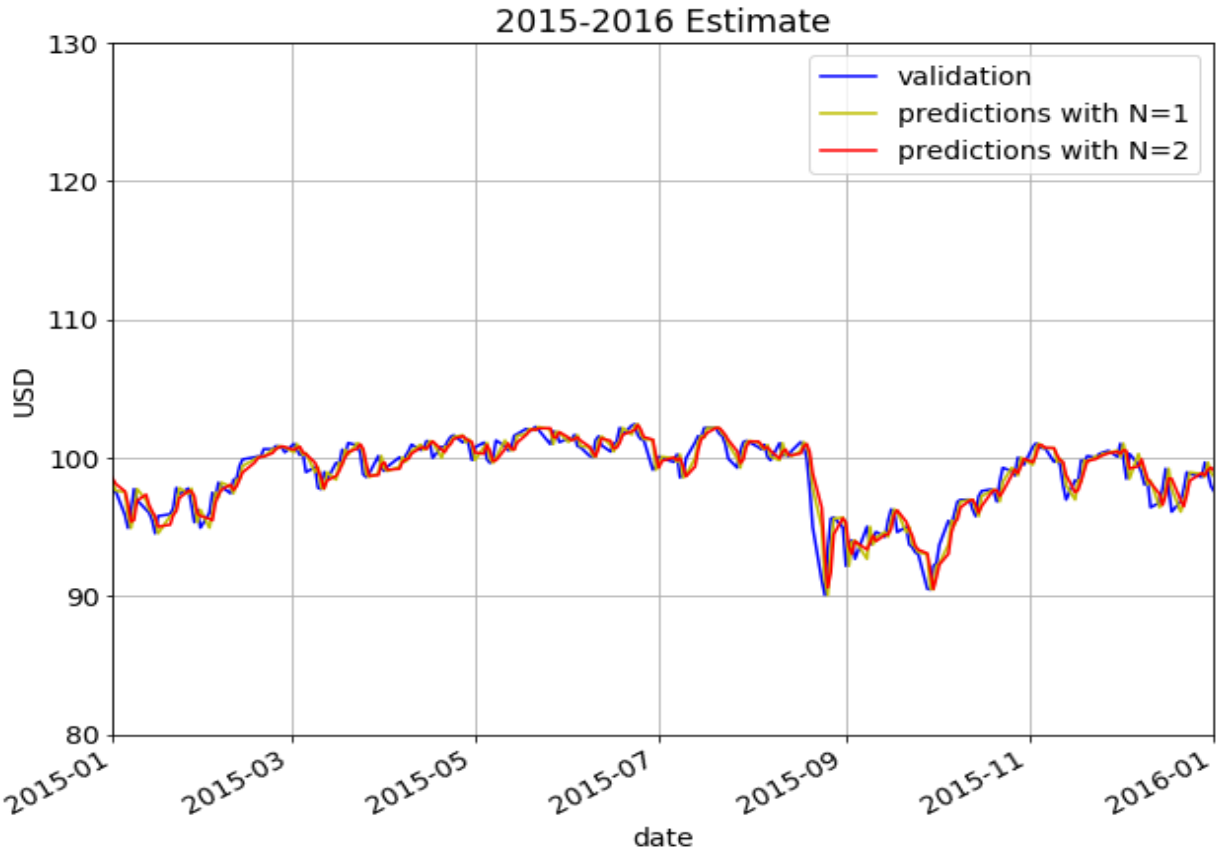
The performance of the model against different datasets is reflected in the table below.

| Data type | RMSE | MAPE |
|---|---|---|
| Validation Data | 0.752 | 0.675% |
| Training Data | 1.564 | 0.842% |

