Machine Learning Engineer Nanodegree

Stock Price Predictor

Sandeep Paulraj
January 9th, 2018

I. Definition

Project Overview

Investing in stocks has been a way to grow wealth for a very long time. This has been done for well over a century and as more and more countries/societies are developing, each country has its own stock markets where securities are listed and investors can invest their money. The fundamental reason why people invest in stocks is to participate in a company's growth. If the company grows and makes money, the stock price will rise. The investor might then get regular dividends and can also sell his/her shares and make a profit if the selling price is greater than the investment. Afcourse, it is also possible to lose money.

We live in an interconencted world with an over abundance of data and we can see a large numbers of fields and industries where machine learning is being used to make informed decisions. The stock market is also one such field/entity where there is a plethora of data and this data can be used to make buy/sell decisions. Machine learning algorithms are already being used by investment banks and hedge funds and these are getting more and more complex.

In this project I will leverage publicly available historical stock price data from yahoo finance in my models.

Some papers which discuss machine learning techniques are sited below.

Predicting Long term price Movement

Machine Learning for predicting Bond Prices

Problem Statement

In this project, I will attempt to predict the next trading day's adjusted closing stock price for Broadcom. This is a regression problem and NOT a classification problem. Broadcom is semiconductor company that is a major Apple supplier and has been very aggresively buying other companies to grow its portfolio. It is in the process of orchestarting a takeover of Qualcomm as well. I will attempt to use Broadcom stock data from yahoo finance in my model.I will also attempt to gauge if Apple stock has an effect on the Broadcom stock price. Also, i will try to guage if the VanEck Vectors Semiconductor ETF has an effect on Broadcom.

Initially while going through the capstone review process, I was attempting to predict the following seven trading day's Adjusted Closing stock price. However, capstone review comments suggested that I should only try to predict the following trading day's Adjusted closing stock price. On thinking about this a little more, we have to be cognizant of the fact that there are several geopolitical uncertainties that exist in the market. Let us take an example: Let us say the US amrkets end a day strong; Japan and Asia start of strong but towards the end of the Asian trading day, there is some bad news on economic indictors or say there was a data breach. Asia may end up weak and this will in all likelihood have an impact on European markets which then digest this information and actually have a bad day. Europe has a bad trading day and markets end down. This sequence of events, results in the US markets starting of week. Thus, it is difficult to make informed judgments of stock prices seven trading days ahead. To think of it, if we are on a Friday evening trying to gauge the following seven trading days' closing adjusted stock price; that is trying to go as far ahead as the next to next Tuesday, we have two weekends in between where a lot of events might happen. Based on this and my first capstone review, it was decided to predict only the next trading day's adjusted closing stock price.

Why do i want to use Apple and SMH?

- I decided to use Apple Data since Broadcom is a major Apple supplier
- I decided to use SMH since it is a semiconductor ETF and broadcom is a semiconductor company.

I will use various regression techniques in my various models and settle for the model that performs the best. Now is a good time to note that we are dealing with Time Series of data and will need special consideration. We cannot shuffle the data in any of our analysis. This actually is a good deviation from what i have learned in the Nanodegree since we didn't exactly learn how to deal with Time Series data. Different types of regressors will be required. In the end, it is anticipated that I will end up learning new models and techniques. Since we are dealing with a regression problem I intend to try out regressors such as RandomForestRegressor, DecisionTreeRegressor, Support Vector Regressor, SGDRegressor and even KNeighborsRegressor.

The strategy I will employ will be to setup a dataframe of Broadcom stock price. Initially atleast, I will use all the data provided as my features. My prediction data will actually be the next trading days' adjusted closing stock price. This will need some amount of data manipulation since. I explain this in the section below. I will run various regression models to predict my "next trading days' adjusted closing stock price". My benchmark model will be a simple linear regression model and the final model should be better then this simple linear regression. This is however not guaranteed as sometimes simple models also provide good results.

Metrics

I will be leveraging sklearn in this project. From sklearn metrics we will have access to an array of metrics from our model. The main evaluation metric I will be using is the root mean squared error. The root mean squared is simple to calculate and can be calulated as shown below. Based on previous projects and experience, i don't think a metric exists to calculate this for us. This has to be derived and is simple to derive as can be seen below.

```
from sklearn.metrics import mean_squared_error
from math import sqrt

rms = sqrt(mean_squared_error(y_actual, y_predicted))
```

The reason for using RMSE is becuase it will give us the difference between the stock's actual and predicted adjusted closing stock price.

