Short-Term Stock Market Price Prediction

Farhanur Rahim Ansari
Vidhey Oza

1. Introduction

1.1. Motivation

Predicting stock prices is of high importance to stock-brokers as well as individual traders. On top of that, when intraday prices are predicted, the traders have an almost real-time analysis of the stock based on the trades done on the day, which is a great value addition. Technical analysis of stock prices has been done for long, and technical indicators have been used to predict the direction in which the price will go. If a machine learns how these indicators “indicate” the change in prices, the analysis that has been done manually till now can be automated.

1.2. Goal

Our goal is to predict stock prices listed on NYSE and NASDAQ on a short-term basis using technical indicators. We plan to predict prices for 1 minute, 5 minutes, and 10 minutes time interval. This makes it a classic case of **Time-Series Analysis**. Since our aim involves predicting a continuous value which is the closing price of a stock for a particular time period, this makes it a **Regression Problem**. We plan to train and verify our dataset on as many as 5 regression models.

2. Dataset

2.1. Overview

Our dataset consists of stock prices of 5 different companies. It includes 4 tech companies namely, Apple (AAPL), Amazon (AMZN), Microsoft (MSFT), and Tesla (TSLA), and 1 retail company namely, Walmart(WMT). The data was fetched in CSV file format. It consisted of minute-wise stock details from 1st January 2018 till 11th November 2019. There are a total of 181578 records and 9 features for each company.

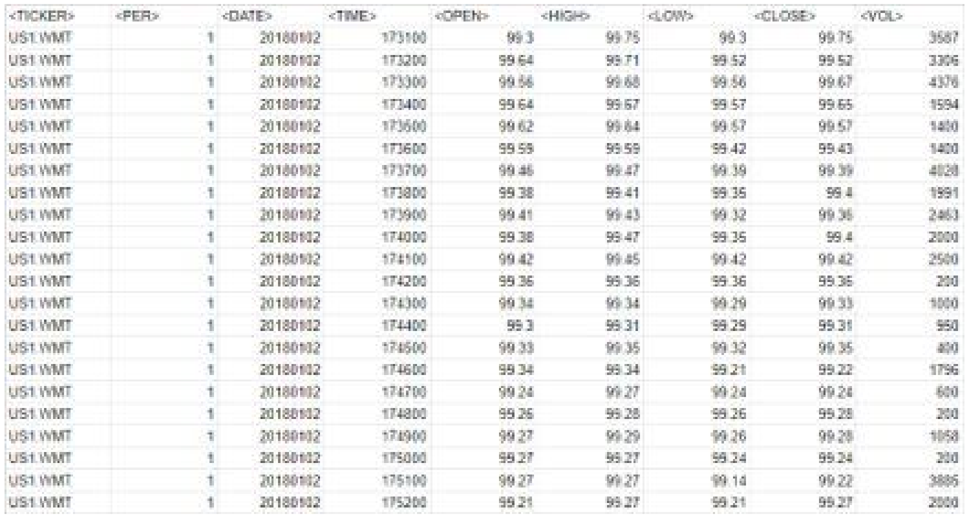
The dataset consists of 5 main features:-

Open: This indicates the opening price of a stock record at a particular time interval in USD.
High: This indicates the highest price of a stock record at a particular time duration in USD.
Low: This indicates the lowest price of a stock record at a particular time duration in USD.
Close: This indicates the closing price of a stock record at a particular time interval in USD.
Vol: This indicates the number(volume) of transactions that have taken place for that stock record in that particular time duration.

We also created datasets for the 5 minute and 10 minute time intervals using 1 min as our base dataset. For creating such records, we re-computed the 5 main features by considering the records in a group of 5 or 10 respectively. We did the following changes:-

Open: It was replaced by the opening price of the first record.
Close: It was replaced by the closing price of the last record.
High: It was replaced by the maximum value among the group of records.
Low: It was replaced by the minimum value among the group of records.
Vol: It was replaced by the sum of all vol values in the group.

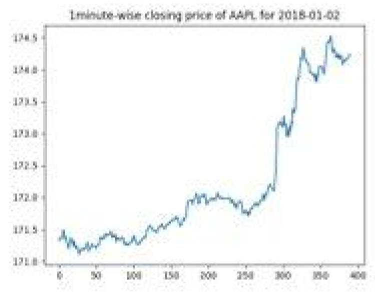
The dataset also contained 4 other features namely, Ticker, Per, Date, and Time. A glimpse of the dataset is given below.

