Machine Learning Capstone Project

Project Overview

Using Financial Models from the past to make future investment profitable returns is a common trend. Investment firms, Hedge funds and individuals use this model all the time to determine when and how to move investments so profits can be maximized and losses can be minimized.

Thousands and thousands of companies are traded by stock in stock markets around the world with hedge funds, investment firms , mutual funds and individuals.

There are many different factors that affect the price of the stock and it is very difficult to predict this with a good level of accuracy.

One of the most important decisions in finance is to buy and sell a stock at a certain price. Predicting the price of a stock based on historical data and other industrial factors is a very important challenge for machine learning.

Problem Statement

Given the begin date and end date for stock trading, use this data to predict a stock price in the future of the end date.

Given the how a stock has traded over a period of time, and the information about short trading and the moving average, predict the price of the stock in the future.

Stock prediction is more of a Machine Learning regression problem. The work set data is clearly defined with begin date and end date, and the predictive date is also clearly defined. In this case we will compute additional features that are needed like Short-Ratio and the Moving Average.

We will use Support Vector Regression for machine learning prediction .

Evaluation Metrics

Two primary metrics are used here.

Mean Absolute Error

Mean Absolute Error = (1/N) * Sigma(i from 0 to N) (prediction_value - real_value)

The MAE measures the mean of the difference between the prediction value and the real value. n. The MAE is suitable to describe uniformly distributed errors. This is likely not the case with stock fluctuations. Because model errors are likely to have a normal distribution rather than a uniform distribution, the RMSE is a better metric to present than the MAE for such a type of data

Root Mean Squared Error

This is because RMSE gives relatively high weight to a large error value. This makes it so RMSE is useful when larger error deltas are not desirable on predicted values.

Data Exploration

Financial Historical data for this project are obtained from Yahoo Finance using the Pandas DataReader. Dataset for the begin_date and the end_date for the stock training data are fetched from Yahoo finance and saved as a csv spreadsheet file.

Features obtained for each trade are:

Date of Trading : The date the trade actually took place for this data record
Volume of Trading : The total number of trades done on this date. This indicates how active the trading is for this stock in the stock market.
Price Opened: The opening trading price for this stock on that day.
Price closed: The price this trading on that day closed it at. This is very useful and is used.
High: The highest trading price for that day

Low: The lowest trading price for that day.
Moving Average for the stock is a feature that is calculated based on the Close price.

We also obtain the Stock shorting data from Bats Global Markets, a Chicago Board of Exchange company. The stock shorting data only includes the volume of shorting and we use this with the total volume of trading and compute the *shorts-ratio* based on these.

Data Sample

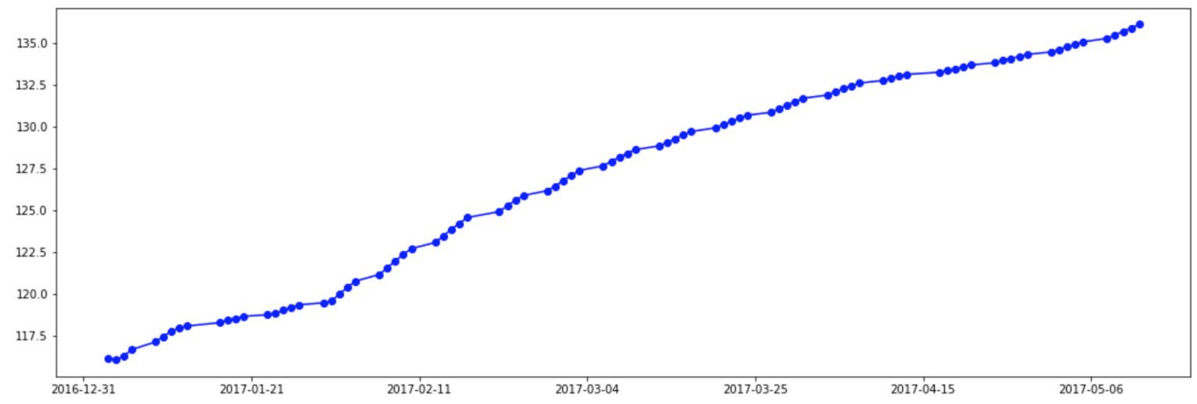
Date	AAPL (Ticker)	Volumes	Open	Close	Short Ratio	Moving Average
2017-01-03	115.173203	28781900	115.800003	116.150002	0.49364349246	116.150002
2017-01-04	115.044292	21118100.0	115.849998	116.019997	0.554171383381	116.0849995
2017-01-05	115.629334	22193600.0	115.919998	116.610001	0.566932201385	116.26
...	...	...	...	...	...	...

