Faraz Mostafaeipour

Applications of Machine Learning on the Stock Market

Dr. Qiang Zhu

9 December 2019

Introduction

The Stock Market has been one of the most controversial and interesting databases for machine learning (ML) and market timing. Day to day, methods, predictions, and models for machine learning and artificial intelligence are becoming more advanced. The idea of being able to predict and time the stock market is an old and controversial topic. The main critique of timing the market is that the stock market is affected by people in a collective manner. It's controlled and influenced by an infinite number of variables and subject to supply-and-demand trends. It is important to note that the analysis tended to by the ML, will be a technical analysis as opposed to a fundamental one.[1] This paper will aim to use different stocks with different behaviors (stable vs highly variable) and different machine learning techniques to attempt to predict the Stock Market's trend.

The stocks used are Microsoft (MSFT), Nike (NKE), Apple (AAPL), The Travelers Companies (TRV), Intel (INTC), and Tata Beverages (TATAGLOBAL). The main stock used for model testing was TATAGLOBAL. This is due to Tata Beverages having a spike in price due to external factors in which the machine learning algorithms have no knowledge of. For the machine learning aspect, the data will be split into a training set and a validation set. The training set will be used to create a model. The validation set will be to cross referenced to the predictions of the ML algorithm and an error rate can be found. The methods used are Linear Regression, K-Neighbor, Auto-ARIMA, and Long Short Term Memory (LSTM).

[1] A Fundamental Analysis involves analyzing the company's future profitability based on financial features and aspects to determine how well it performs in its respective market. A Technical Analysis tends to evaluate the stocks behavior based on statistical trends and charts.

Data Loading

First, the dataset for Tata Beverages will be loaded. The data is displayed with variables such as open, low, volume, and close that describe the stock.

```
Stock:  NSE-TATAGLOBAL11.csv
```

	Date	Open	High	Low	Last	Close	Total Trade Quantity	Turnover (Lacs)
0	2018-10-08	208.00	222.25	206.85	216.00	215.15	4642146.0	10062.83
1	2018-10-05	217.00	218.60	205.90	210.25	209.20	3519515.0	7407.06
2	2018-10-04	223.50	227.80	216.15	217.25	218.20	1728786.0	3815.79
3	2018-10-03	230.00	237.50	225.75	226.45	227.60	1708590.0	3960.27
4	2018-10-01	234.55	234.60	221.05	230.30	230.90	1534749.0	3486.05

Profit-Loss calculations are usually based on the closing price of the day, hence, the target variable for the machine learning algorithms will be the ‘Close’ column of the data. The other data will lead to added noise and overfitting due to its high correlation with the target variable (closing price). In order to visualize the data, the indexing of the dataframe must be updated in order to support the standard date format. This can be simply converted with a ‘DateTime’ converter.

```
1 df['Date'] = pd.to_datetime(df.Date,format='%Y-%m-%d')
2 df.index = df['Date']
```

The graph follows as:

Before dividing the data into training and validation sets, various time features will be added as supplemental training input of the ML. For example, the closing price for a particular day will have attributes of time such as: day of week, day of year, end of month, end of quarter, etc. This not only helps provide supplemental information to train the ML algorithm but is effective in stock market analyses due to the quarterly periodicity of the market and trend of prices on different days. A fast and effective way of adding date features to the data is by using the Fastai library. Using 'Fastai.add_datepart(),' the features can be added.

```
from fastai.structured import  add_datepart
add_datepart(new_data,'Date')
```

The new data will follow as:

Date	Close	Year	Month	Week	Day	Dayofweek	Dayofyear	Is_month_end	Is_month_start	Is_quarter_end	Is_quarter_start	Is_year_end	Is_year_start
2018-10-01	230.90	2018	10	40	1	0	274	False	True	False	True	False	False
2018-10-03	227.60	2018	10	40	3	2	276	False	False	False	False	False	False
2018-10-04	218.20	2018	10	40	4	3	277	False	False	False	False	False	False

Now with the target variable (closing price) and training variables (date and features) have been defined, ML algorithms can be utilized.

```
    train = new_data[:987]
    valid = new_data[987:]
    x_train = train.drop('Close', axis=1)
    y_train = train['Close']
    x_valid = valid.drop('Close', axis=1)
    y_valid = valid['Close']
```

1. Linear Regression

Linear Regression is a method for fitting a best-fit-line between independent and dependent variables. The best-fit-line is the line with the minimum amount of error from all the points. In the context of the Stock Market, the ML algorithm tries to find the optimal line based on the multimodal training set to establish a pattern. This follows the relationship of

$$Y = W_1 X_1 + W_2 X_2 + W_3 X_3 + \ldots + W_n X_n$$

where $X_1, X_2, X_3, \ldots, X_n$ represent the independent variables (date and features in this context) and $W_1, W_2, W_3, \ldots, W_n$ represent the weight of the variables. With the data set split into the training sets, the Linear Regression model will be used, and its predictions will be stored.

```
27 model = LinearRegression()
28 model.fit(x_train,y_train)
29 preds = model.predict(x_valid)
```