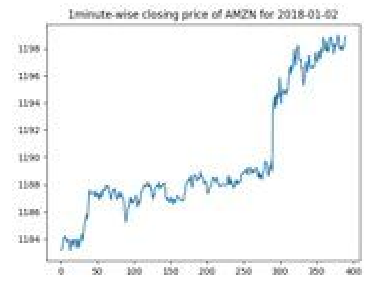


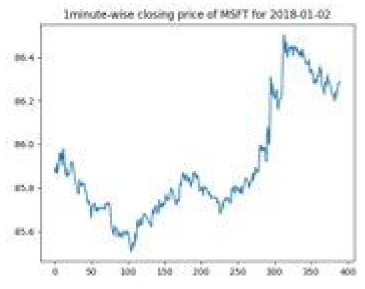
<TICKER>	<PER>	<DATE>	<TIME>	<OPEN>	<HIGH>	<LOW>	<CLOSE>	<VOL>
US1.WMT	1	20180102	173100	99.3	99.75	99.3	99.75	3587
US1.WMT	1	20180102	173200	99.64	99.71	99.52	99.52	3306
US1.WMT	1	20180102	173300	99.56	99.68	99.56	99.67	4376
US1.WMT	1	20180102	173400	99.64	99.67	99.57	99.65	1594
US1.WMT	1	20180102	173600	99.62	99.64	99.57	99.57	1400
US1.WMT	1	20180102	173600	99.59	99.59	99.42	99.43	1400
US1.WMT	1	20180102	173700	99.46	99.47	99.39	99.39	4028
US1.WMT	1	20180102	173800	99.38	99.41	99.35	99.4	1991
US1.WMT	1	20180102	173900	99.41	99.43	99.32	99.36	2463
US1.WMT	1	20180102	174000	99.38	99.47	99.35	99.4	2000
US1.WMT	1	20180102	174100	99.42	99.45	99.42	99.42	2500
US1.WMT	1	20180102	174200	99.36	99.36	99.36	99.36	200
US1.WMT	1	20180102	174300	99.34	99.34	99.29	99.33	1000
US1.WMT	1	20180102	174400	99.3	99.31	99.29	99.31	950
US1.WMT	1	20180102	174500	99.33	99.35	99.32	99.35	400
US1.WMT	1	20180102	174600	99.34	99.34	99.21	99.22	1796
US1.WMT	1	20180102	174700	99.24	99.27	99.24	99.24	600
US1.WMT	1	20180102	174800	99.26	99.28	99.26	99.28	200
US1.WMT	1	20180102	174900	99.27	99.29	99.26	99.28	1058
US1.WMT	1	20180102	175000	99.27	99.27	99.24	99.24	200
US1.WMT	1	20180102	175100	99.27	99.27	99.14	99.22	3886
US1.WMT	1	20180102	175200	99.21	99.27	99.21	99.27	2000

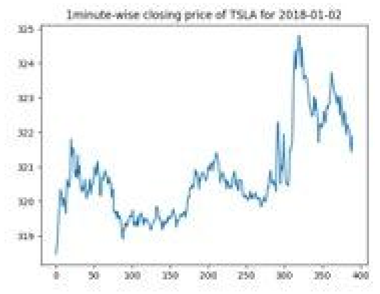
2.2. Preliminary Data Analysis

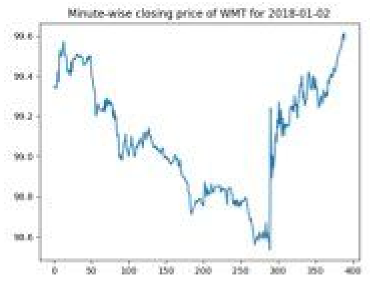
Closing price:







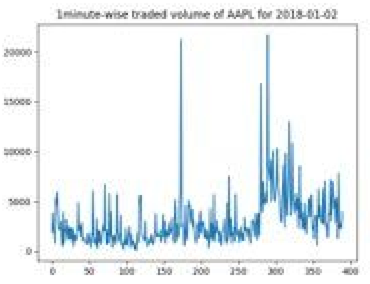


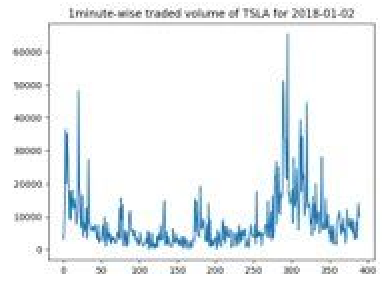


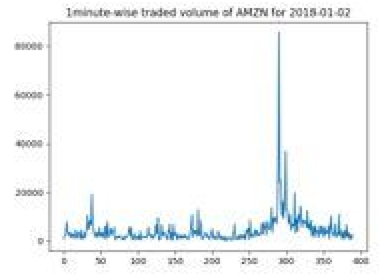
Closing price is the response variable which we aim to predict through machine learning. Using the above preliminary analysis we can observe it's minute-wise variation for a day of 2018-01-02, for all the 5 stocks. It's important to notice the amount of variation one can observe within a day for the closing price. Hence, it further strengthens the importance of intraday stock price prediction.