(Date, Ticker , Volumes, Open, Close) are all obtained from the Yahoo Finance source using the Pandas DataReader. Shorts-Ratio is calculated from the shorts data obtained from Bats Global Markets (http://www.batstrading.com) . Moving average is calculated from the Close data that is obtained for the dates specified.

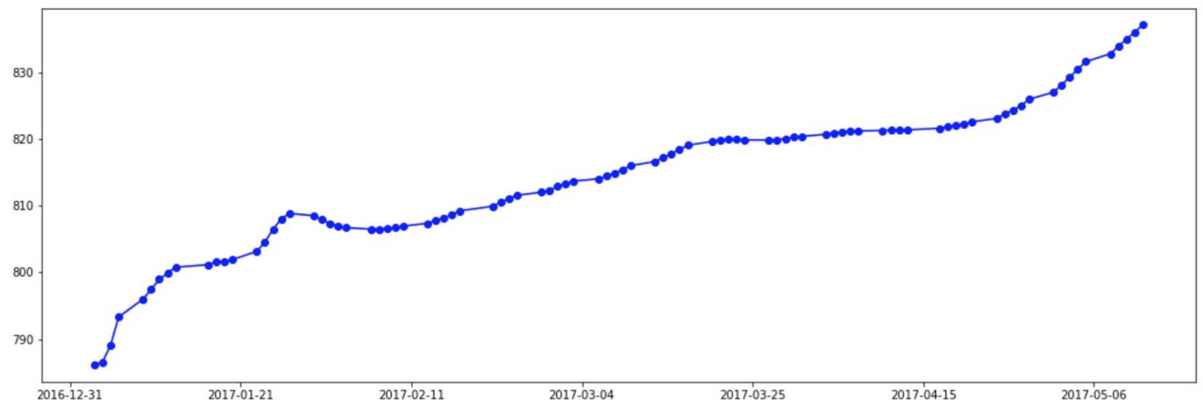
Exploratory Visualization

The “Moving Average” for each of the stock tickers are as follows:

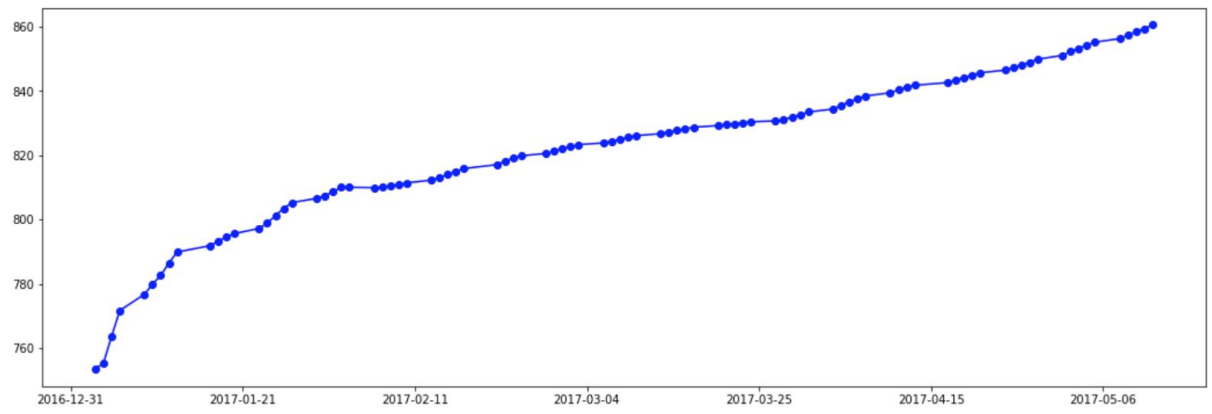
```
show('AAPL')
```



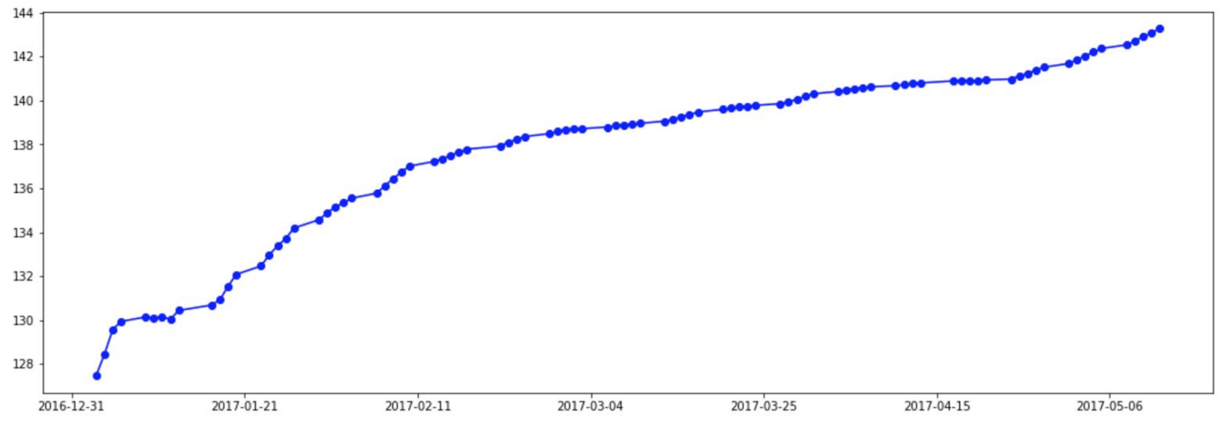
```
show('GOOG')
```



```
show('AMZN')
```