The mean squared errors(MSE) will give us the sqaure of the difference between the stock's actual and predicted adjusted closing stock price. Thus we will have a positive value even when the differnce is negative. If we don't use a squared value then positive and negative values will cancel each other, i.e. underestimates and overestimates will cancel each other. The reasoning behing taking the square root using **sqrt** function is to get an error value in terms of stock price and not the squared value of the stock price.

II. Analysis

Data Exploration

All the analysis and code cna be seen at

`stock_price_estimator.ipynb`

It is possible to obtain stock price financial data from various sources. It is also possible to use python api and the yahoo finance library to obtain this data. For some reason, I am having trouble installing the yahoo finance library in python 3.6. So I have decided, to obtain the csv data from the yahoo finance website and read in the dataframe using pandas.

I will obtain Broadcom stock, Apple stock and SMH ETF price data from yahoo and the source is below.

Broadcom(AVGO) stock price

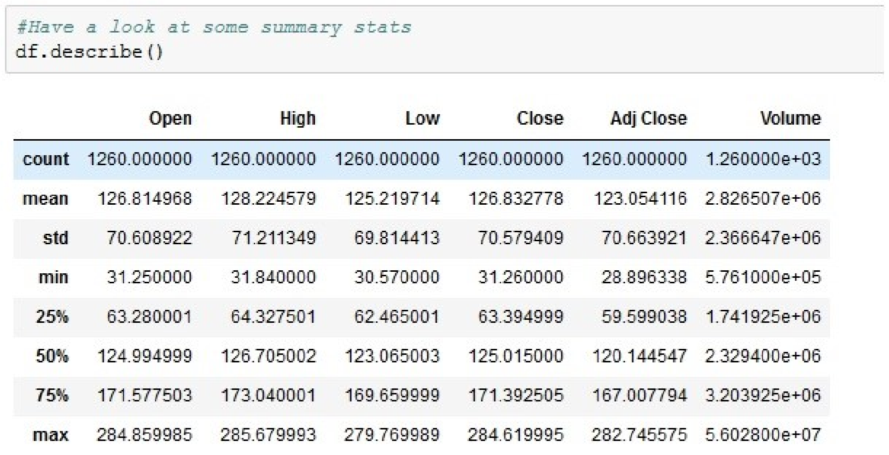
Apple(AAPL) stock price

VanEck Vectors Semiconductor ETF(SMH)

Clicking on the above links will also show the data I will be leveraging. I initially intended to use 1 years worth of data but it became obvious that this was nearly not enough data. Infact, I would argue that even 5 years worth of dats is not enough. However, my hands are tied here; I can't go back any further since Broadcom(AVGO) has changed massively over the last several years acquiring companies to grow its EPS. Beyond 5 years, it was literally a totally different company. In a way that makes this analysis a little unique in my opinion. We have to make do with 5 years worth of historical data.

The individual pieces of information that I will leverage for each day will be "Open", "High", "Low", "Close", "Volume" and "Adjusted Close". 1 year worth of data will provide approximately 250 data points. 5 years worth of data will result in approximately 1250 data points. The data range of the dataset will be between January 07, 2013 to January 05 2018.

After obtaining the data and converting to a dataframe, I decided to display the summary statistics. This is depicted in the ipython notebook but I display the summary statistics below as well.



```
#Have a look at some summary stats
df.describe()
```

	Open	High	Low	Close	Adj Close	Volume
count	1260.000000	1260.000000	1260.000000	1260.000000	1260.000000	1.260000e+03
mean	126.814968	128.224579	125.219714	126.832778	123.054116	2.826507e+06
std	70.608922	71.211349	69.814413	70.579409	70.663921	2.366647e+06
min	31.250000	31.840000	30.570000	31.260000	28.896338	5.761000e+05
25%	63.280001	64.327501	62.465001	63.394999	59.599038	1.741925e+06
50%	124.994999	126.705002	123.065003	125.015000	120.144547	2.329400e+06
75%	171.577503	173.040001	169.659999	171.392505	167.007794	3.203925e+06
max	284.859985	285.679993	279.769989	284.619995	282.745575	5.602800e+07

Figure 1: alt text

What I will attempt to predict is the adjusted stock price 1 day ahead.

We are dealing with time series data and data has to be handled chronologically. Also the stock price data is continuous is nature. So essentially we will have 5 years worth of data to predict the next trading day's closing stock price.