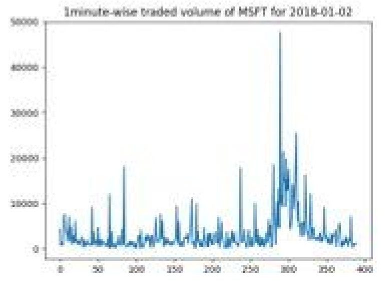
One interesting thing to notice here is the scale of different stocks. There is a huge difference in the scale of Amazon's stock prices and other company's stock prices. Amazon's stock prices are in the low thousands, whereas for other companies combined it is currently in the lower to mid-hundreds.

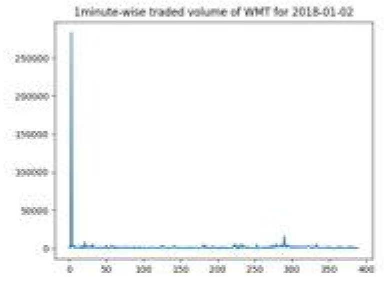
Volume:











Using the above preliminary analysis we can observe the minute-wise variation of volume for a day of 2018-01-02, for all the 5 stocks. Here, we can notice that the minute-wise volume of Apple, Microsoft, and Tesla is quite volatile. On the other hand, the minute-wise volume of Amazon is moderately volatile and for Walmart, it's the least.

3. Feature Engineering

3.1. Technical Indicators

We started with 5 features in the dataset: opening price, closing price, high, low and volume of stock traded. All these values were with respect to the time period of one minute. That means, for example, the high price is the high trading price in the given minute.

From that we engineered a total of 22 features from 6 categories:

- Smoothing (moving averages)
- Momentum (shows the continuous 'momentum' of stock price)
- Volume (analysis from the volume of stock traded)
- Overbought/oversold signals (generate signals when stock is overbought or oversold)
- Volatility (analysis from the variability of stock in a given period)
- Trends (analysis of the general trend of price movement)

This way, we derived the following 22 technical indicators: MACD, RSI, William's %R, Stochastic %K and %D, MFI, ROC, SMA, WMA, EMA, HMA, CCI, ADL, CMF, OBV, EMV, ATR, Mass Index, Ichimoku clouds (made from 2 indicators), Aroon Index, ADX.

Another aspect of technical indicators to be noted is the window of the time period for which they are calculated. Indicator window essentially denotes the number of data points (minutes in this case) the algorithm looks back to generate the indicator values. We checked for 5-min, 10-min, 15-min, 20-min windows on one stock (AAPL), and found the best results for the 10-min window.

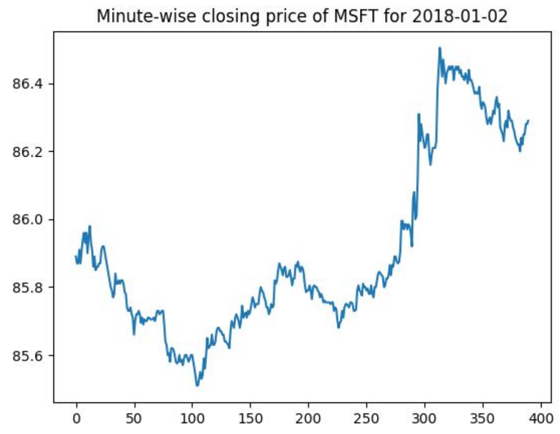
3.2. Analysis

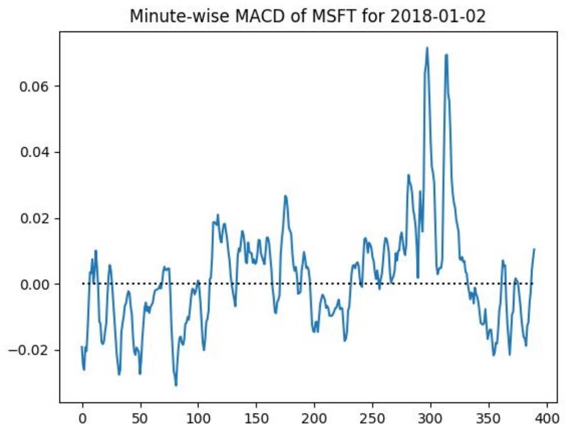
For a basic understanding of indicators, we take the example of the MACD (Moving Average Convergence/Divergence) indicator. This is a complex momentum indicator, meaning it essentially denotes the momentum of the price at a given point in time. Thus, for MACD to be continuously high the price need to be continuously increasing for that period. In the accompanying examples, this property is made clear (figures on the next page).