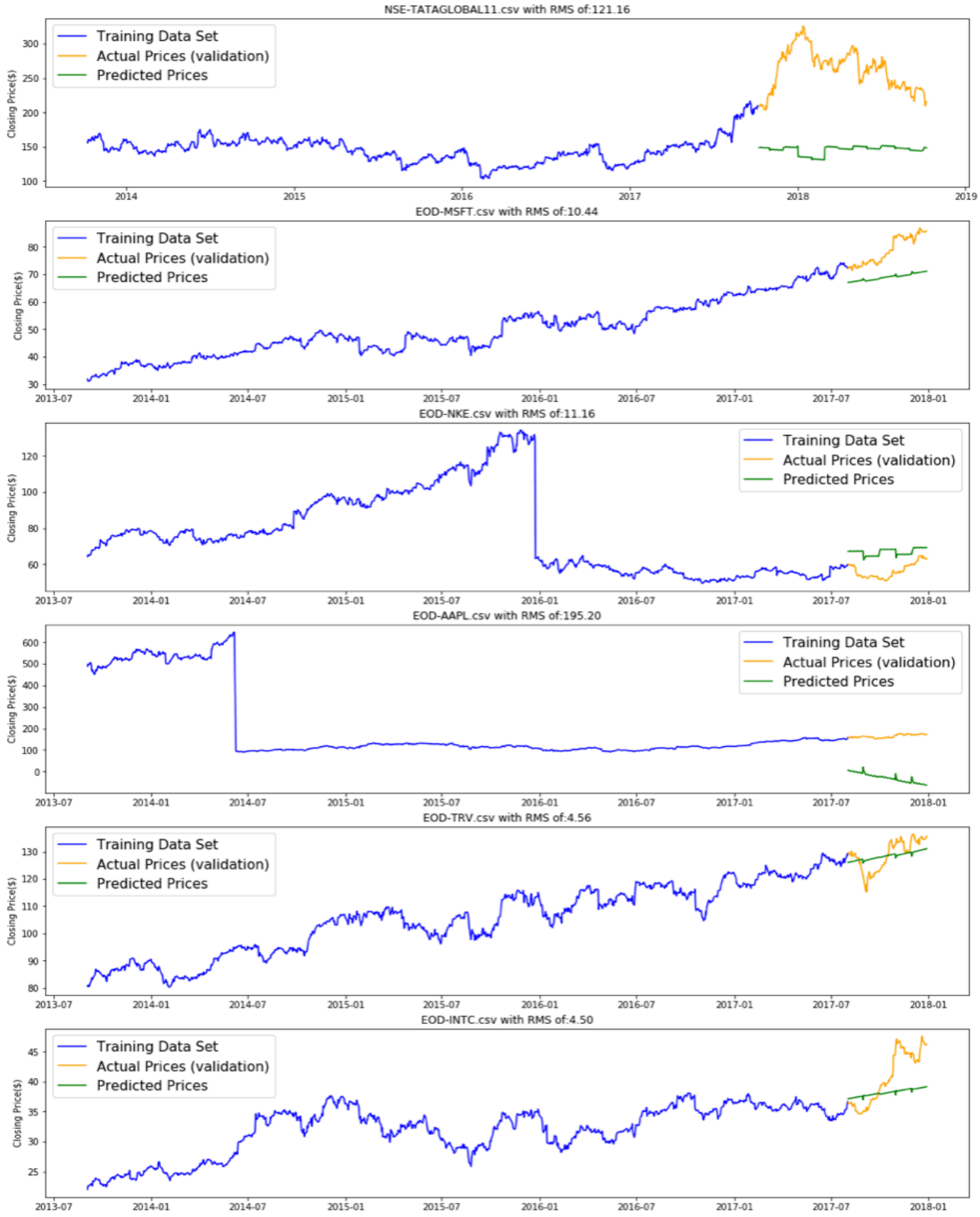
In order to determine the accuracy and efficiency of the model with the validation set, the Roots Mean Squared (RMS) value will be used to measure the average variation from the validation set.

```
31 rms=np.sqrt(np.mean(np.power((np.array(y_valid)-np.array(preds)),2)))
32 print("Root Means Squared value for Linear Regression is: {0:.2f}".format(rms))
```

```
Root Means Squared value for Linear Regression is: 121.16
```

The algorithm determines an RMS value of $121. This is a high error value given the price variation of the stock. To better visualize this, the graph follows accordingly:

Now this analysis will be done on the other stocks as well to recognize relationships between this model and stock variants.



Inference

Based on the high RMS values and graphs, Linear Regression doesn't seem to be an efficient and reliable method for predicting the stock market. The disadvantages of Linear Regression become apparent due to its overfitting to the date and month column. Instead of giving more weight to the points preceding the prediction point, it emphasizes the data from a year/month ago. For the Tata Beverage stock, the ML algorithm recognized a dip in price in January 2016 and 2017, predicting a dip in price in January 2018. The Linear Regression model doesn't take covariate and response variable into consideration and establishes a linear relationship between the closing price and the date.

2. K-Nearest Neighbor

K-Nearest Neighbor (KNN) is method of regression using a neighbor-based model. The model classifies labels to the training set of data used as a reference later. Then, the label assigned to a query point is computed based on the mean of the labels of its nearest neighbors (Pedregosa). This type of regression considers the qualities of the variables around the query point and attributes the mean of those qualities as its own. The query point determines its closet neighbors based on different methods such as: Euclidean Distance, Manhattan Distance, and Hamming Distance. To determine the optimal distance for the proximity of neighborhood, a grid search is done. Thereafter, the model was fitted with the training set and the predictions were stored:

```
32 params = {'n_neighbors':[1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20]}
33 knn = neighbors.KNeighborsRegressor()
34 model = GridSearchCV(knn, params, cv=5)
35
36 model.fit(x_train,y_train)
37 preds = model.predict(x_valid)
```