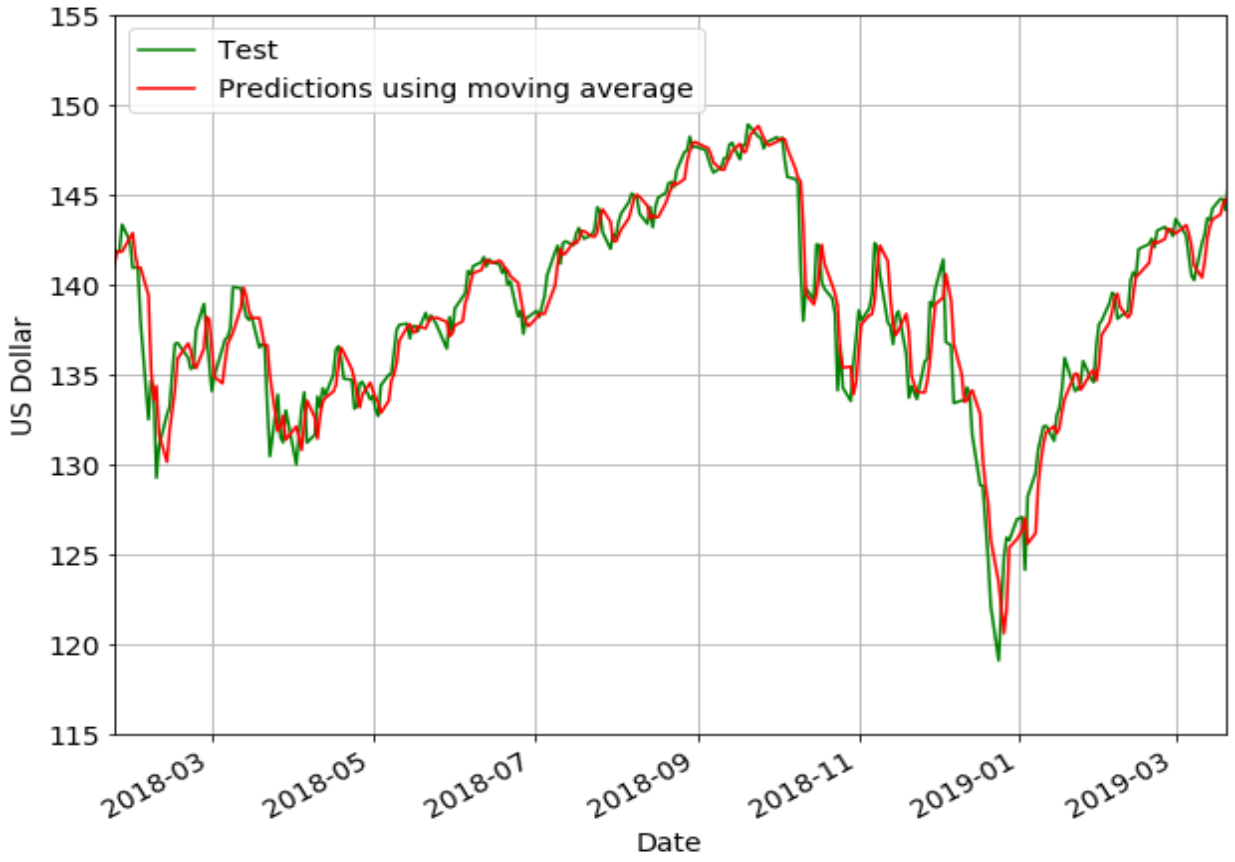
Now, shown below is the view of the complete dataset after the validation step which was meant for hyperparameter tuning.



Here, we are not able to comprehend the difference between actual and predicted values in the validation dataset. Hence, the plot for validation set (only) is shown below.




After choosing N=2 as our optimal hyperparameter, we apply the model on the test set to evaluate the performance. Below is the plot for the moving average technique on the test set.




## Linear Regression:

Python Scikit library's linear regression module has been used for this part. So any kind of regularization is not performed.

### Dataset Preprocessing:

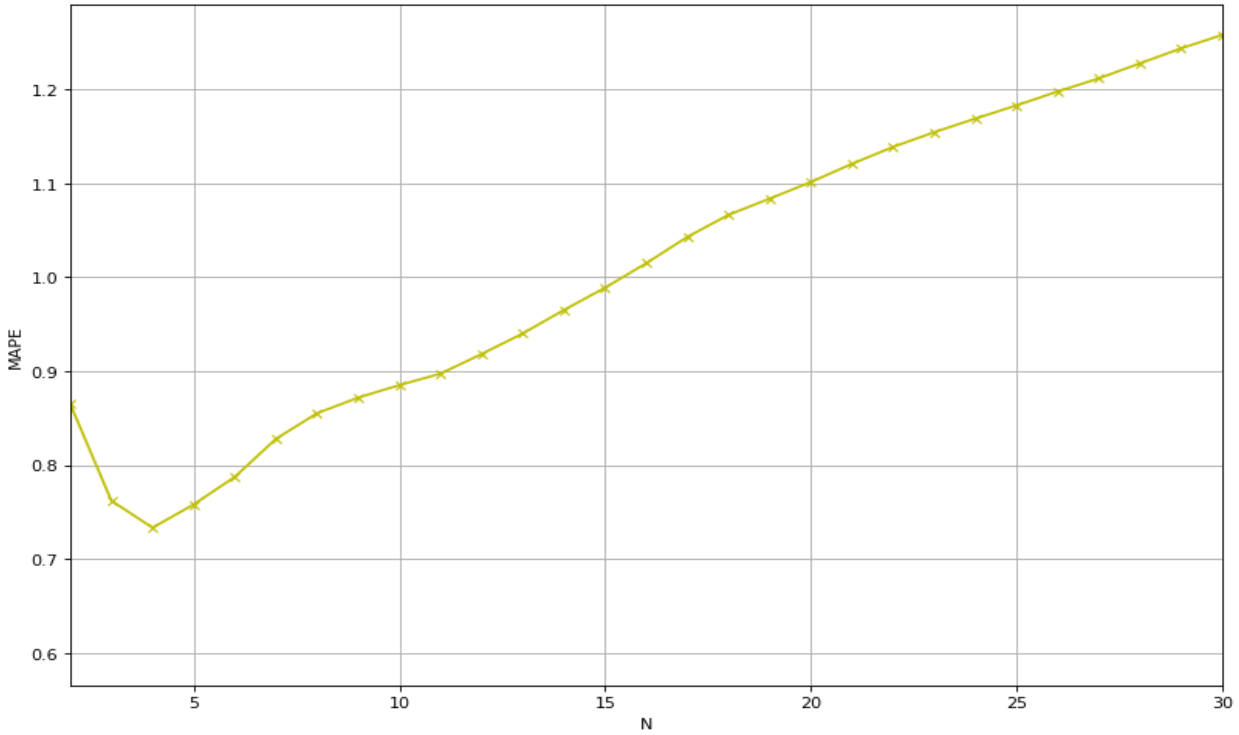
The major steps taken during data preprocessing are as follows:

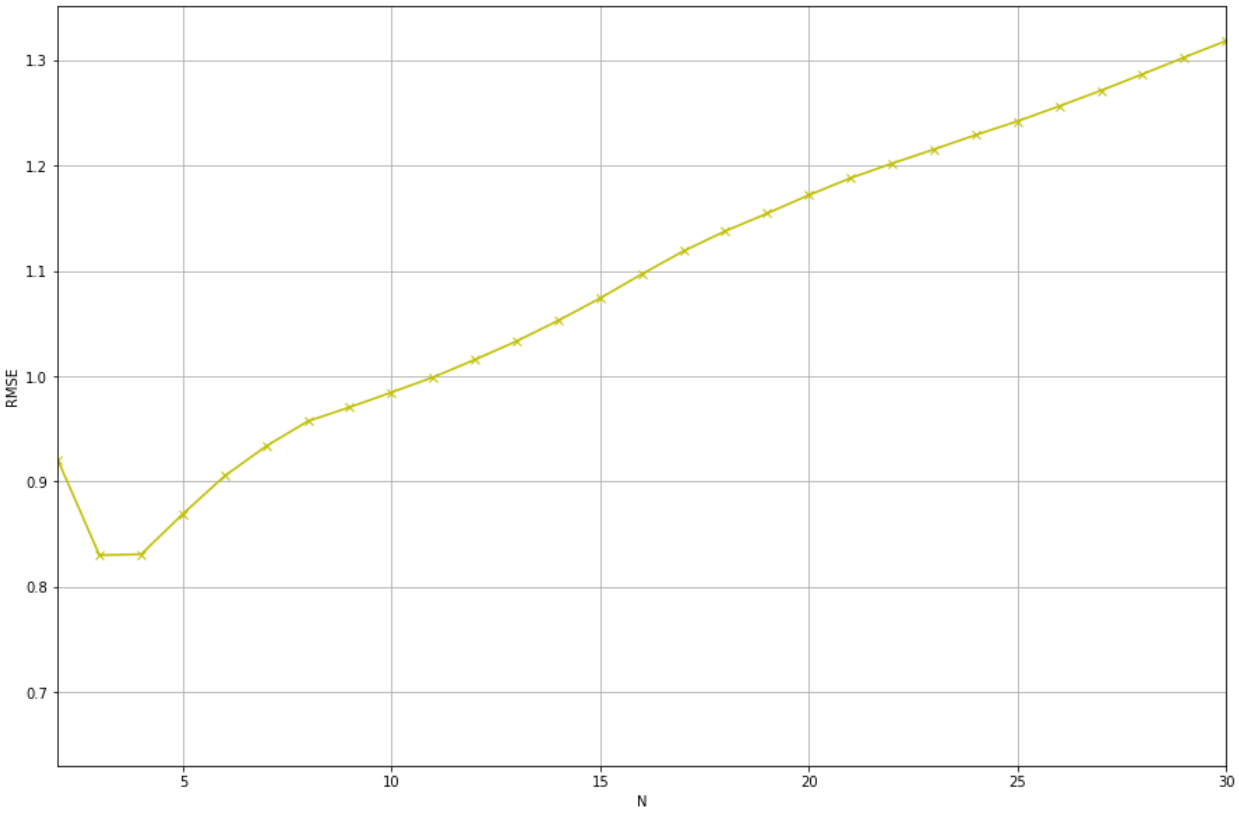
- Training - Validation - Test Split in the ratio of 60% - 20% - 20% respectively.
- During the pre-processing of validation/test data-points, the stock value for the next day is predicted using stock values from the previous n = (2 to 30) days and appended in the data tuples.

### Finding the best possible N value:

Using the values predicted for each n = (2 to 30) days, RMSE and MAPE for all the possible values of n are calculated, the value of n which gives the minimum error opts as the Optimum_N for further calculations.

Below are the graphs for variations of RMSE and MAPE with the values of n.


