Investment and Trading:

Stock Predictor

Definition

Project Overview

The stock market is a collection of buyers and sellers of fractions of company ownership. Buyers and sellers can take a variety of forms from individuals to investment firms to funds. Their goal is to understand how to make more profitable decisions with their investments. Copious data is available in the form of historical prices. These prices will be used to generate training data for machine learning algorithms.

Data used in this project is retrieved from the Yahoo Finance API provided through the pandas library. Historical data including the information of concern, adjusted closing prices, are retrieved using a company symbol, an uppercase group of characters representing how the company is referred to in the market ("AAPL" for Apple, "ABC" for Google, "IBM" for IBM). The historical adjusted closing prices are requested for all trading days in the past year up to the day the program is run as specified in the project README.md file.

Accurate predictions of future stock returns provide the opportunity to make profitable and responsible investments. Stock price movements, increases and decreases in stock value, are known to be difficult to predict. Stock prices are subject to a lot of context created by participants in the stock market. While saturated with noise, there may still be information that can be exploited to make more responsible market strategies.

Problem Statement

The endeavor of this project will be to provide return predictions for investments over a user defined period from the present day. Data, retrieved from the Yahoo Finance API, will be used to construct features and an output for a machine learning algorithm. Making these predictions falls under the category of supervised machine learning models, and even further in the realm of regression. The prediction, a numerical output, will mostly range between positive and negative one. The inputs, or features, retrieved from Yahoo Finance will be the Opening Price, Closing Price, High Price, Low Price, Adjusted Closing Price for a user provided Stock specification.

The final model will output an estimated percent return on purchasing stocks from the specified company. For example, given the stock ticker symbol, "ABC", and an investment period, 5 trading days or 1 week, provide an estimate of how much that stock will increase or decrease.

Metrics

Several performance metrics are used to determine the efficacy of the predictions. The metrics used to measure the performance of the predictions as they relate to the actual results will be Root Mean Squared Error (RMSE) and Correlation. Further value of the predictions will be determined using the returns of a simple trading strategy.

RMSE is used because it gives a scale of the regression error in the same units as the prediction itself. Since the outcome this project will focus on is investment return, the percent increase or decrease over the present value of the stock, normally on a scale from -1 to +1. The benchmark to beat would be a prediction of 0 for all future returns. This does not seem like much, but in short time periods stock prices are not expected to change by large amounts. RMSE is calculated using the below formula:

RMSE = Square Root(Average((Actual Returns - Predicted Returns)^2))

The correlation between the predicted results and the actual results is observed to determine the extent to which a linear relationship exists between the predicted and actual results. A high positive correlation, meaning the predicted returns were low when the actual returns were low, would mean valuable results. The benchmark for this is set in the analysis section by analyzing how well the current price correlates with future results.

In addition to those metrics, the predictions will be measured by their performance when used by a simple trading strategy. Through this strategy a portfolio will be developed. The metrics used here will be overall portfolio return vs stock return. In addition, the risk adjusted return will be calculated using the Sharpe ratio, a measure commonly used in trading for calculating the return as adjusted by the risk.

N = number of trading days per year

Ret = returns

Rfr = risk free rate of return (0.0 will be used.)

Sddr = standard deviation of returns

Sharpe Ratio = Avg(Ret – Rfr) / sddr * √(N)

Analysis

Data Exploration and Visualization

The below data is a sample of raw data as retrieved from the Yahoo! Finance API. This particular data represents what would be returned if the historical pricing were requested for IBM from June 14th, 2013 to Jun 19th, 2013.

Date	Open	High	Low	Close	Volume	AdjClose
6/14/13	203.970001	204.740005	201.809998	202.199997	2804500	185.494397
6/17/13	203.440002	205.169998	202.550003	203.039993	3219900	186.264993
6/18/13	203.020004	206.089996	202.869995	204.869995	3277800	187.943802
6/19/13	204.440002	205.029999	201.929993	201.940002	2846100	185.255883
6/20/13	200.669998	201.699997	197.279999	197.350006	4514800	181.045108
6/21/13	198.5	198.520004	193.539993	195.460007	8914800	179.311259

The raw data shown above is as follows. The 'Open' column represents the opening price, or the price the stock was at the time the markets opened on that day. The 'High' and 'Low' columns represent the highest and lowest observed price of the day, respectively. The 'Close' column represents the closing price, or the price the stock was at the time the markets closed on that day. The 'Volume' column reports the number of times the stock was bought and sold. Finally, the 'AdjClose' column is the adjusted closing price is the closing price adjusted for corporate actions that took place before market open.