For MSFT, the MACD value increases at around the 300th data point since the price is increasing continuously for around 40 data points. Then, when it starts dropping at around the 320th data point, the MACD drops to negative value quickly as well.

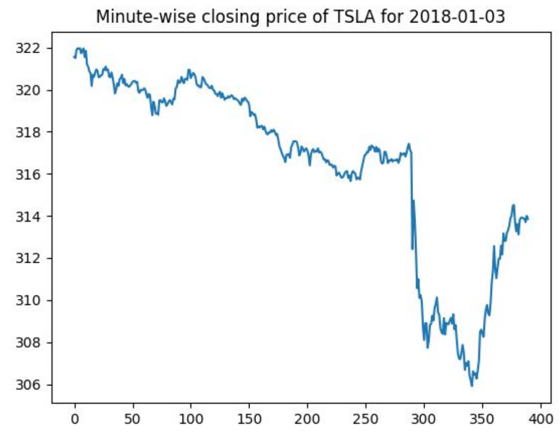
On the other hand, for TSLA, we see that when the price suddenly dropped at the 300th data point, MACD stayed negative for some time before recovering (since the price recovered to around the previous values).

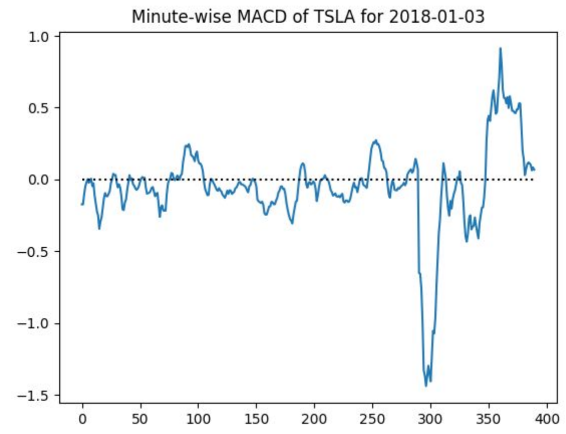
MSFT closing price and MACD for January 2nd, 2018





TSLA closing price and MACD for January 3rd, 2018





4. Data Preprocessing

4.1. Time Intervals

Our goal is to predict stock market prices on a short term basis. By short term basis, we mean how closing prices vary intraday. We aim to predict prices with the least error by optimizing the time period. Hence the lesser the time interval, the more real-time our prediction is. Our dataset contains records on a minute basis. In order to achieve our goal, we trained our model by trying to fit our model at different time intervals. And accordingly, we changed our 5 main features.

We tried 3 different time intervals – 1 minute, 5 minutes, and 10 minutes. Each interval behaved differently as their volatility varies over time. For 1 minute we prepared a dataset containing 50000 records. Similarly, for 5 minutes and 10 minutes, we prepared different datasets containing 20000 and 10000 records respectively.

4.2. Data Scaling

We normalized the dataset features using Z-score normalization in order to achieve standardization in our dataset. The formula for Z-score normalization is –

$$X = \frac{X - \mu_X}{\sigma_X}$$

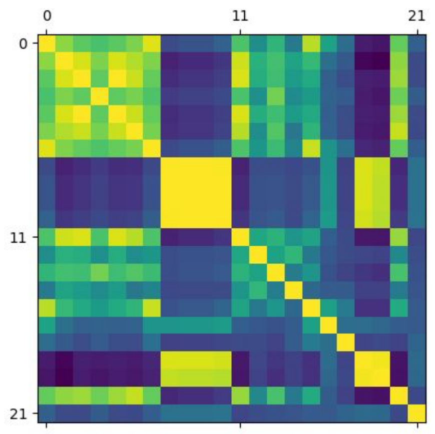
We did not opt for min-max scaling as this type of scaling fix an upper and lower limit on the test dataset to be normalized. Whereas, in our case, there are chances that the new data point might have features with a value greater than the maximum value in the training dataset or a value less than the minimum value in the training dataset.

4.3. Data Cleaning

Other than the main features and the 22 engineered features, the dataset also contained some unwanted features which came along with the CSV file, like ticker, per, date, and time, which were of no use in the model training part. So we removed these columns before training the model.