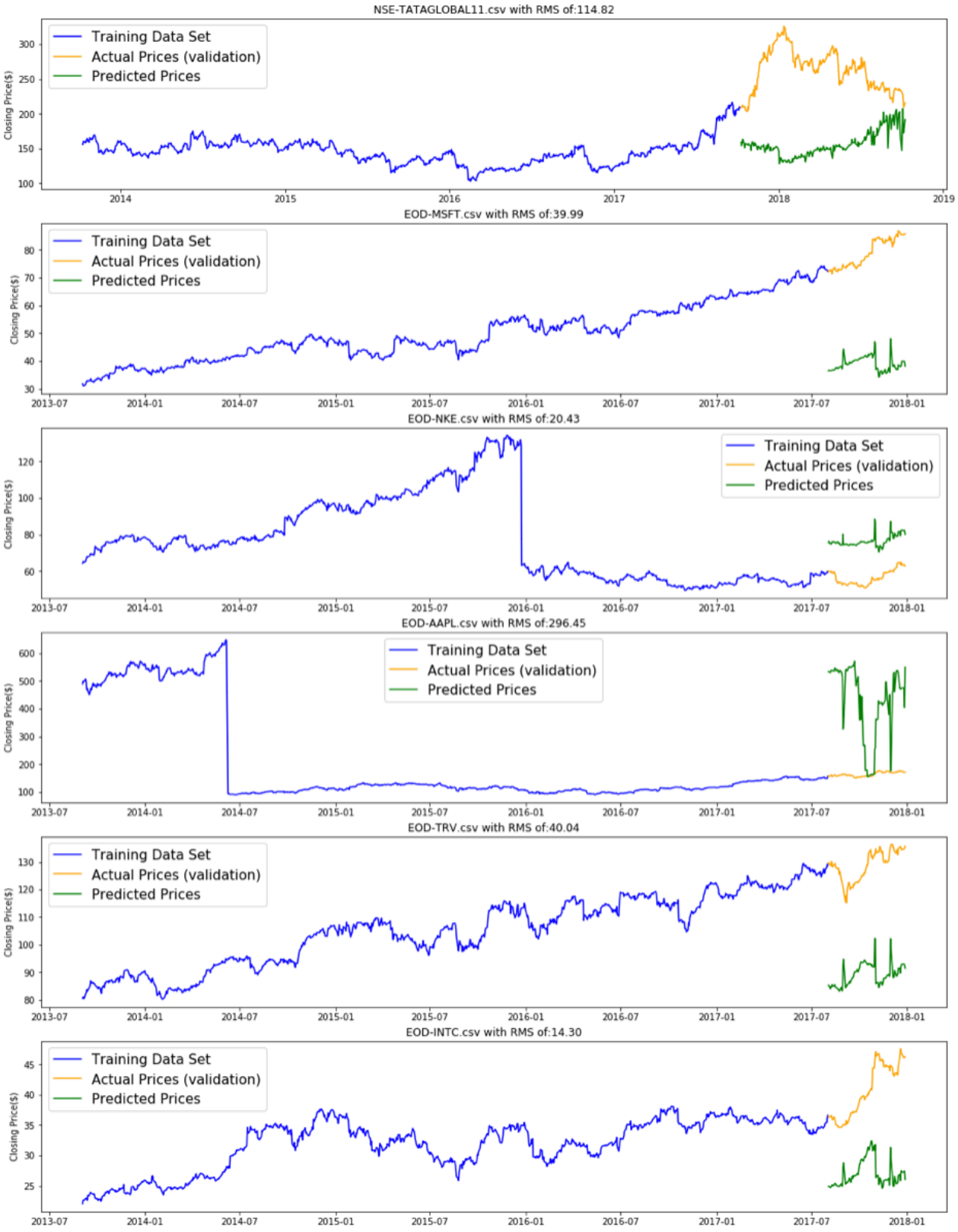
In order to determine the accuracy and efficiency of the model with the validation set, the Roots Mean Squared (RMS) value will be used to measure the average variation from the validation set.

```
39 rms=np.sqrt(np.mean(np.power((np.array(y_valid)-np.array(preds)),2)))
40 print("Root Means Squared value for K-Nearest Neighbor is: {0:.2f}".format(rms))
```

```
Root Means Squared value for K-Nearest Neighbor is: 114.82
```

The algorithm determines an RMS value of $114. Like Linear Regression, this model has a high error value given the price variation of the stock. To better visualize this, the graph follows accordingly:

Now this analysis will be done on the other stocks as well to recognize relationships between this model and stock variants.



Inference

Again, based on the high RMS values and the non-conforming fit on the graphs, K-Nearest Neighbor isn't a reliable and trustworthy method of regression. A drawback of this method is that it does not account for the data being sequential, giving equal weight to all data points uniformly and recognizing a pattern. Another fault of this program is that it tends to become less accurate as the number of features increase. This comes from the dimensionality curse, when a dataset has too many features. This causes an equidistant distance from all other feature. With the Stock Market being highly multimodal and is under the influence an infinite amount of variable, the K-Nearest Neighbor method would fail to accurately describe the stock market.

3. ARIMA

ARIMA stands for Auto-Regressive Integrated Moving Averages. It is a very popular statistical method for forecasting sequential and time-series data. ARIMA works on the assumptions:

- The time series is stationary, meaning the mean and standard deviation don't vary with time. The series can be made stationary using differencing techniques to better fit the model.
- Since ARIMA utilizes previous values to predict future values, the input for the model must be a univariate series.