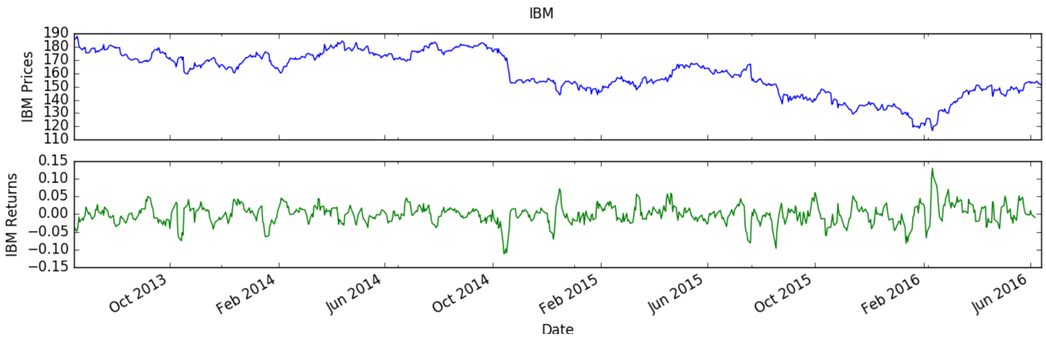
The output for this project, 5-day return, is calculated using the adjusted closing price. Adjusted close is used instead of close price because it is adjusted for events like stock splits, dividends and more. It is a more reliable representation of the true value of the stock. The formula for calculating 5-day return is shown below:

Return = (Adj. Close[t+5]/ Adj. Close[t]) -1

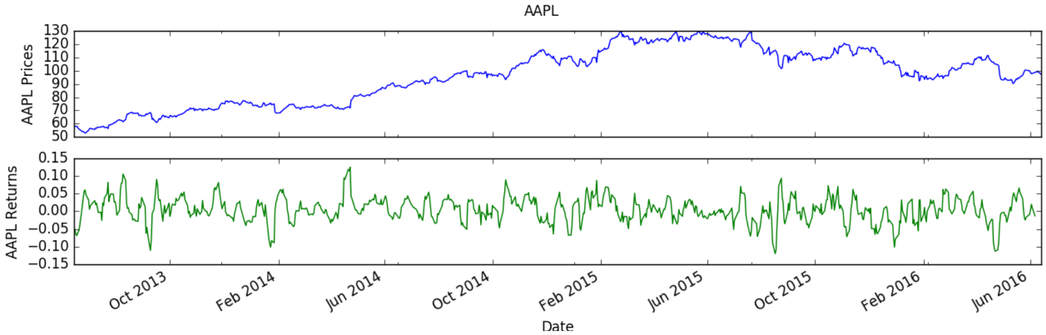
As an example, the 5-day returns of buying an IBM share at $185.494397/share on 6/14/13 and selling that share on 6/21/13 at $179.311259/share is -0.0333332872. In other words, the trader would receive a negative 3.33% return on their investment. They would lose money. However, this calculation was done after knowing the true future value of the stock. An investor seeking to buy stocks on the 14th of June, 2013 would only see the current value of $185.494397/share. This is where prediction can help.

Very few abnormalities existed in the raw data. Gaps in time where no historical data was provided were the only issues that needed to be addressed prior to building the input feature set. It is common practice to feed the last observed price forward to the next date where data is present.