For the initial exploratory analysis, I read in the data and realize that the dates increase, i.e in the csv file and data frame February 1 will come before February

2. I reverse the order of the rows of data. I would like to explain the reversal of order: This is done since while visualizing my data, I prefer to see the dataframe with the more recent date on top. I depict this in my data frame that can be seen in the ipython notebook.Another reason for doing this is becuase when we plot some graphs the most recent dates will be on the right.

In the associated ipython notebook I also have another column where I store the difference between the highest and lowest daily stock price. This may prove to be useful in my analysis as this provides a daily trading range.

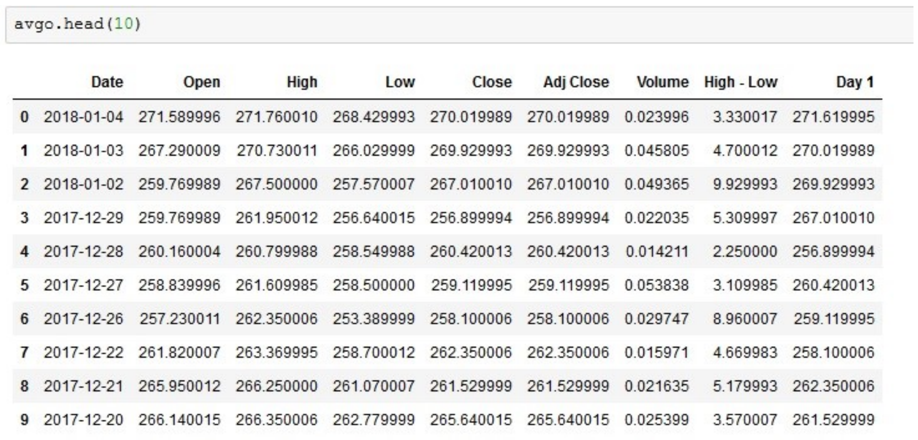
It is important to use standard pandas routines to set up the dataframe. This essentially will result in a more elegant and cleaner final solution. Please take a look at the accompanying notebook to look at all the exploratory analysis.

The following are two good sources of how to avoid lookahead bias. I am adding those links here.

9 Mistakes Quants Make

Avoiding Look Ahead Bias in Time Series Modelling

Below, I depict the dataframe. This has the Stock Volume after having been preprocessed, the Stock "High-Low" price and also the project prediction column that is called Day1 prices.



```
avgo.head(10)
```

	Date	Open	High	Low	Close	Adj Close	Volume	High - Low	Day 1
0	2018-01-04	271.589996	271.760010	268.429993	270.019989	270.019989	0.023996	3.330017	271.619995
1	2018-01-03	267.290009	270.730011	266.029999	269.929993	269.929993	0.045805	4.700012	270.019989
2	2018-01-02	259.769989	267.500000	257.570007	267.010010	267.010010	0.049365	9.929993	269.929993
3	2017-12-29	259.769989	261.950012	256.640015	256.899994	256.899994	0.022035	5.309997	267.010010
4	2017-12-28	260.160004	260.799988	258.549988	260.420013	260.420013	0.014211	2.250000	256.899994
5	2017-12-27	258.839996	261.609985	258.500000	259.119995	259.119995	0.053838	3.109985	260.420013
6	2017-12-26	257.230011	262.350006	253.389999	258.100006	258.100006	0.029747	8.960007	259.119995
7	2017-12-22	261.820007	263.369995	258.700012	262.350006	262.350006	0.015971	4.669983	258.100006
8	2017-12-21	265.950012	266.250000	261.070007	261.529999	261.529999	0.021635	5.179993	262.350006
9	2017-12-20	266.140015	266.350006	262.779999	265.640015	265.640015	0.025399	3.570007	261.529999

Figure 2: alt text

Exploratory Visualization

In the ipython notebook I have 4 different plots related to Broadcom stock price data. Since the stock has been in the news lately for its takeover attempt of Broadcom and also due to continually beating result estimates, I decided to plot daily volume for the last 5 years. Clearly we can see spikes associated with important news and results.

In another plot, I also depict the intraday "High - Low" stock price and this too has shown a lot of unusual spikes lately due to a variety of reasons that I mention below.

- A news conference with President Trump to announce moving back headquarters to the United States.
- Closing the Acquisition of Brocade
- Announcing the Acquisition of Qualcomm.