ARIMA operates based on certain parameters p, q, d. The 'p' parameter is the autoregressive term of the model, referring to using the past values to predict the future value. A PACF graph is used to determine the p-value. The 'd' parameter is the differencing operator for the model, providing input on the degree of differencing required to create a stationary time series. The d-value is found using ADF and KPSS tests. The 'q' parameter is the moving average term of the model, referencing number of past forecast errors used to predict the future values. The q-value is determined by an ACF plot.[2] Often times, the selection and tuning of these parameters to determine the best fit is time consuming and involves a deep understanding in the data structure. Therefore, the Auto-ARIMA function will be used to find the optimal parameters for the fit. The model was fitted with the training set and the predictions were stored:

```
training = train['Close']
validation = valid['Close']
model = auto_arima(training
                   ,start_p=1, start_q=1,max_p=3, max_q=3, m=12,start_P=0, d=1
                   ,D=1, trace=True,error_action='ignore'
                    ,seasonal=True, suppress_warnings=True)
model.fit(training)
forecast = model.predict(n_periods=len(valid))
```

In order to determine the accuracy and efficiency of the model with the validation set, the Roots Mean Squared (RMS) value will be used to measure the average variation from the validation set.

```
rms=np.sqrt(np.mean(
    np.power((np.array(valid['Close'])-np.array(forecast1['Prediction'])),2)))
print("Root Means Squared value for the ARIMA method is:",rms)
```

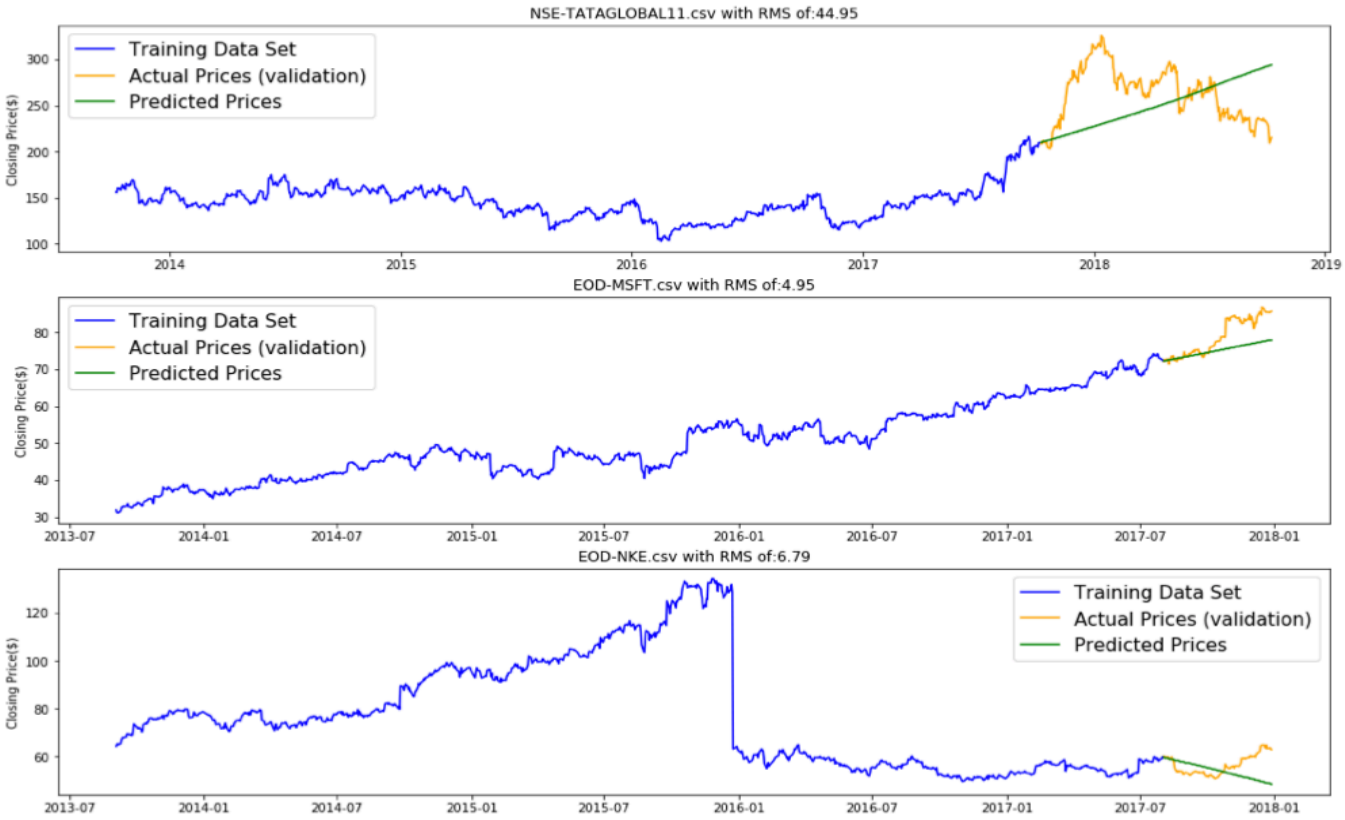
```
Root Means Squared value for the ARIMA method is: 44.95275646807023
```