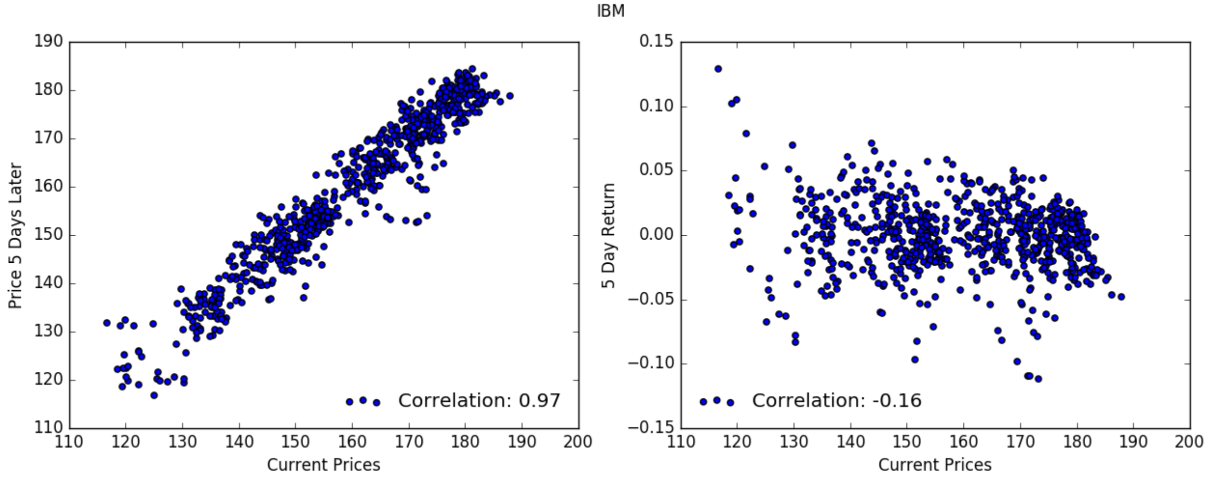
The data used in the analysis portion of this project is similar to that shown above except that it ranges from June 14th, 2013 through June 13th, 2016. With that time frame there are 755 days, or observations in each stock dataset. 375 or 49.67% of those days observe a *positive* 5-day future return. 375 or 49.67% of those days observe a *negative* 5-day future return. When calculating the returns, the last five rows, or the number of days in the horizon, is removed from the end of the dataset. That means that after removing those five rows, 50% of the days have a 5-day return above zero, and 50% of the days have a 5-day return below zero. The average 5-day return for all rows in the dataset is -0.09% with a standard deviation of 0.0270. These particular results were calculated specifically for the IBM data.

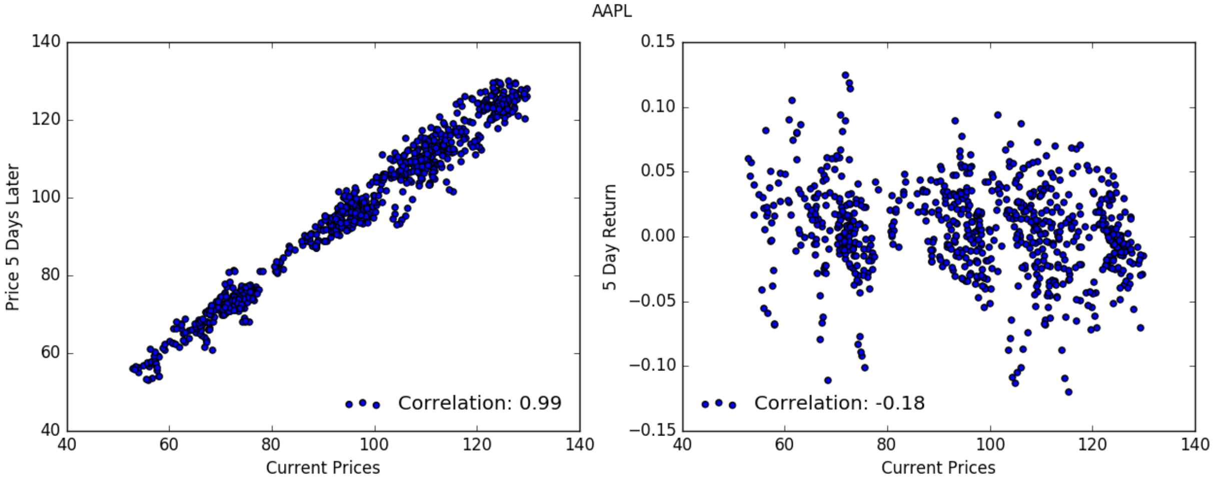


For another example, Apple or "AAPL" also has 755 rows in its raw dataset. After creating the returns, the dataset has 750 rows. 413, or 55.07 %, of those days observe a *positive* 5-day future return. 337, or 44.93% of those days observe a *negative* 5-day future return. The average 5-day return for all rows in the dataset is -0.43% with a standard deviation of 0.0352.



Over all the datasets, approximately 54.58% of the observations in each dataset reported a *positive* 5-day future return. However, the average of returns over all datasets was only 0.26% with a standard deviation of 0.0352. The complex nature of presenting this data is that trends in the stock market are constantly changing. In this project, a variety of Company symbols will be used to display how the analysis was conducted.