In general, we can see that these have resulted in pushing up the stock price higher. However, this also gives a clue that it may be prudent not to use all the features of the input dataset. For example, we need to answer and be cognizant of the following

- If we take volume into consideration, do we need High stock price and Low stock price
- Do we need the delta(High - Low) Stock price in our analysis
- Do we need to augment Broadcom data with that of Apple and SMH

My analysis will try to answers these questions.

Below is a plot of the Opening Stock Price of Broadcom(AVGO) over the last 5 years.

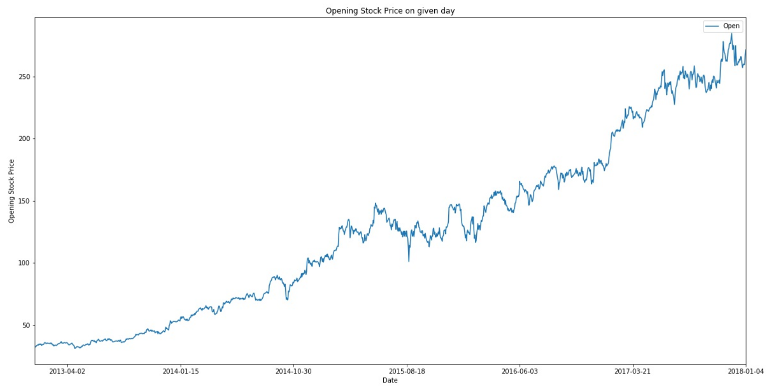


Figure 3: alt text

Below is a plot of the Adjusted Closing Stock Price of Broadcom(AVGO) over the last 5 years.

Below is a plot of the Normalized Trading Volume of Broadcom(AVGO) over the last 5 years.

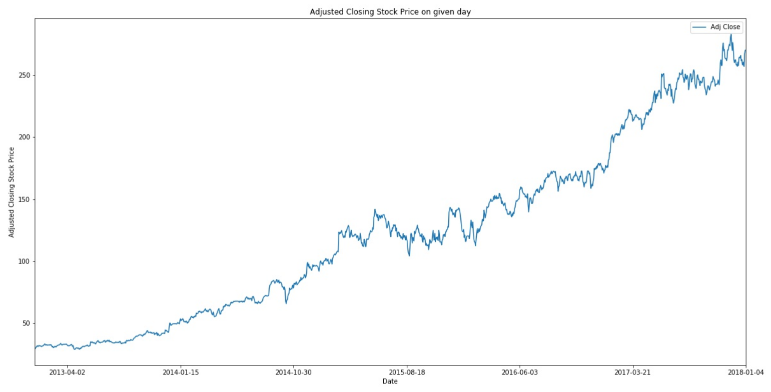


Figure 4: alt text

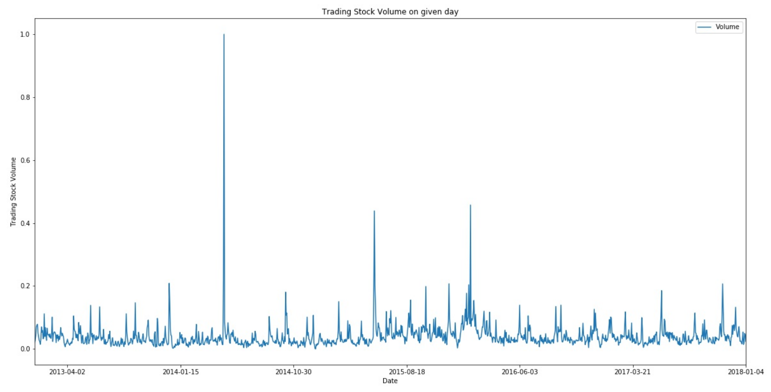


Figure 5: alt text

Below is a plot of the Difference between Daily High and Low Stock Price of Broadcom(AVGO) over the last 5 years.

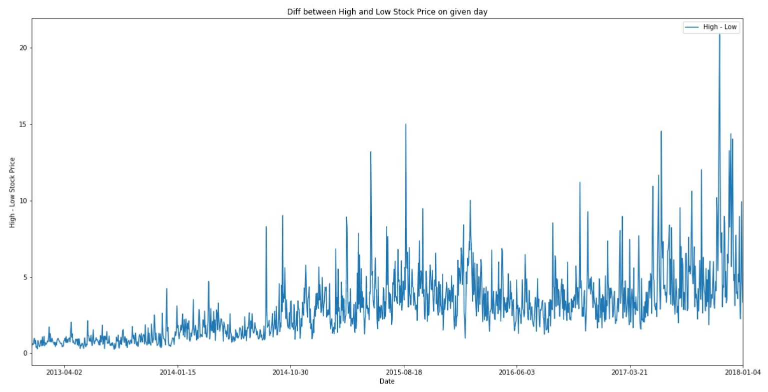


Figure 6: alt text

Below is a combined plot of the above four plots. Please notice that there is usually a “stock price movement” with volatility and volume. In some respect the difference between High and Low Stock Price is a measure of volatility.