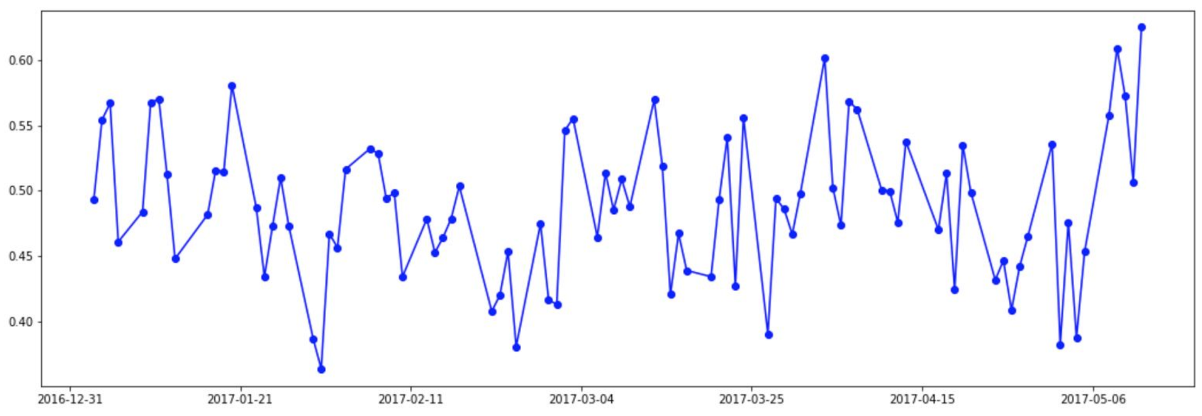


```
show('NFLX')
```

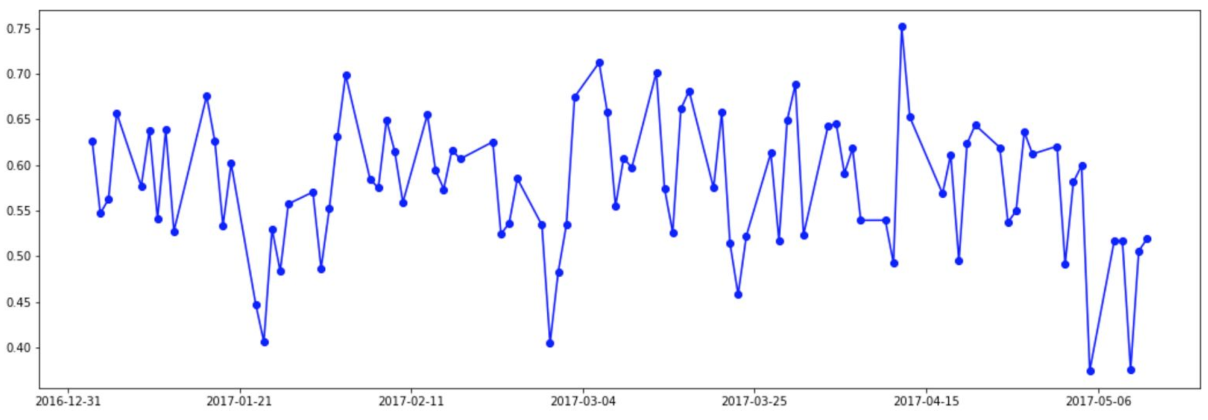


It is pretty obvious from the moving average visualizations that this gives a good general idea of where the stock is headed with pricing. Now let's look at the visualizations for the Short-Ratio