MAPE
N
1.2
1.1
1.0
0.9
0.8
0.7
0.6
5
10
15
20
25
30

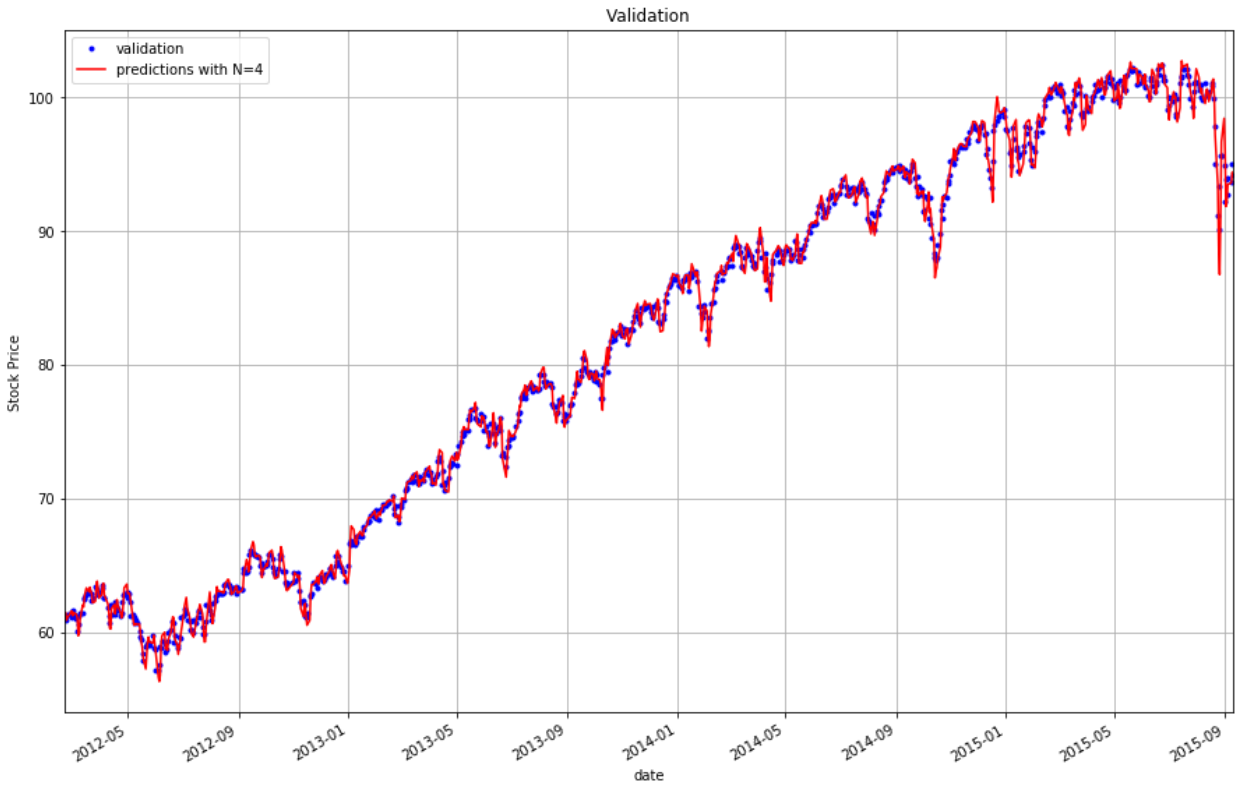

RMSE
N
1.3
1.2
1.1
1.0
0.9
0.8
0.7
5
10
15
20
25
30

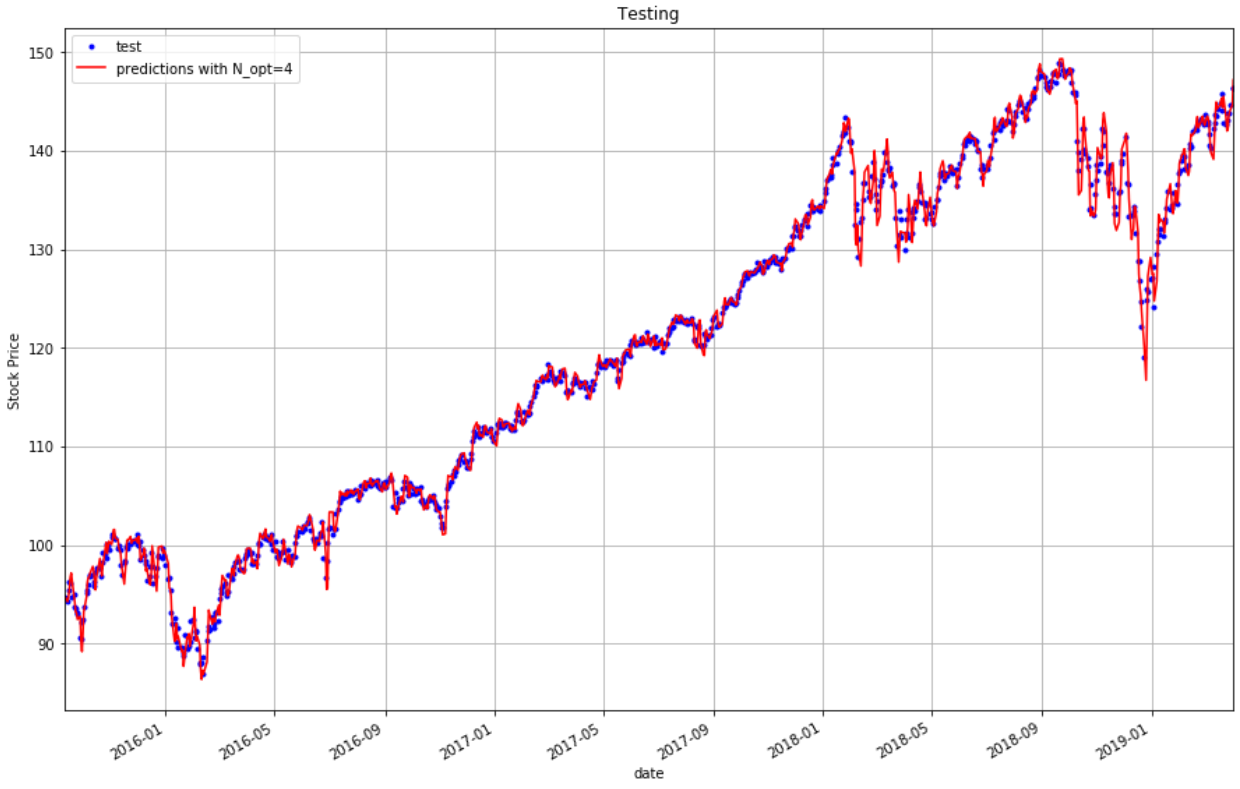
### Performance, Observation, and Graphs:

From the Above step, Optimum_N = 4 is obtained. Using this value of N, linear regressor is trained on the training dataset and prediction for validation/testing dataset is carried out.