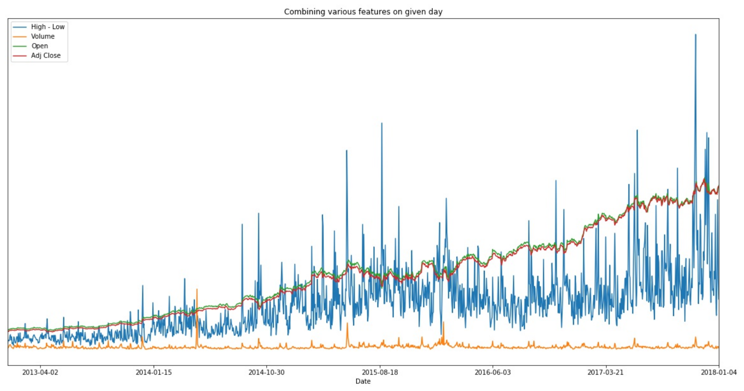


Figure 7: alt text

Algorithms and Techniques

Since we are dealing with time series data we have to deal with the data in a chronological manner. We also need to do cross validation. The links provide a good reference for time series cross validation.

Cross-validation for time series

Pythonic Cross Validation on Time Series

Using k-fold cross-validation for time-series model selection

sklearn time series split

sklearn: User defined cross validation for time series data

As mentioned previously in this writeup, we are fundamentally dealing with a regression problem and not a classification problem. Hence this project is unique in that I have not had to deal with the time series regression before so it is a good learning experience.

I will try Linear Regresssion and it is possible for a simple model to provide good results. However, I intend to try other regressors such as SVR(Support Vector Regression), Decision Tree Regressor and Random Forest. Random Forest is a time series algorithm implemented in time series forecasting. Please see citations below in the vartious sub sections

I will be trying the various regressors mentioned above along with GridSearchCV. It is at this stage that i will setup a dictionary of parameters from which to try out various options.

My prediction data will be the **next trading day's adjusted closing stock price**. Initially atleast I intend to use all the featuers present in the data set downloaded from yahoo finance.

Decision Tree and Random Forest

Decision Tree Regressor

A decsion tree is a type of directed acylic graph. One of the reaons i decided to try the **DecisionTreeRegressor** is because they are fairly robust on small datasets and we have a small number of data points in our study. Decision trees are easy to understand and easy to interpret. They are also much faster to train. They perform well with a mix of both categorical and numerical data. With decision trees you don't really need to be concerned about normalizing/scaling the data.

Decision Tree Regressor

Random Forest Regressor

Random Forests are an extension to decision trees. The basic premise of random forests are that they combine decision trees with bootstrap aggregation making them more robust and less prone to overfitting compared to a decision tree. Bootstraping is a powerful method to reduce model variances. At each split point, decision trees split based on one feature variable, whereas random forest split based on multiple feature variables. Random Forests essentially are an ensemble of decision trees. Random Forests are more computationally expensive when compared to decision trees.

Stock Price Prediction using Random Forest

Support Vector Regressor

I initially tried Support Vector Regression but dropped it for later analysis as the initial results itself were not good. Even after preprocessing the volume data with MinMaxScaler, I did not obtain good results. Support Vector Regressors uses the same basic idea as Support Vector Machine (SVM). SVM happens to be a classification algorithm. We are dealing with a regression problem and thus SVR predicts real values rather than a class. Support Vector Regressors acknowledges the presence of non-linearity in the data and provides a proficient prediction model.

Support Vector Regressor

Benchmark

As suggested in the capstone proposal review, I will train and test an out-of-the-box linear regression model on the project data. My final solution should outperform this linear regression model. By using a linear regression model as the benchmark, and training/testing it on exactly the same data as my final solution I will be able to make a clear, objective comparison between the two.

Using the RMSE metric I found out that that for the Benchmark Linear model the RMSE was 1.79. This can also be seen in the ipython notebook.