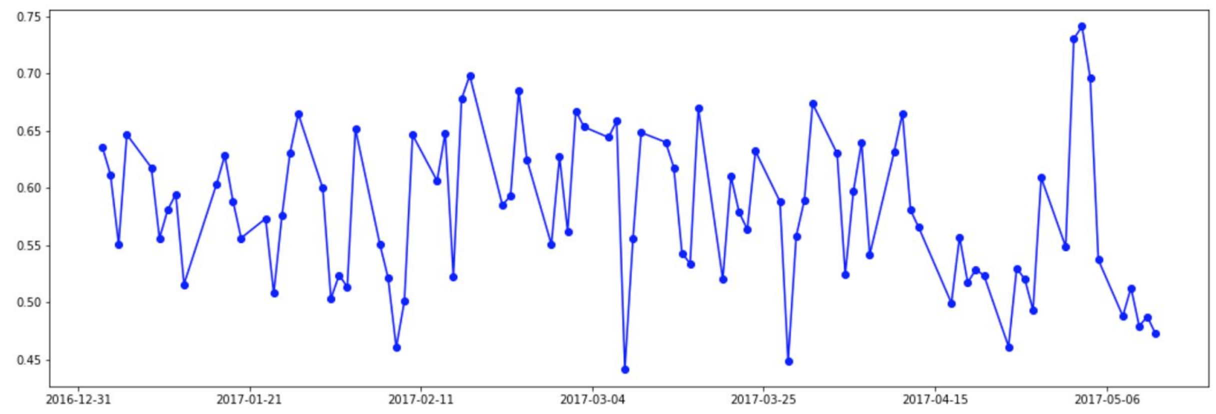
```
show('AAPL')
```



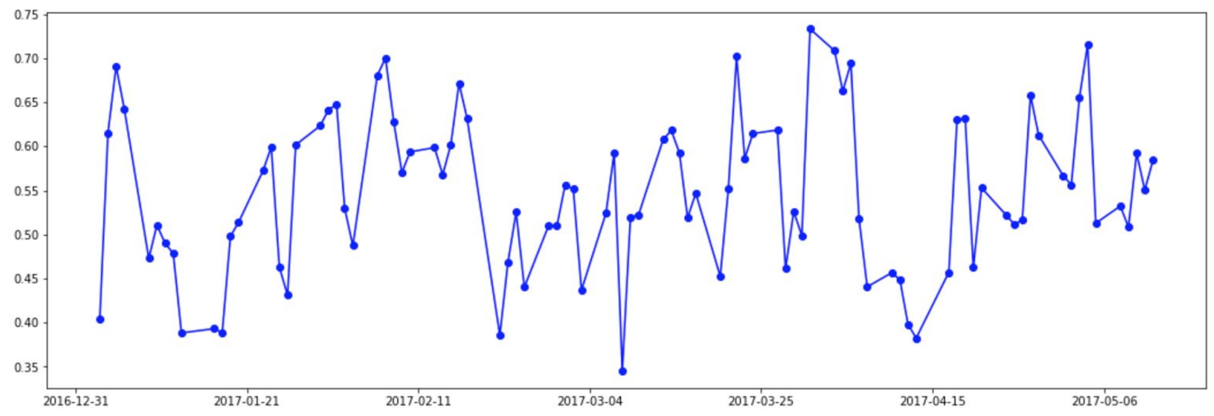
```
show('GOOG')
```



```
show('AMZN')
```



```
show('NFLX')
```



It is not very obvious from these stock visualizations but the Shorts_Ratio does affect the rapid price drops and increases in stock price and is indicative of the trend. This fact is understood from the stock trading domain knowledge.

Algorithms and Techniques

Primary technique used here is Scalable Vector Machines (SVM) algorithm called Scalable Vector Regression (SVR) . The problem here involves lots of continuous data . A variant of SVM called SVR is very good in dealing with lots of continuous forms of data. SVM is an algorithm that looks to find a hyperplane to separate data into classes.

In mathematical language, SVMs construct linear separating hyperplanes in high-dimensional vector spaces. Optimal classification occurs when such hyperplanes provide maximal distance to the nearest training data points. Using regression using SVR in this space allows for the creation of a much closer representation internal functional structure.

The main reason for choosing SVM is that this algorithm is very good for high-dimensionality, versatility and Memory-Efficiency for datasets. This algorithm performs poorly in cases where the number of features exceeds the number of data samples. We do not run into this disadvantageous case in this problem. SVM is also good at determining the multinomial factors in data.

Complex stocks may perform complex stock trading patterns that may be far away from simple linear regression. In simple words, SVM divides the dataset into multiple classes using hyperplanes and using SVR on this allows the algorithm to form the creation of a closer functional representative internally.

Prediction used in such a model with SVR is likely going to stay very close to the data set behavior, and this is likely a good solution for stock prediction.

Once all the features are collected and the Short-Ratio and the Moving Average has been calculated, we create internal data for the time difference, the number of days the query_date is ahead of the end_date of the training set. Then we add columns for the days by standardizing the dataset for the number of days we considered relevant. The StandardScaler allows us to scale the data based on the mean. Here we use Scalable Vector Machine with Support Vector Regression (Epsilon) on the test and training data that we created using the days we cared about between the end_date and the query_date because the closing price of these days is what affects the stock price the most.

Support Vector Regression is a very useful and a powerful technique to find a data pattern in a time series data. Use this pattern to do the prediction of stock price seems the most sensible.

Benchmark

Why MAE (Mean Absolute Error) and RMSE (Root Mean Squared Error) ?

Both MAE and RMSE are used heavily in model evaluation techniques on most data studies. Both the techniques allows us to look at the predicted values and re-evaluate the algorithm for prediction.

Research material like:
http://www.int-res.com/articles/cr2005/30/c030p079.pdf

indicate that there are cases where MAE is termed better than RMSE in some problems.

Mean Absolute Error is calculated between the predicted value and the test value to calculate the benchmark error.

Root Mean Squared Error is calculated between the predicted value and the test value to calculate the benchmark error

In our case we calculate the benchmark numbers internally on a part of the dataset.