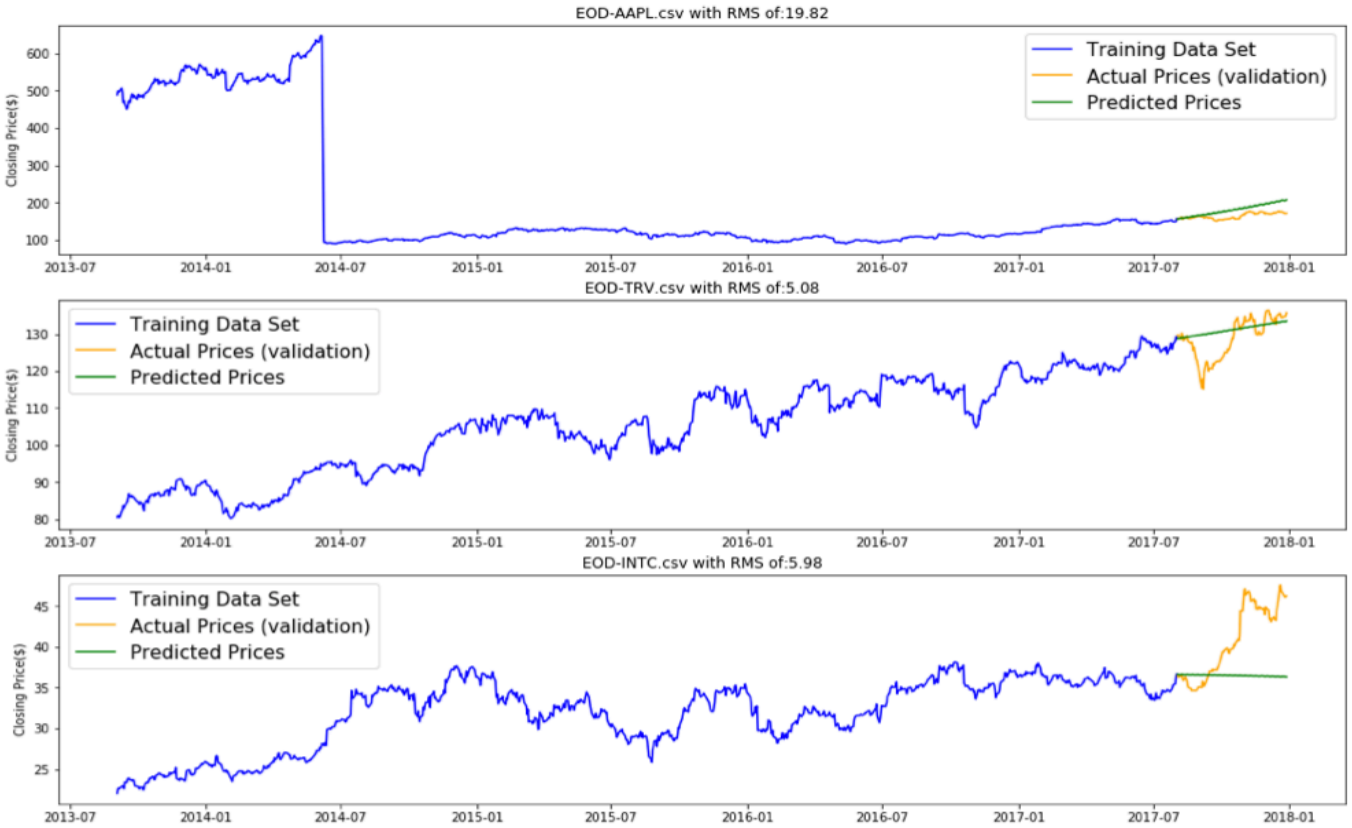
The algorithm determines an RMS value of $45. Based on this, ARIMA performs better than Linear Regression and K-Nearest Neighbors due to accounting for the time-dependency of the target variable but still has an RMS value high enough to be out of comfort for investing.

[2] The series of tests and plots mentioned are popular statistical measures namely: Partial Auto Correlation Function (PACF), Auto Correlation Function (ACF), Augmented Dickey–Fuller (ADF), and Kwiatkowski–Phillips–Schmidt–Shin (KPSS).

To better visualize the error in ARIMA forecasting, the graph follows accordingly:

Now this analysis will be done on the other stocks as well to recognize relationships between this model and stock variants.





Inference

Using the ARIMA method of forecasting resulted in a better RMS value compared to the previous methods. The RMS value is still large enough to not accurately predict the market but does a good job predicting the overall trend of a stock. This can be a dangerous tool due to its lack of consideration for outside features. For example, the method determined an increasing trend for the Tata Beverages Company but a weak earnings report or rumors about the company's sustainability can drop the closing price without the algorithm ever accounting for it. It also lacks a key feature of recognizing the seasonality of the data. Assigning a seasonality analysis for the method becomes very important due to the periodic nature of the stock market. This is not to state that the ARIMA method is useless though, this method can be used as an investing tool to better understand the trend of a stock based on its most recent closing prices. The coupling of this method with other methods can result in a better understanding of timing the market and recognizing trends in the data.

4. Long Short Term Memory (LSTM)

Long Short Term Memory (LSTM) is a method of regression and data analysis, highly effective in sequential series problems. It is an upgraded method based on Recruiting Neural Networks (RNN) with the advantage of possessing a memory/forgetting function to consider weighted and significant pieces of data. This memory/forgetting function of the algorithm is executed through cell states like neural networks with hidden layers. A key feature of this forecasting method is the weight it not only attributes to important pieces of data stored in the memory, but to the most recent data prior to predicating. For example, it uses the previous 60 data points proceeding the query (prediction) point, with the weighted information stored in the memory cell stored from all the other data points. The entire data set is run through the cells of the model and sequentially uses the previous 60 points to learn the model. This prioritization of the most recent information holds valuable in sequential series, preventing from overfitting to the month/year.