After the new features were engineered, some null values were introduced at the start and end of the dataset. These values were generated because of the nature of some of the categories of the new features like moving averages, which work on a collection of data points. These null values are represented by NaN in the dataset. There were 100 such values at the start and 100 values at the end of the dataset. We removed all such records.

4.4. Feature Selection



Selected 16 features out of the 22 engineered features. Feature selection was done by understanding the mathematical interpretation of the technical indicators and removing the noisy ones. This was also verified by analyzing the correlation matrix and removing highly correlated features to reach a final count of 16 features.

The reason we didn't select the original 5 features (open, low, high, close, volume) was mainly due to the fact that these values are well within the margin of error of the target variable (the closing price will not change even 0.1% in a minute), and hence the model could get away with copying these values to reduce the error instead of actually predicting the price. Hence, the final model was trained on 16 technical indicators only.

4.5. Train-Test Split

For the train-test split, we divided the dataset into 75% training set and 25% testing set. Since it's a Time-Series problem, so we couldn't apply cross-validation. Stock market prediction is done for future values based on past values, but cross-validation works on every subset of data, which gives model data points that it won't have on test time. So, in this case, the training set consists of the first 75% of records whereas the testing set consists of the last 25% records.

5. Model Learning

Now since our dataset is all ready we summarize our machine learning problem. Our training set consists of 16 engineered features which we have derived from our 5 main features and then selected through correlation matrix and our domain knowledge. Our response variable is the closing price of each stock for a particular time interval. Since it's a continuous non-linear value, this becomes a classic example of a Regression problem. We have prepared our dataset for 3 time-intervals separately, 1 minute, 5 minute, and 10 minutes. We'll train our model for all the 5 stocks individually for each time interval. We plan to train our model on 5 different non-linear machine learning algorithms.

For hyperparameter tuning, in each of the applied models, we varied and tried different hyperparameter values and trained and tested our model on the AAPL stocks. After the optimal value was selected we then trained the model on the rest of the stocks.

5.1. Support Vector Regressor

Support Vector Regressor(SVR) is a regression equivalent of the Support Vector Machine(SVM) algorithm. SVR uses the same principles as the SVM for classification, with only a few minor differences.

SVR algorithm is supported by sklearn package in python, SVR(). It consists of 2 hyperparameters for tuning, C and γ. These 2 parameters are used for selecting and tuning the kernel family to be used by the SVR function. Out of these, γ was auto supported by sklearn package, and for C we tried different values of 1, 5, 10, and 100, and got the best results for C = 5.

5.2. Random Forest Regressor

Random Forest Regressor is the regression equivalent of the Random Forest algorithm. Random Forest is an ensemble learning method built on top of Decision Trees that utilizes Bootstrap sampling and Random Vector method, and trains different models parallelly.

Random Forest Regressor algorithm is supported by sklearn package in python, RandomForestRegressor(). It consists of 1 hyperparameter for tuning, n_estimators. This hyperparameter captures the number of weak learners to implement in the algorithm. We tried different values for n_estimators, 25, 50, 100, and 150. We got the best results for n_estimators = 100.

5.3. AdaBoost Regressor

Adaboost Regressor is the regression equivalent of the Adaboost algorithm. Adaboost stands for Adaptive Boosting, is also an ensemble learning method built on top of Decision Trees that trains models sequentially.

AdaBoost Regressor algorithm is supported by sklearn package in python, AdaBoostRegressor(). It also consists of 1 hyperparameter for tuning, n_estimators. This hyperparameter captures the number of weak learners to implement in the algorithm. We tried different values for n_estimators, 25, 50, 100, and 150. We got the best results for n_estimators = 100.