E.g.
Using Linear Regression

```
%./stock_predict.py -b linear 0.6
Using TensorFlow backend.
Already exists: data/GOOG/GOOG-stock-data.csv
data/GOOG/GOOG-stock-data.csv
data/GOOG/GOOG-stock-data.csv
Row:  54  total:  91
MAE :  0.364408007568
RMSE :  0.539528135869
```

Running SVM SVR

```
%./stock_predict.py -b svm 0.6
Using TensorFlow backend.
Already exists: data/GOOG/GOOG-stock-data.csv
data/GOOG/GOOG-stock-data.csv
data/GOOG/GOOG-stock-data.csv
Row:  54  total:  91
MAE :  0.326305714535
RMSE :  0.41205645074
```

This was done only for the GOOG stock sample where we trained for 60% of the dataset and compared the prediction to the remaining 40% of the dataset. We then calculated the MAE and the RMSE with the prediction value of the algorithm against the real value. The case above was done for SVM and Linear Regression.

The SVM case produced benchmarks slightly better than the Linear Regression.

Data Preprocessing

Base Stock trading data is read from Yahoo Finance. Base shorting data is read from batstrading.com. We care most about the “Close” price for each trading day. We process the data internally for the days we have a difference between the query_date and the end_date with some additional days added in. It is done this way because the stock trading prices in the dates closer to the query date and the most important. We standardize the data on these days.
We build up a internal table keyed on the time difference and create columns on the configurable value of buffer updates and the data for number of days time difference between the end_date and the query_date.

This is the only data preprocessing we do.

Implementation

In this project we used the Linear Regression, Scalable Vector Machines (using SVR) and Neural Nets. We used the prediction with the MAE and RMSE error to calculate which technique produces the best prediction result.

In the data processing we have to make sure the Shorts-Ratio and the Moving Average is calculated. The date of the Shorts value had to be aligned with the dates for the trade after fetching the two datasets separately.

In the SVM algorithm we split the data into test and training set at 30%. We use the Epsilon-Support Vector Regression. It needs the value of epsilon to specify and we chose 0.1. Within a distance of 0.1 (10%) of the points predicted we consider no loss in this case. For internal scoring in the algorithm itself, we choose r2 scoring with the Closing price.

In the Linear Regression case, we chose the simple fit_intercept and the normalize options with only 25% split into training and test data.

In call cases, the data used is the scaled data that is output from the Standardized scaler from preprocessing.

We calculate the metrics and determine which technique works the best in this scenario. We found the SVM (svr) technique to work very well.

Refinement

Initially we started out with Epsilon of 0.3 and reduced it to 0.1 to match the desired 10% allowable error case. This was the hyperparameter tuning we did in SVM SVR case.

The only tweaks we made in the model were adding the Moving Average, making sure that the Shorts-Ratio fits the same date range, and filling up some of the error cases on the data alignment.

Sample Run

With the Query of the following :

```
q0 = { 'ticker' : 'GOOG', 'begin' : '1-1-2017', 'end' : '5-12-2017', 'query' : '6-12-2017'  }
q1 = { 'ticker' : 'AAPL', 'begin' : '1-1-2017', 'end' : '5-12-2017', 'query' : '6-12-2017' }
q2 = { 'ticker' : 'AMZN', 'begin' : '1-1-2017', 'end' : '5-12-2017', 'query' : '6-12-2017' }
q3 = { 'ticker' : 'NFLX', 'begin' : '1-1-2017', 'end' : '5-12-2017', 'query' : '6-12-2017'  }
```

Sample run looks like this :

```
%./stock_predict.py -s
Using TensorFlow backend.
Already exists: data/GOOG/GOOG-stock-data.csv
data/GOOG/GOOG-stock-data.csv
stock  GOOG  predicted for  6-12-2017  is :  932.219971
Already exists: data/AAPL/AAPL-stock-data.csv
data/AAPL/AAPL-stock-data.csv
stock  AAPL  predicted for  6-12-2017  is :  156.100006
Already exists: data/AMZN/AMZN-stock-data.csv
data/AMZN/AMZN-stock-data.csv
stock  AMZN  predicted for  6-12-2017  is :  961.349976
Already exists: data/NFLX/NFLX-stock-data.csv
data/NFLX/NFLX-stock-data.csv
stock  NFLX  predicted for  6-12-2017  is :  160.809998
```

After the first review, we added a -b option to evaluate the predictions with real values to see what we get.