Root Mean squared error: 1.79

III. Methodology

Data Preprocessing

After obtaining my data I intend to follow these steps.

- Reverse the rows since as an example February 1 comes before February 2 in the downloaded data. I want it to be the other way round for analysis purposes. To clarify again, the reversal of the rows is just a matter of convenience while depicting the data. I prefer to see recent dates on the top.

- Append a column with trading price range (High - Close)
- Add the following day's stock price. This will be what we are trying to predict.
- Depending on results of various models/scenarios I may and in all likelihood have to append data pertaining to Apple and SMH to the Broadcom data.

I have to mention that the data obtainted from yahoo was clean and did not contain any "NaN". In a way this is on expected lines. The ticker symbols I am dealing with are well known and the various features such as "Open", "Close", "Volume" have no reason to be missing.

As can be seen in the notebook, there is some amount of work involved to obtain the next trading day's adjusted closing stock price and concat this to the Broadcom data. However, there are well known routines such as **concat** to aid in this. Infact **concat** is used agin to append Apple and SMH closing adjusted stock price to the Broadcom data.

Since the stock volume data has a very high unit, i thought it would be necessary to do some preprocessing on this. Thus, i have performed preprocessing using **MinMaxScaler**. The relevant is also shown below.

```
scaler = MinMaxScaler()

avgo[['Volume']] = scaler.fit_transform(avgo[['Volume']])
```

Implementation

So the first thing to attempt is to successfully use the various regressors such random forest regressor, decision tree regressors and svr. I did not use time series cross validation initially. This is becuase I had never previously used these regressors and i did not know how to integrate the time series cross validator.

Immediatley with the initial data without appending Apple and SMH data, I could see that among the various regressors the Random Forest Regressor was giving the best results by checking the RMSE metric.

The next stage of the project involved using the time series cross validator and integrating this into the project flow.

```
tscv = TimeSeriesSplit(n_splits = 10)
tscv.split(X_train,y_train)
```

I use the sklearn **TimeSeriesSplit** with a splits parameter of 10 and this becomes my time series cross validator, tscv. My next step is to choose a regressor. I chose the **DecisionTreeRegressor** and the **RandomForestRegressor**. I did not use SVR since in the previous step I did not get good results with SVR.

Also I began to notice that training was taking a very long time with SVR so i dropped it for further analysis. I used **GridSearchCV** with the 2 mentioned regressors and I see an improvement in the results for Random Forest Regressor. This also happend to be the first time I have successfully integrated time series cross validation, regression and grid search cv in my project workflow.

The RandomForestRegressor gave good results with an RMSE of **1.01**

This means that I am able to predict the next day's adjusted closing stock price to the dollar which is good. In the ensuing sections I will attempt to make this even better.

Root Mean squared error: 1.01

The general flow for all the regressors and trials will be as follows and is also labelled in the notebook.

```
- Do Time Series Split
tscv = TimeSeriesSplit(n_splits = 10)
tscv.split(X_train,y_train)

- Create a regressor object, it can be random forest or decision tree or K neighbours.
regr = RandomForestRegressor()

- Setup a dictionary of params to search
parameters = {'min_samples_leaf':range(1,10)}

- Peform Grid Search
grid = GridSearchCV(regr, parameters, cv=tscv)

- Fit The Data
grid = grid.fit(X_train,y_train)

- Find Optimal Model
grid.best_estimator_
```

Refinement

Now, I proceed to add Apple and SMH data to my feature set. I do this in the following way mentioned below.

```
avgo_enhanced = pd.concat([avgo,aapl,smh], axis = 1, join='inner')
```

I tried various combinations by adding/removing features from the training data. After much experimentation I found that the best features to leverage to predict the next trading day's adjusted closing stock price were **'Open','Volume','Adj Close' and 'High'**

I also have training data called **X_with_aapl_smh** with the features **'Open','Volume','Adj Close', 'High', 'Low', 'SMH Adj Close' and 'AAPL Adj Close'**.

```
X = avgo_enhanced[['Open','Volume','Adj Close', 'High']].as_matrix()
X_with_aapl_smh = avgo_enhanced[['Open','Volume','Adj Close', 'High', 'Low', 'SMH Adj Close'
```

My prediction values are below.

```
y = avgo_enhanced['Day 1'].values
```

At this stage I proceed to try and find an optimal solution. I try out various regressors. I had earlier dropped SVR and at this stage I decided to try **DecisionTreeRegressor** and **RandomForestRegressor**. I decided to stick with the random forest and decision tree regressor.