5.4. XGBoost Regressor

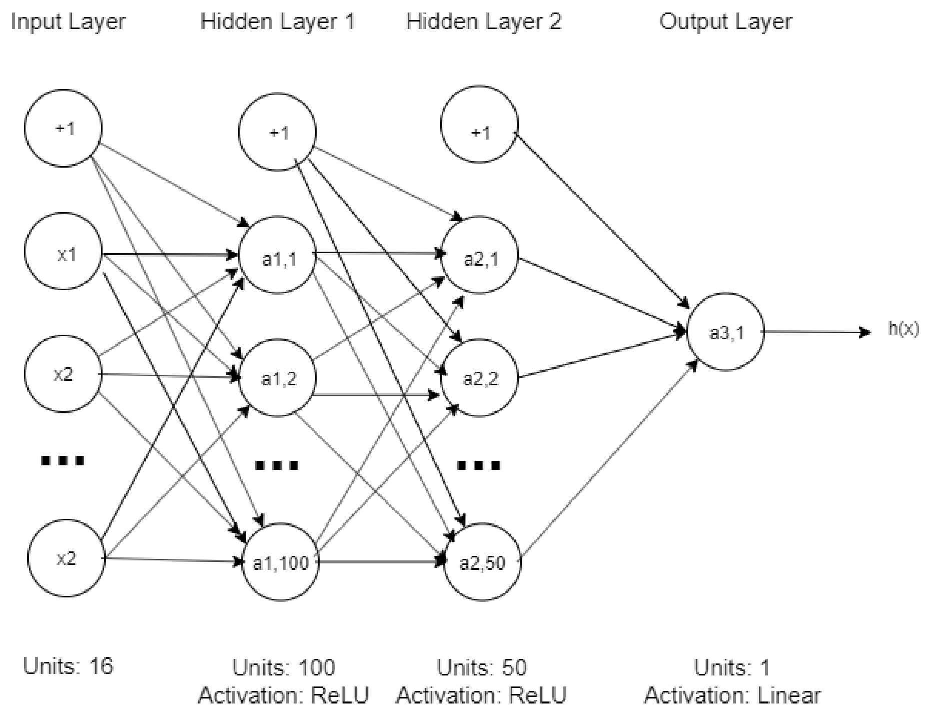
XGBoost Regressor is the regression equivalent of the XGBoost algorithm. XGBoost, stands for eXtreme Gradient Boosting, as the name suggests is a type of gradient Boosting algorithm.

XGBoost Regressor algorithm is supported by sklearn package in python, XGBRegressor(). It also consists of 1 hyperparameter for tuning, n_estimators. This hyperparameter captures the number of weak learners to implement in the algorithm. We tried different values for n_estimators, 25, 50, 100, and 150. We got the best results for n_estimators = 100.

5.5. Artificial Neural Network

We also tried training our model on an Artificial Neural Network (ANN). We choose Feed-Forward neural network for this, as the number of inputs are limited to only 16 features and our response variable deals with real numbers. Our ANN architecture consists of 1 Input Layer (16 nodes), 2 hidden layers (100 nodes and 50 nodes, ReLU activation on both), and 1 output layer (1 node, Linear activation). All the layers are fully connected with each other. Our architecture consisted of a total 6801 parameters. We trained the dataset on our ANN model for 50 epochs.

We selected this architecture by tuning different FNN hyperparameters. We started the hyperparameter tuning by varying the number of hidden layers, we didn't get any better results with more than 2 hidden layers. We also tried different activation functions, sigmoid, tanh, and ReLU, we got the best results with ReLU for hidden layers. For the output layer, we selected the Linear activation function, as the closing price is a continuous value without any bound.



Artificial Neural Network

6. Results and Evaluation

Below are the results of our models on 1-minute, 5-minute as well as 10-minute data. The figures in bold show the best performance among the models for a given stock based on its MAPE values.

6.1. Results

We evaluated our model on 3 different metrics:

Mean Squared Error(MSE): It gives the average of the square of errors of actual and predicted value.
Root Mean Squared Error(RMSE): It gives the square root of the average of the square of errors of the actual and predicted value.
Mean Absolute Percentage Error(MAPE): It is a measure of prediction accuracy of a forecasting method in statistics.

1 minute:
Here, we see that the results, though good for some cases, are wavering. Due to the high volatility of the target variable (closing price), the models find it difficult to predict for every single point with consistency.

Stocks	SVR			Random Forest			AdaBoost			XGBoost			ANN		
	RMSE	MSE	MAPE	RMSE	MSE	MAPE	RMSE	MSE	MAPE	RMSE	MSE	MAPE	RMSE	MSE	MAPE
AAPL	**0.40**	**0.16**	**0.17**	1.17	1.39	0.31	2.27	5.18	0.75	1.25	1.57	0.34	3.81	0.22	1.62
AMZN	**10.39**	**108.1**	**0.52**	78.61	6180	3.89	101.4	10294	5.30	79.35	6296	3.93	63.95	1585	3.26
MSFT	**0.30**	**0.09**	**0.22**	2.00	4.01	1.54	2.49	6.22	2.05	2.09	4.37	1.63	1.88	0.91	1.5
TSLA	1.24	1.55	0.31	**1.70**	**2.90**	**0.14**	3.49	12.21	0.66	1.83	3.35	0.17	42.02	1.26	10.73
WMT	0.37	0.14	0.35	**0.04**	**0.001**	**0.02**	0.70	0.50	0.70	0.06	0.004	0.05	1.67	0.008	1.57

5 minute:
Here, the results are more regular. XGBoost performs the best in general, while Random Forest works good for AMZN.