The LSTM method operates based on common dependencies:

- The previous cell state
- The previous hidden state (i.e. this is the same as the output of the previous cell)
- The input at the current time step

In context of the stock market this can translate into:

- The trend that the stock has taken on in previous days, increasing or decreasing relative to that time.
- The price of the stock on the previous day, because traders consider the closing price of the previous day before buying
- The factors that can affect the price of the stock for today. This can be external factors such as a new criticized company policy, beating of earnings report, or change in executive board

The model was fitted with the training set and the predictions were stored:

```
27
28          # create and fit the LSTM network
29  model = Sequential()
30  model.add(LSTM(units=50, return_sequences=True, input_shape=(x_train.shape[1],1)))
31  model.add(LSTM(units=50))
32  model.add(Dense(1))
33
34  model.compile(loss='mean_squared_error', optimizer='adam')
35  model.fit(x_train, y_train, epochs=5, batch_size=1, verbose=1)
36
37          #predicting 246 values, using past 60 from the train data
38  inputs = new_data[len(new_data) - len(valid) - 60:].values
39  inputs = inputs.reshape(-1,1)
40  inputs  = scaler.transform(inputs)
41
42  X_test = []
43  for i in range(60,inputs.shape[0]):
44      X_test.append(inputs[i-60:i,0])
45  X_test = np.array(X_test)
46  X_test = np.reshape(X_test, (X_test.shape[0],X_test.shape[1],1))
47  closing_price = model.predict(X_test)
48  closing_price = scaler.inverse_transform(closing_price)
```

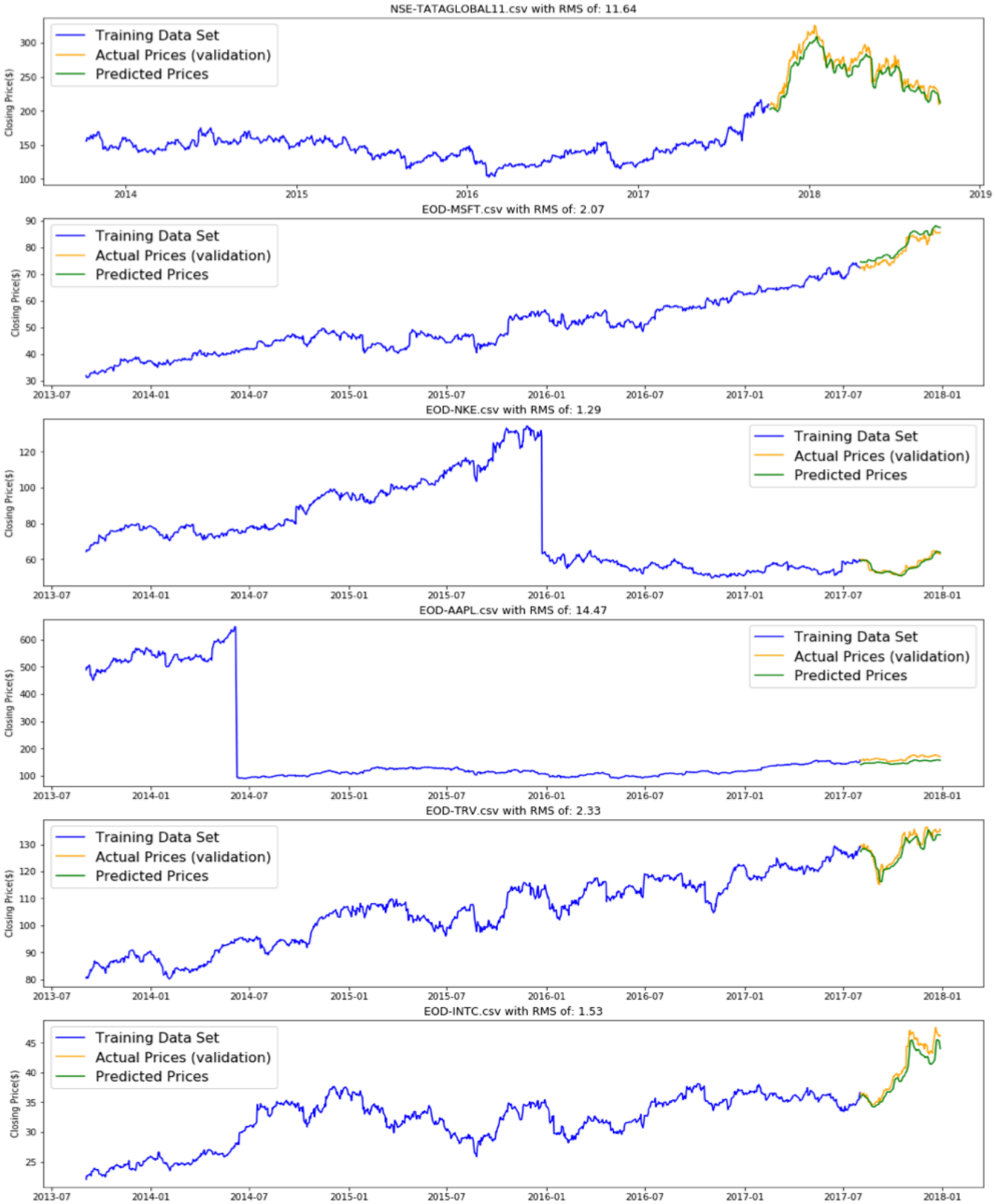
In order to determine the accuracy and efficiency of the model with the validation set, the Roots Mean Squared (RMS) value will be used to measure the average variation from the validation set.

```
50  rms=np.sqrt(np.mean(np.power((valid-closing_price),2)))
51  print("Root Means Squared value for Long Short Term Memory is: {0:.2f}".format(rms)
```

```
Root Means Squared value for Long Short Term Memory is: 6.11
```

The algorithm determines an RMS value of $6.1. Based on this, Long Short Term Memory outperformed all other methods of regression. With an average deviance of $6, an investor is more probable to accept a risk at that rate, given the range of variation along the closing price. To better visualize the error in ARIMA forecasting, the graph follows accordingly:

Now this analysis will be done on the other stocks as well to recognize relationships between this model and stock variants.