Results Observed:

| Data type | RMSE | MAPE |
|---|---|---|
| Validation Data | 0.831 | 0.734% |
| Testing Data | 1.215 | 0.713% |




## XGBOOST Method:

XGBoost stands for “Extreme Gradient Boosting”. XGBoost is an implementation of gradient boosted decision trees designed for speed and performance.

The implementation of the model supports the features of the scikit-learn and R implementations, with new additions like regularization. Three main forms of gradient boosting are supported:

- **Gradient Boosting** algorithm also called gradient boosting machine including the learning rate.
- **Stochastic Gradient Boosting** with sub-sampling at the row, column and column per split levels.
- **Regularized Gradient Boosting** with both L1 and L2 regularization.

### Why use XGBOOST:

XGBoost dominates structured or tabular datasets on classification and regression predictive modeling problems. It is the go-to algorithm for competition winners on the Kaggle competitive data science platform.

XGBOOST is an open source library available for all.

### Dataset Preprocessing:

The major steps taken during data preprocessing are as follows:

- Training - Validation - Test Split in the ratio of 60% - 20% - 20% respectively

- Feature Extraction: The features used for the training are:
    - The volume of stock.
    - Difference between high and low
    - Difference between opening and closing price.
    - The closing price of every day.

      All these above features are extracted not only for the current day, but also for a slack/timestamp of N days.
- Scaling: All the feature values and target value are scaled using StandardScaler having mean = 0 and std-dev = 1.

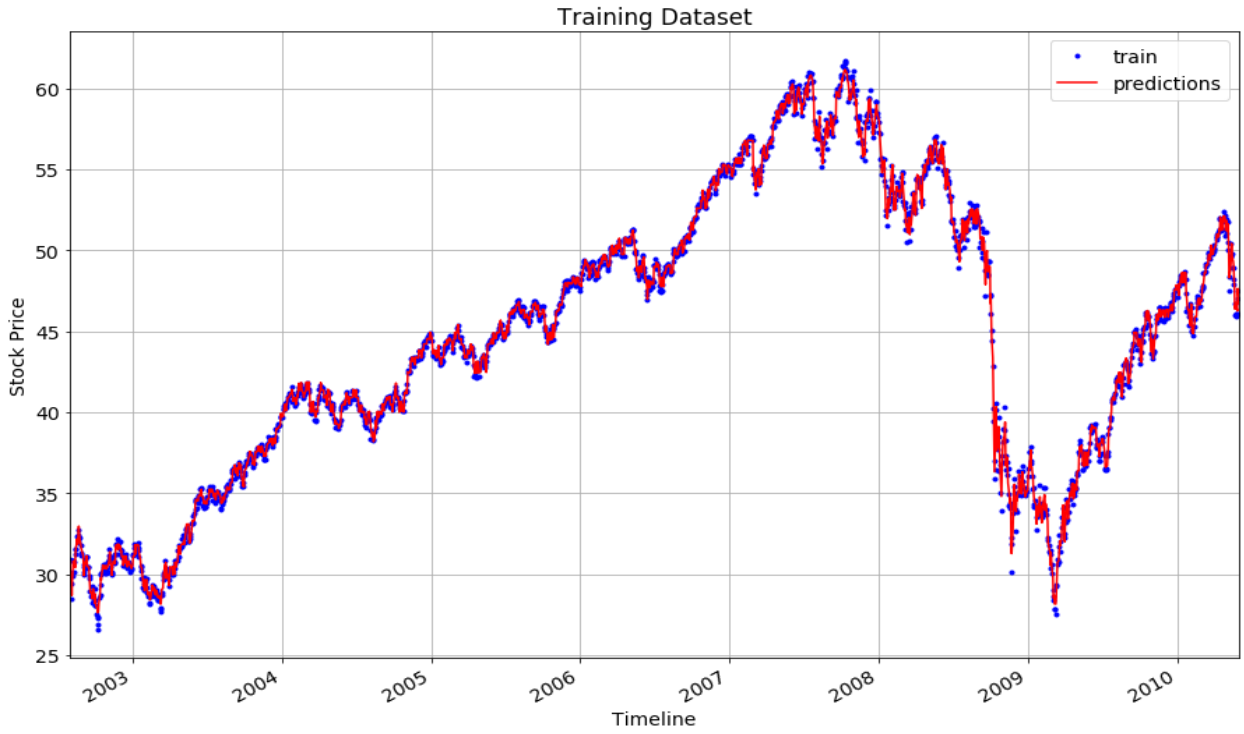
### Performance, Output, and Observations:

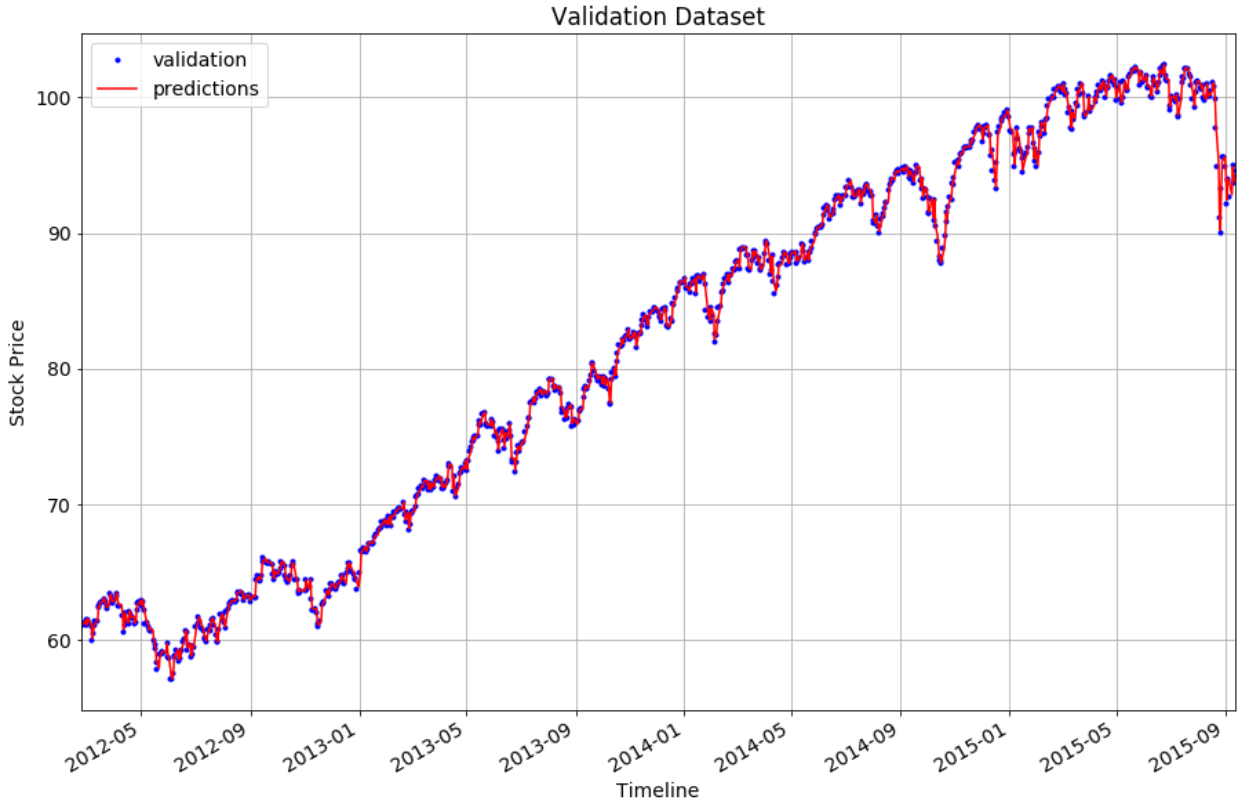
The model requires hyperparameters as max_depth_tree, learning_rate, n_estimators, etc.

For the first instance performing training on the default values of the hyper-parameters, the following performance was observed:

| Data type | RMSE | MAPE |
|---|---|---|
| Training Data | 0.463 | 0.779% |
| Validation Data | 0.668 | 0.603% |
| Testing Data | 1.032 | 0.596% |

Images for the same:


Training Dataset
train
predictions
Stock Price
Timeline
60
55
50
45
40
35
30
25
2003
2004
2005
2006
2007
2008
2009
2010


Validation Dataset
validation
predictions
Stock Price
Timeline
100
90
80
70
60
2012-05
2012-09
2013-01
2013-05
2013-09
2014-01
2014-05
2014-09
2015-01
2015-05
2015-09