Stocks	SVR			Random Forest			AdaBoost			XGBoost			ANN		
	RMSE	MSE	MAPE	RMSE	MSE	MAPE	RMSE	MSE	MAPE	RMSE	MSE	MAPE	RMSE	MSE	MAPE
AAPL	1.32	1.74	0.52	1.75	3.06	0.46	2.20	4.84	0.74	**1.72**	**2.96**	**0.45**	36.4	0.75	16.05
AMZN	10.93	119	0.5	**6.74**	**45.5**	**0.25**	16.38	268	0.78	6.98	48.7	0.27	164.3	32.9	8.12
MSFT	0.58	0.34	0.41	0.27	0.07	0.15	0.58	0.34	0.43	**0.18**	**0.03**	**0.12**	5.4	0.07	4.07
TSLA	1.42	2.01	0.33	1.01	1.02	0.17	2.79	7.82	0.68	**0.79**	**0.63**	**0.16**	50.06	0.89	12.47
WMT	0.29	0.08	0.23	0.17	0.02	0.09	0.45	0.21	0.37	**0.12**	**0.01**	**0.09**	5.98	0.01	4.94

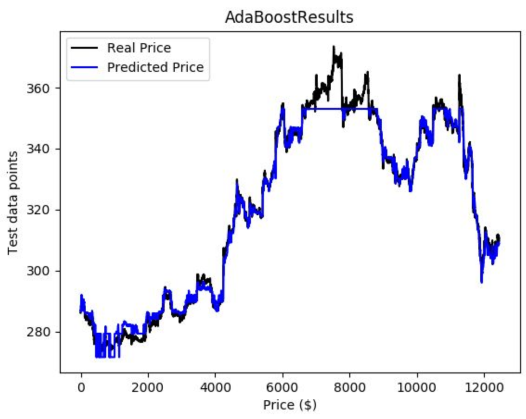
10 minute:
Even though the results here are not the best as compared to results from 1-minute data, they are the most consistent here. Also, with more stability in the target variable, XGBoost delivers the best metrics across the entire selection of stocks.

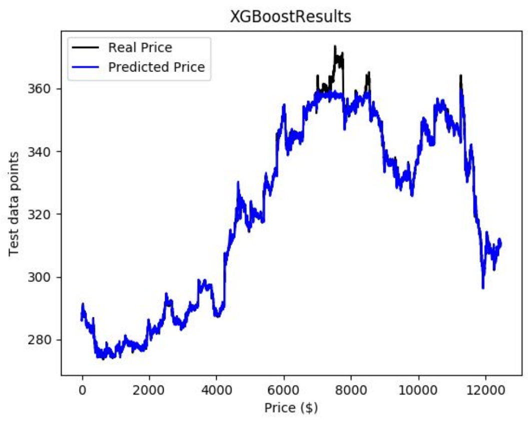
Stocks	SVR			Random Forest			AdaBoost			XGBoost			ANN		
	RMSE	MSE	MAPE	RMSE	MSE	MAPE	RMSE	MSE	MAPE	RMSE	MSE	MAPE	RMSE	MSE	MAPE
AAPL	1.67	2.81	0.67	2.28	5.20	0.66	2.42	5.90	0.82	**2.12**	**4.51**	**0.59**	36.18	1.88	15.89
AMZN	13.47	181	0.64	9.69	93.9	0.38	16.5	274	0.75	**8.22**	**67.5**	**0.32**	168	60.8	8.3
MSFT	0.73	0.54	0.55	0.36	0.13	0.21	0.64	0.41	0.45	**0.29**	**0.08**	**0.18**	5.43	0.12	4.07
TSLA	1.65	2.74	0.37	1.32	1.76	0.25	3.74	14.0	0.91	**0.94**	**0.89**	**0.20**	51.37	6.46	12.9
WMT	0.38	0.15	0.29	0.23	0.05	0.15	0.54	0.29	0.45	**0.17**	**0.03**	**0.13**	6.00	0.03	4.96

6.2. Model Evaluation

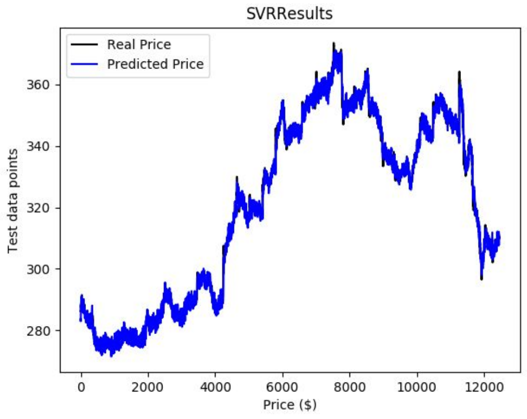
We now give a visual evaluation of the models and see why a given model performed the way it did.