Inference

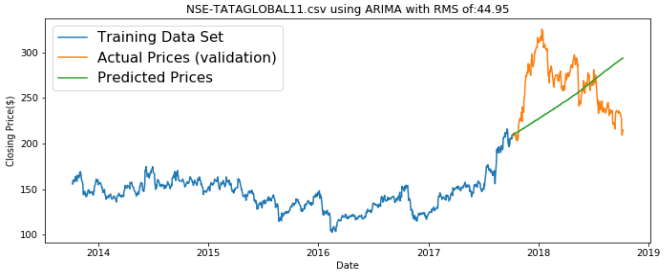
Using the Long Short Term Memory method to analyze the Stock Market has by far been the most accurate method of ML. With its average RMS value being well below $10, an investor would invest in the stock market confidently. This is because the percent profit/loss is calculated relative to the stock price. Y' is the actual stock's price if we assume x-percent gain/loss from the predicted value (Y).

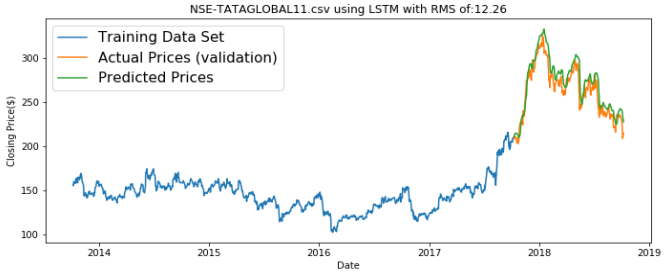
$$Y' = Y \pm xY = Y(1 \pm x)$$

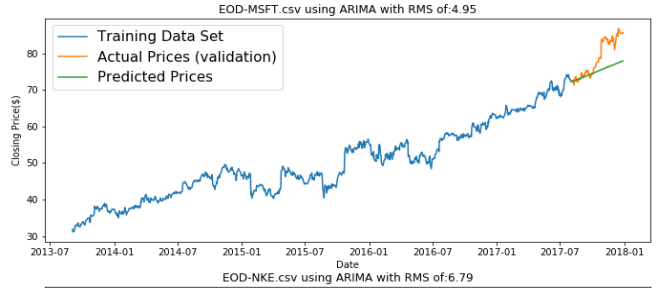
$$x = \frac{\pm(Y' - Y)}{Y} \; where \; RMS = Y' - Y$$

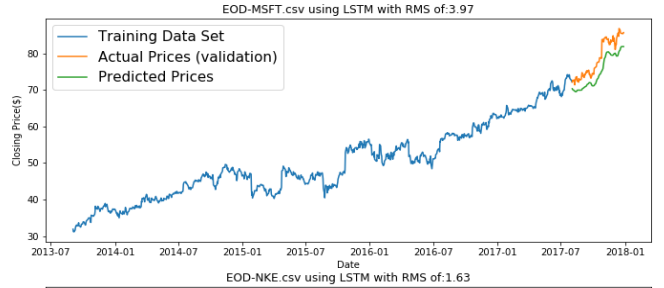
Therefore, in the context of Tata Beverages Company, $x = 3\%$ is the extreme average loss which is arguable well enough for investing. This method did the best out of all the other ML algorithms stated due to its accountability towards sequential data, utilizing the most recent data with more weight, and taking the seasonality of the Stock Market into consideration. One of the drawbacks to this model is that the entire data set is quantumly passed through the cells to predict the query point; this is not hardware friendly and has a high cost of computation (Culurciello). It also requires a greater volume of data for training to achieve accurate results. This method of analysis can be coupled with Auto-ARIMA to understand the overall trajectory of the stock trend, and utilize LSTM to account for season ability, noisiness, and short term prediction over the long term analysis of the dataset.

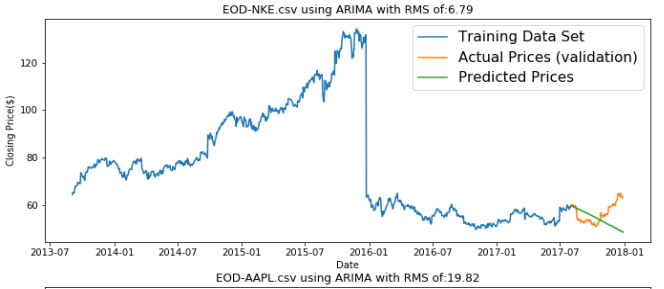
So in an analysis of the stocks using the ARIMA method to determine the stocks trend and using the LSTM method to find the forecasting in the future using the most recent data, we'll