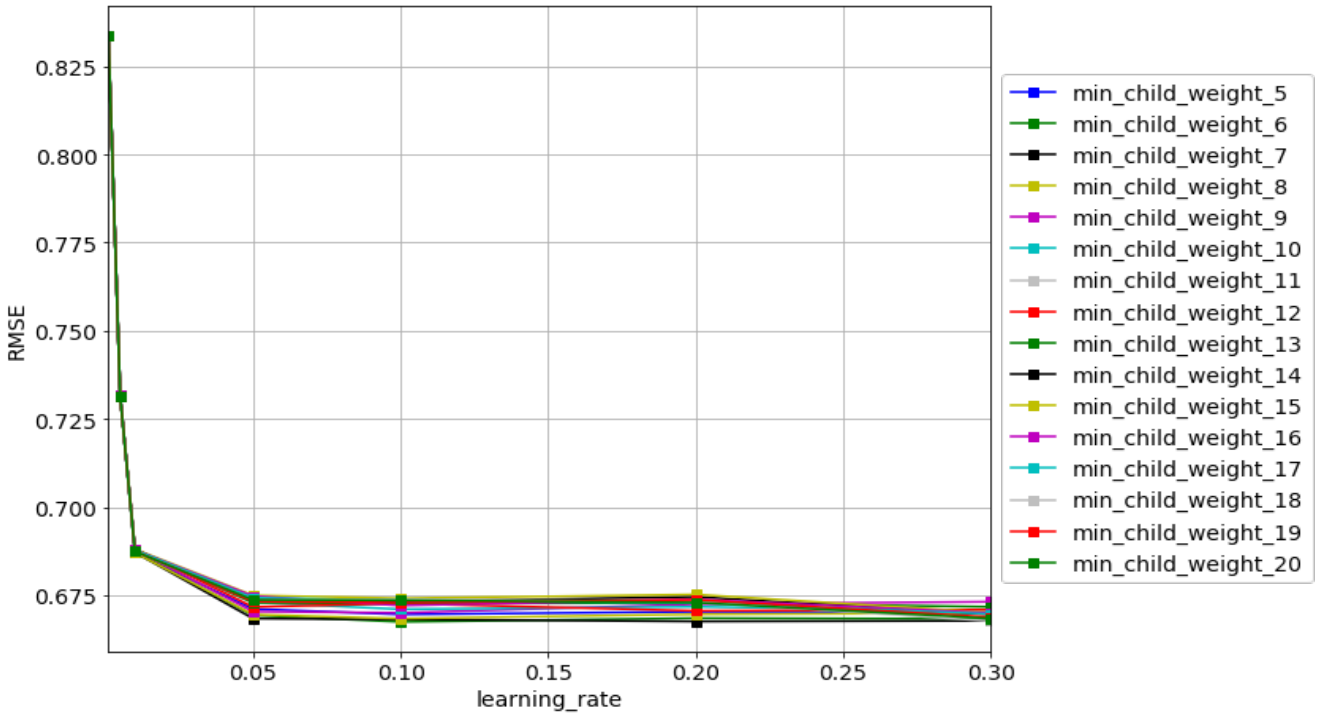
### HyperParameter Tuning:

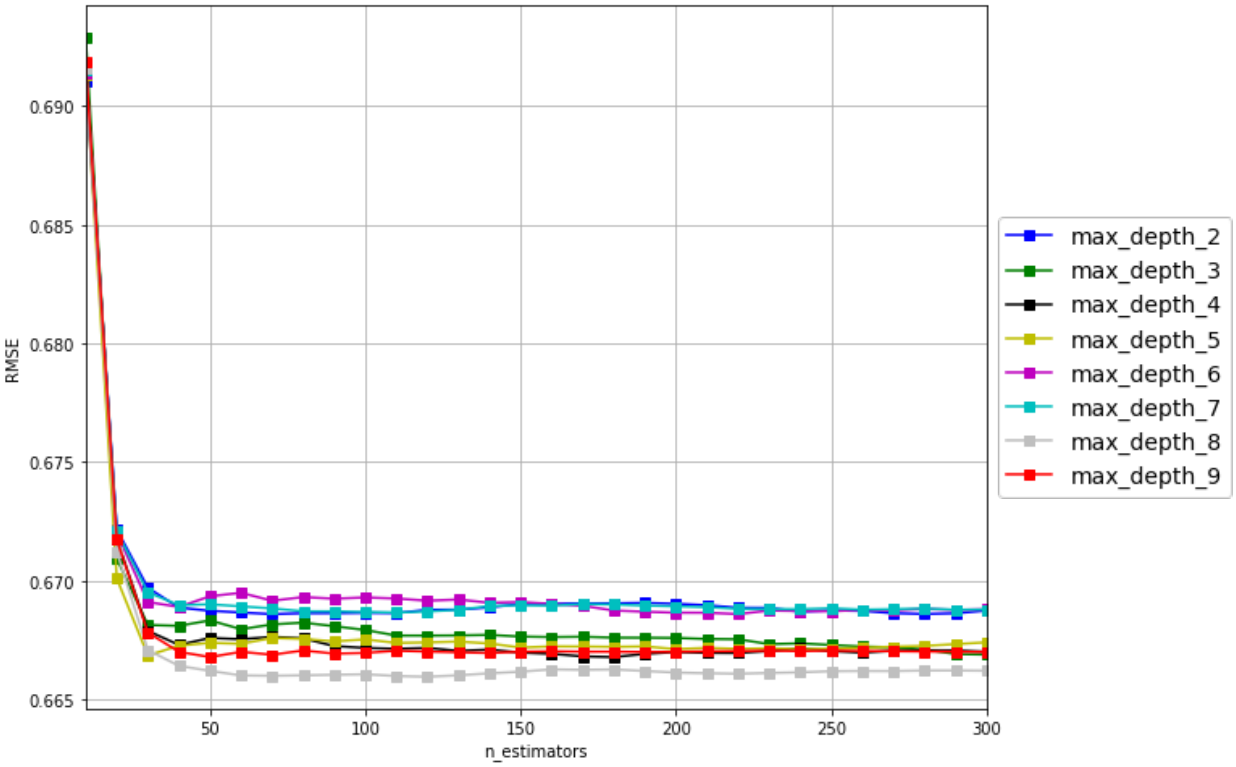
For Hyperparameter Tuning, the model is re-trained for all possible values of the hyperparameter. After that, the value of hyperparameter which gives the best performance is selected.