The **DecisionTreeRegressor** regressor gave me an RMSE of **4.68**.

The **RandomForestRegressor** regressor gave me an initial RMSE of **0.97**

I then decided to modify some of the parameters of the RandomForestRegressor to gauge if this could improve my model.

My model is as follows.

```
def rfr(X, y):
    #Do Time Series Split
    tscv = TimeSeriesSplit(n_splits = 10)
    tscv.split(X_train,y_train)

    #Create a Random Forest regressor object
    regr = RandomForestRegressor(n_estimators = 20)

    parameters = {'min_samples_leaf':range(1,10)}

    #Peform Grid Search
    grid = GridSearchCV(regr, parameters, cv=tscv)

    #Fit The Data
    grid = grid.fit(X_train,y_train)

    #Find Optimal Model
    return grid.best_estimator_
```

With this my RMSE reduced to **0.61**

I also tried the same model with the training data **X_with_aapl_smh** that I mentioned above.

The results were worse with an RMSE of **3.11**. This clearly shows that we don't need to use Apple and SMH in our models.

IV. Results

Model Evaluation and Validation

If we were predicting an entire index then maybe we could have leveraged decades worth of data. This is fine if we are trying to predict an index such as Dow Jones, FTSE 100, German DAX. However, with Broadcom stock I decided to make do with 5 years worth of data. I mentioned previously in this writeup as to why I chose to do so. The original company Avago(AVGO) has been buying up companies to grow their EPS and finally bought Broadcom and took it company name but kept the stock ticker as AVGO. Before 5 years it was a totally different company.

While experimenting with the data, it became clear that the time series cross validators need data and thus is decided to keep 1248 points for training and validation. Thus I had a remaining 10 points for testing my model. This is low but I feel this is a compromise since the training/validation phase needs a lot of data. In some respect apart from doing fundamental company research if we use this model to predict the next day's closing adjusted stock price we should in theory have to use all available data upto the curent trading day.

Here are some results.

The Benchmark Linear Regression Model had an RMSE of **1.79**

A simple Decision Tree Regressor without time series split had an RMSE of **7.43**

A simple Random Forest Regressor without time series split had an RMSE of **1.86**

Using Time Series Split and Grid Search CV the Decision Tree Regressor had an RMSE of **2.49**

Using Time Series Split and Grid Search CV the Random Forest Regressor had an RMSE of **1.01**

After this in my final updates, i added Apple and SMH data.

With Apple and SMH data using Random Forest Regressor the RMSE was **3.11**

Without Apple and SMH data and dropping the "Low" stock price of the day the RMSE was **0.61**

This happens to be my final model.

The features that I used in this are specific only to Broadcom stock itself and are **'Open','Volume','Adj Close' and 'High'**

An RMSE of 0.61 is good and means that we can predict with 60 cents of the next day's stock price. This is an improvement on the Benchmark model.

Justification

As mentioned above the Benchmark model had an RMSE of **1.79**

The final model based of a random forest regressor has an RMSE of **0.61**

This is an improvement.Hence, the final results are found stonger than the initial benchmark model results.

We cannot shuffle the data since we are dealing with time series data which have a chronological dependence.

The results found from the model can be trusted because for all but the first data point we have the original **next trading day's closing adjusted stock price**. We can easily compare the actual and predicted. However, i would have liked to predict within a few cents instead of around 60 cents. It is entirely possible that this model is susceptible to noise and perturbations. Let me explain this; the last 20 odd training days have been interesting for Broadcom, it is involved in a proxy fight with Qualcomm, then there are concerns over Iphone shipments, then there seems to be a sector rotation in stocks from Tech to Financials so the model may be susceptible to events.

My final model is better than the benchmark. However, we can see that though we get an improvement it is not within cents. High frequency traders these days are able to leverage algorithms and detect minute market inefficiencies and make cents on a stock. So in that respect, i expect there to exist several other propriety models that will perform better.

V. Conclusion

Free-Form Visualization

We can see the actual vs predicted stock price. I would have loved for the model to be more accurate but we can also see that predictions are inline with actual adjusted closing stock price. They are not exactly way off.