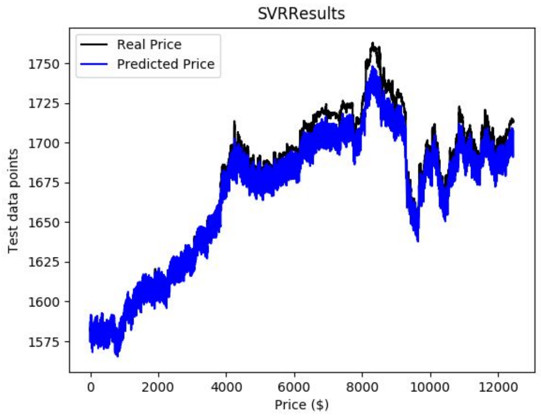
ANNs perform the best for the MSE metric (the metric used as loss for training). Also, it has a high tendency to converge at a local minimum, so it does not give the best results and is not as reproducible.



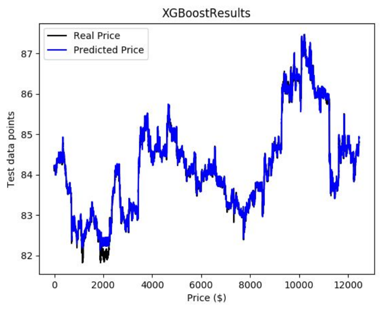


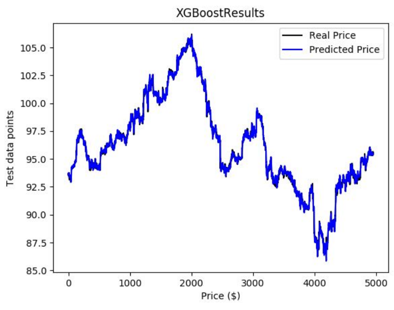
XGBoost performs the best among the three ensemble models, while AdaBoost performs the worst. We observe that in general, tree-based ensembles don't work well. Since stock prices are generally increasing and testing data came chronologically after training data, tree-based predictions quickly lead to highest price possible, but cannot go further than that (due to averaging at the leaf node of the tree in regression). Because of this, we trained XGBoost for linear as well as tree-based models, and received the best results on linear models, validating our theory. The figures above show sample results for XGBoost and AdaBoost on TSLA stock.

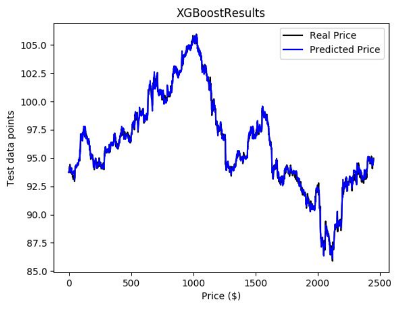




An interesting revelation is that SVR performs second best after XGBoost, above other ensemble models. This validates the powerful math behind the model, even though in theory they are much less interpretable and in practice much slower than the other models. Though for the 1-minute data, its predictions are highly volatile, even more so than the data itself. The examples given above are for TSLA and AMZN stock.







Interval wise, 1-min interval is the most volatile, so doesn't give as consistent results as 10-min data. The examples here are for XGBoost predictions on WMT stock on data from all 3 periods.

7. Conclusion

From the results and evaluation section, we can observe that our machine learning models did perform quite well and the predicted closing price fitted well on the actual closing price on the test data. SVR and XGBoost generally perform the best owing to its mathematical prowess and its resistance to overfitting. Tree-based ensemble models are usually not the best choice for regression if the model is used on data where the target is greater than the data it is trained on. ANN converges on its loss really well, but better tuning would be required to achieve consistent results across different metrics.

All in all, this suggests that machine learning models can be used to solve tasks that seem mathematically highly complex, and interesting insights can be gained from them.

References

1. Guo, Yanhui, et al. "An adaptive SVR for high-frequency stock price forecasting." IEEE Access 6 (2018): 11397-11404.
2. Patel, Jigar, et al. "Predicting stock market index using fusion of machine learning techniques." Expert Systems with Applications 42.4 (2015): 2162-2172.
3. Patel, Jigar, et al. "Predicting stock and stock price index movement using trend deterministic data preparation and machine learning techniques." Expert Systems with Applications 42.1 (2015): 259-268.
4. https://school.stockcharts.com/doku.php?id=technical_indicators
5. https://www.investopedia.com

Appendix

Github link for project: https://github.com/vidheyoza/IntradayStock