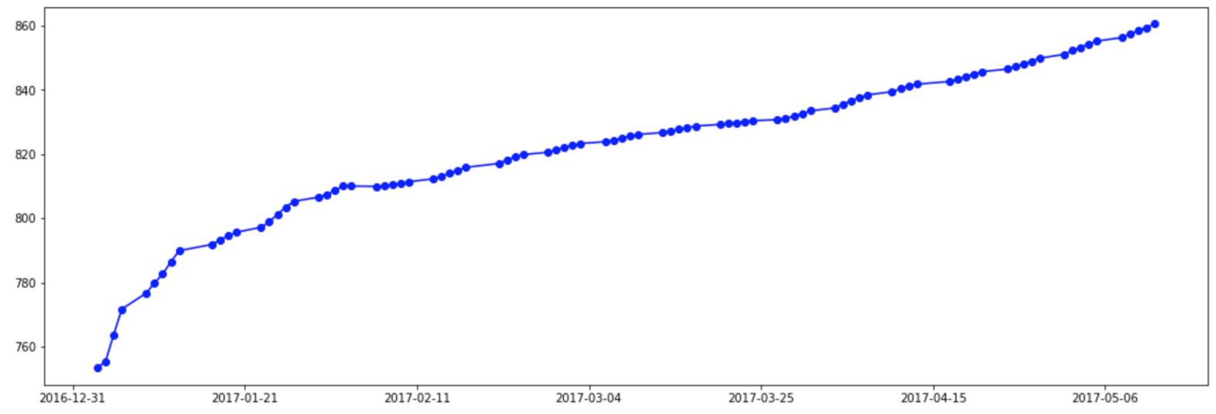
E.g.
Here is a sample run for the bench marks for 95% of the data used as begin and end for the algorithm and the remaining 5% we predict to see accuracy

```
%./stock_predict.py -b svm 0.95
Using TensorFlow backend.
Already exists: data/GOOG/GOOG-stock-data.csv
data/GOOG/GOOG-stock-data.csv
data/GOOG/GOOG-stock-data.csv
Row:  86  total:  91
MAE :  0.249172149919
RMSE :  0.249851495915
Already exists: data/AAPL/AAPL-stock-data.csv
data/AAPL/AAPL-stock-data.csv
data/AAPL/AAPL-stock-data.csv
Row:  86  total:  91
MAE :  0.0627773272789
RMSE :  0.063713540784
Already exists: data/AMZN/AMZN-stock-data.csv
data/AMZN/AMZN-stock-data.csv
data/AMZN/AMZN-stock-data.csv
Row:  86  total:  91
MAE :  0.273752188731
RMSE :  0.278292882498
Already exists: data/NFLX/NFLX-stock-data.csv
data/NFLX/NFLX-stock-data.csv
data/NFLX/NFLX-stock-data.csv
Row:  86  total:  91
MAE :  0.0251218844963
RMSE :  0.0301494026927
```

This shows that the predictions are very close.