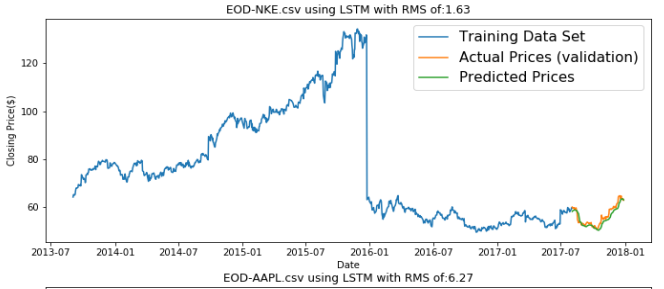


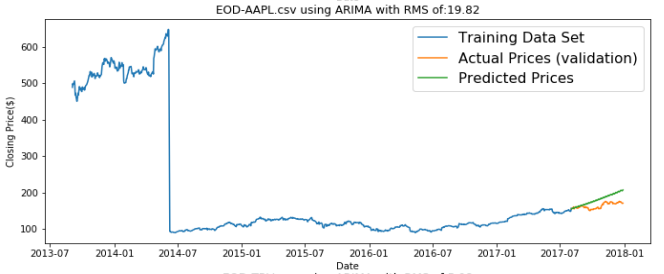


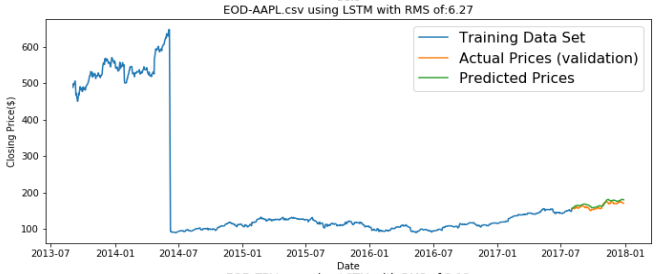


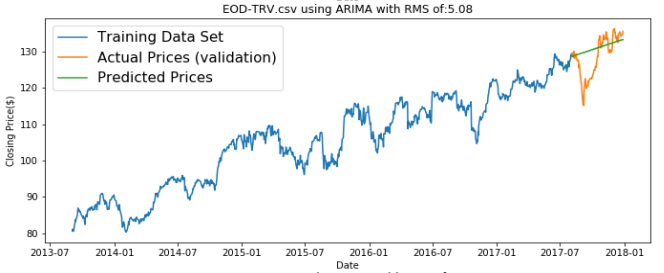


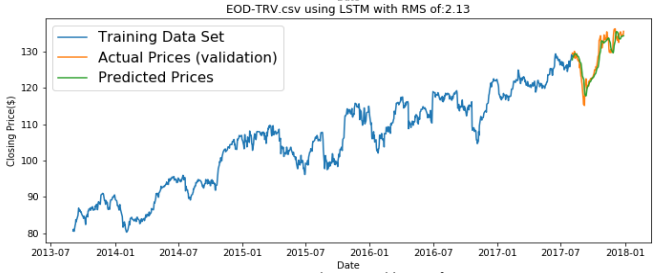


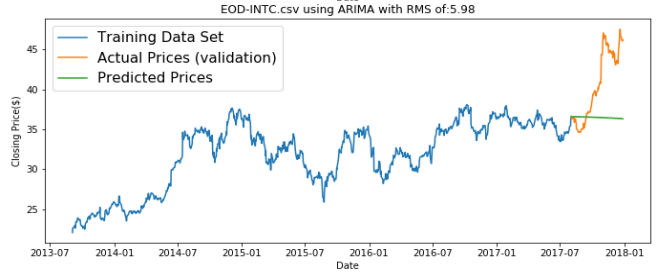


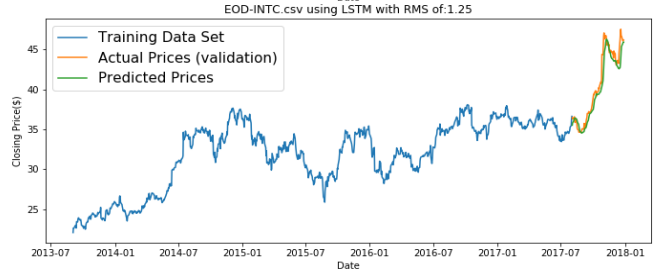












have:

Works Cited

Chatterjee, Sharmistha. “ARIMA/SARIMA vs LSTM with Ensemble Learning Insights for Time Series Data.” *Medium*, Towards Data Science, 13 May 2019, https://towardsdatascience.com/

Culurciello, Eugenio. “The Fall of RNN / LSTM.” *Medium*, Towards Data Science, 10 Jan. 2019, https://towardsdatascience.com/

Mostafaeipour, Faraz. “Stock Market Project.” 2 Dec. https://github.com/farazmost/ComputationalPhysics300/blob/master/Project%202%20material/STOCK%20Market%20Project.ipynb

Pedregosa *et al.*, *Scikit-learn: Machine Learning in Python*, JMLR 12, pp. 2825-2830, 2011.
for Time Series Data.” *Medium*, Towards Data Science, 13 May 2019, https://towardsdatascience.com/

Singh, Aishwarya. “Build High Performance Time Series Models Using Auto ARIMA in Python and R.” *Analytics Vidhya*, 7 May 2019, https://www.analyticsvidhya.com/

Singh, Aishwarya. “Predicting the Stock Market Using Machine Learning and Deep Learning.” *Analytics Vidhya*, 4 Sept. 2019, https://www.analyticsvidhya.com/

Srivastava, Pranjal. “Essentials of Deep Learning: Introduction to Long Short Term Memory.” *Analytics Vidhya*, 23 Dec. 2017, https://www.analyticsvidhya.com/