Given that I am not attempting to use any other stock's data to predict the next day's closing adjusted stock price of Broadcom, it is conceivable that we can use the same featuers and use it for all other stocks. In this way we can perhaps do a stock screen. This however would take up a lot of time and processing power. I started off, thinking that using SMH and Apple Data will enhance the Broadcom data and perhaps give a better solution. This was however found not to be the case. This is surprising; however this is good in anothr way. This probably goes onto show that a very large component of my prediction is "baked" into the various features of the stock itself. Lets us take 2 examples

- Let us say there is a good report about Apple selling more iphones than predicted. This will lead to increased volume/incresed stock price of apple.

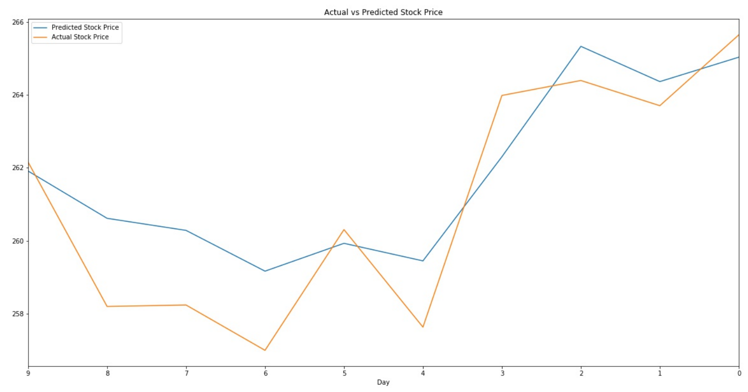


Figure 8: alt text

Based on market dynamics, investors trying to diversify may choose to buy Broadcom since it supplies various companies. So Broadcom will aslo see buying interset and increased volume may lead to increased stock price. Thus there is no need to depend on Apple Data.

- Let us say there is a "sector rotation" and investors want to move out of semiconductors stock. The whole sector maybe down. Negative sentiment from the sector will translate to selling pressure on semiconductor stocks including Broadcom. Thus there may be a case to not need SMH data for Broadcom analysis.

These 2 show that it is possible to make predictions on a certain stock by simple concentrating on data pertaining to that stock itself. This is clearly shown by the RMSE which did not improve when I appended Broadcom data with Apple and SMH.

Reflection

The most important aspect of this project was that I was essentially dealing with time series of data. This type of problem is something that I have not dealt with previously. This means that we cannot do any random shuffling of the data since the ensuing data point depends on the order of the previous data points. The order had to be maintained. I had to figure out how to use the sklearn TimeSeriesSplit which is a time series cross validator. The example provided in the sklearn documentation is helpful but rudimentary. To have this succesfully inegrated into my models was something that I had to spend a lot of time on. Afcourse, once it is integrated into the "code flow", it is easy to use this in all the various different models.

The other important aspect of this project was using various different types of regressors. The project was essentially a regression problem and not a classification problem. I had to understand and use regressors that I was not previously exposed to. It took me a finite amount of time to use Grid Serach CV along with these new regressors, time series split and a dictionary of parmaters to try in my models. In the end , I ended up learning several new techniques and models. In hindsight, choosing stock price predition as my capstone was a good decision since I ended up learning several new things.

The final model and solution does fit my expectation to some extent. I would have liked to be able to make even better predictions. With the improvements I mention below it can be used in a general setting to solve these types of problems.

Improvement

I would have liked to do the following in my project.

- Allow the user the flexibility to select a certain stock and do the analysis presented in the project notebook on the stock. There is one immediate concern that comes to my mind. Since Broadcom is a semiconductor company and is a big Apple supplier, i decided to use Apple stock and the SMH ETF data for my analysis. On further analysis these were found not to be necessary. However for Southwest Airlines, Oil industry data might be useful. I will also need to code up more general user defined functions that can take a stock symbol and dataframe and perform all the necessary analysis.
- Allow the user to select the date range. python apis can be leveragred to pull in data for only these date ranges. Please be aware that a sufficient amount of data will be required for training purposes.
- I would loved to have used neural networks and try out LTSM. The main reason for not doing so is becuase I am not yet fully aware of LTSM and i didn't want to use something that i didn't have a good grasp of. If I had a good grasp of LTSM, i would defintely have tried using it in my project. LTSM should give good results for time series data. I intend to try this out at a later date as time permits.
- Machine learning for stock trading and prediction is getting more and more complicated and fancy. I see sentiment analysis using tweet data also being used. I think a better solution will defintely exist if we set up our analysis using more enhanced data points with data from tweets and using perhaps LTSM. Thus if my final solution were used as the new benchmark, I do believe an even better solution exists.