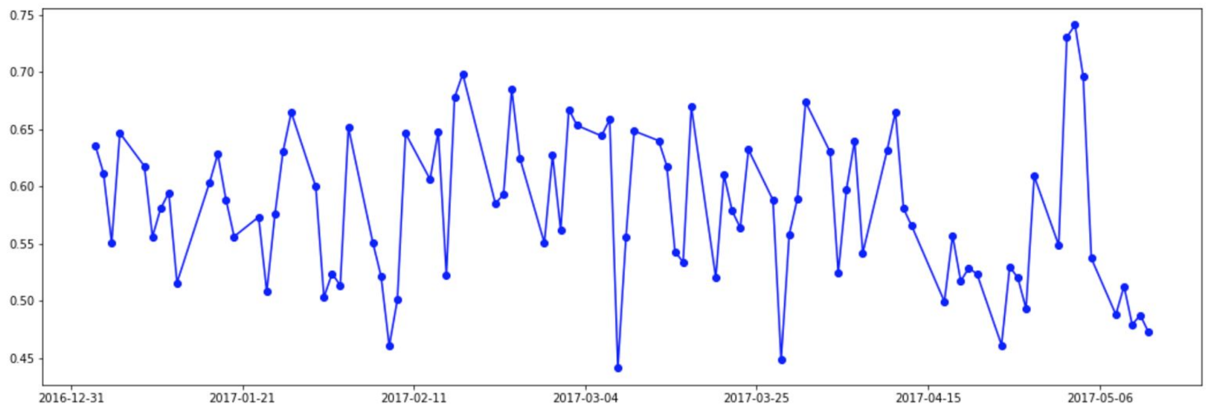
Understanding the Visualizations

```
show('AMZN')
```



This is a pretty normal steady growth moving average. Now let’s look at the shorts ratio.

```
show('AMZN')
```



As it is clear the steady growth continues as the shorts have fallen, although there is a brief period of time the shorts have increased. Now of the shorts data is obtained in the future date closer to the prediction date, it could be applied to affect the prediction value.

The moving Average visualization is pretty clear that this is a very valuable feature in the training set as it determines the prediction direction.

Model Evaluation

The most important features used were : Price Closed, Moving Average and Shorts-Ratio and the time difference
The shorts-ratio and the moving average are extremely important in the prediction

Justification

Data preprocessing to the time frame that is more relevant by taking the time difference, and using SVM with SVR (for best time series based data) to do the modeling is preferred. Looking at the MAE and RMSE data , it appears that the SVM SVR was the best way to predict the stock for the query_date value.
This was proven with the benchmark method for the past data.

Summary

The problem presented was to predict the price of a stock at a later date given the stock trading data between two past dates. We approached this problem by first fetching the basic stock trading data from Yahoo finance, then we fetched the Stock shorting data from batstrading.com , calculated the moving average and put all the data together.

We then used the SVM SVR method to train and test the dataset, because this is a very good method for time series data. The same mechanism was used to benchmark a portion of the data from the past itself.

Interesting parts of the project were to relate the shorting data on the stock. In this project we used it in the model itself, although the value of shorts data is much closer to the prediction date. If the prediction was to be done within a day, the shorts data would be much more valuable, but for a trend it is still pretty good. Another interesting use was the use of Moving Average for the closing price of the stock. This feature is commonly used in stock trading industry as a useful feature.

Potential Improvements

There are many more features that can affect the stock price prediction, and they could not be considered in this project. One area we wanted to include was the news items , and stock pricing index of the companies considered to be in the same industry as this stock. E.g. for the Stock GOOG, other stocks like MSFT, FB, AAPL , BIDU are in the same group. Taking a prediction index for others and used it with GOOG would have been very valuable.

The final model we chose in this project is suitable for a decent solution but there are many more features that affect the stock price and they are at different parts of the time series. E.g. the shorts data is more valuable towards the prediction date. This data cannot be obtained in the problem given. If the prediction was to be done within hours in a day, the shorts ratio has to be used directly with the model generated prediction price.

The improvements should make the stock price predicted value a lot more accurate to the prediction for the benchmark we used.

Although it is impossible to precisely predict the stock price, it is possible to get fairly close prediction. In practice, hedge funds and other stock related finance industrial exports use systems built using Deep Learning RNN (Recurrent Neural Networks). The features used are a lot more extensive like Global country GDP Index, Economic Data, Geopolitical events and Index performance. Most of these solutions have a lot more data feeds for all other features, and many different algorithms and perform much better,

In this project, we attempted a simple case of stock prediction that we can evaluate with a measurable benchmark.