After performing tuning, the best values for each hyperparameter is as follows:

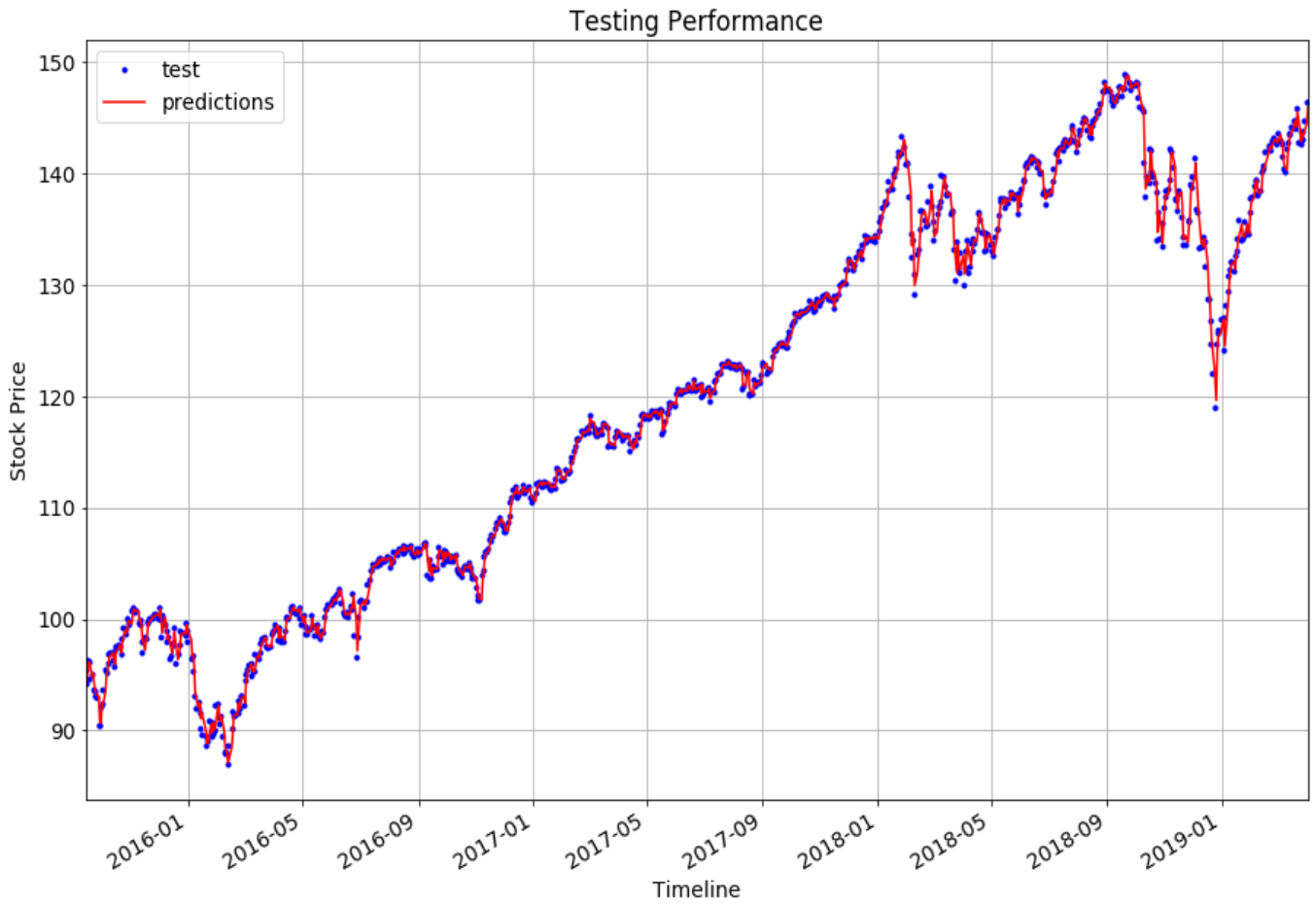
| param | original | after_tuning |
|---|---|---|
| n_estimators | 100.000 | 120.000 |
| max_depth | 3.000 | 8.000 |
| learning_rate | 0.100 | 0.100 |
| min_child_weight | 1.000 | 6.000 |
| subsample | 1.000 | 0.900 |
| colsample_bytree | 1.000 | 1.000 |
| colsample_bylevel | 1.000 | 1.000 |
| gamma | 0.000 | 1.000 |
| rmse | 0.668 | 0.666 |
| mape_pct | 0.603 | 0.601 |

Graphs for tuning of certain hyperparameters are shown below:




Performance of the final model using the tuned hyper-parameters is shown below:




**Future Work:**

- Data collection for Indian Stock Market is currently in lines from the official nseindia.com website through scraping by selenium.
- We intend to explore the dual behavior of stock in Indian Stock Market spectrum.

- Nifty 50 and residing stocks of Nifty 50 shall be used for stock market prediction.
- Various research papers related to Indian Stock Market predictions are being studied to understand the possibilities and challenges involved in the stock market prediction.
- Real-time stock prediction and stock prediction with higher time horizon is also an explorable work which is not a much-researched topic or study.