The above plots are examples of how the prices and returns relate to the current price. The high correlation between the current and future price is easy to explain for a short prediction period. When prices are low, prices 5 days later are likely to remain low by comparison. Prices are unlikely to make many huge jumps in price. On the other hand, there is a moderate negative correlation between current price and 5 day returns. This implies that when prices are low, a positive return would be expected, while when prices are high, a negative return is somewhat more likely. As the main concern of any investor is the future return, the overall goal of the project is to out-perform the benchmark for predicting returns.

Stocks were chosen from the S&P 500 as of May 2016. However, the model is meant to perform generally well for predicting future returns on any stock the user inputs. The companies from the S&P 500 were chosen simply for being a large collection of stocks where a list would be available. A benefit to using these stocks is that the companies in the list are well-established companies that have a large history of data and will probably exist for a good time into the future.

Algorithms and Techniques

Features

A variety of indicators are explored to analyze the movement of prices in the market. The most commonly referenced and easily implemented indicators were selected for use. While more indicators can be added to the collection of indicators tested, the section below illustrates the current set of indicators and how they were tested to determine which would be most effective.

The list of window based indicators tested was:

- N-Day Bollinger Value
  - *(Price[t] - N-day Moving Average) / (2 * N-day Standard Deviation of the Price)*
- Exponential Moving Average (EMA)
  - *(Price[t] - EMA[t-1]) * (2/(N+1)) + EMA[t-1]*
- N-day Simple Moving Average (SMA)
  - Average(*Price[t-N] to Price[t]*)
- N-day Momentum
  - *Price[t] / Price[t-N] - 1*
- Relative Strength Index (RSI)
  - RSI = *100 - 100/(1+RS)*
  - RS = *N-day Average Gain / N-day Average Loss*
- Volatility
  - N-day Standard Deviation of the Price

The list of non-window based indicators tested was:

- Weekdays
  - Is it Monday? [0 or 1]

    - Is it Tuesday? [0 or 1],
    - etc.
- High – Low
    - The spread of the last trading day's High Price and Low Price
- Volume Percent Change
    - The percent change of number of shares trades from the previous day

Every window indicator was analyzed for varying window size from a one-day window to a thirty-day window. Due to the nature of some of the indicators, a later start of three or four days may have been used. For every member of the S&P 500 list found in spylist.csv:

1) A dataset was created with the specified indicator and specified window.

2) The dataset was split into training and testing data.

3) A linear regression model was trained using training data.

4) The linear regression model predictions were compared the test data.

5) Test error for the indicator was added to a running total over the S&P 500 list.

6) Test errors were sorted to find the best performing indicator window.

For example, a linear model would be trained using adjusted close and the simple moving average (SMA) of the past two days and then the model would be assessed on a test set. Then a model using the adjusted close and a SMA of the past 3 days would be trained and the performance assessed on the same test set. This went on until the window was up to 20 days.

The top three performing windows for each indicator were added to another list. All combinations up to a size of 4 indicators were then using the same process as listed above. This would determine the best set of indicators to use for the final model.

Models

As for the models, this is a supervised machine learning regression task. Therefore, it will require the use of regression models. A variety of regression models are available from the Scikit Learn library. These models provided by Scikit Learn are quick to train and quick to query on the amount of data they will have to work. This is important given that the models will be frequently retrained on later data to adapt to the dynamics of the market.

A variety of models are explored because they each take a different approach to approximating the target value. Some are parametric models, like linear regression and support vector machine regression, while others are non-parametric, like k-Nearest Neighbors. Because the underlying forces that drive the market are unknown, it seems prudent not to assume the relationship all the features have with the output. For that reason, a variety of model and their hyper-parameters are explored.

The models are tested and tuned to find the hyper-parameters for each model with the best performance on the test data. Those models include Support Vector Machines, Linear Regression, k-Nearest Neighbors, Decision Trees, and Bagging, which uses the previously listed algorithms. After model tuning, the best of each model will be compared against the others to find the best performing models. After the best two models are determined, a combination of those models will be explored.

Benchmark

There are a few benchmarks that this project will endeavor to beat. The first is to achieve a better test root mean squared error (RMSE) than predicting zero. This benchmark is essentially the equivalent of the standard deviation of the returns considering the average is essentially zero. Returns are well known to average out to 0 over short periods. Short term returns are known to be very noisy. Aside from that discovered in research, the data exploration section showed roughly half of the 5 day returns to be above and below zero.

Another benchmark is to achieve a better correlation than that of the adjusted close to the 5 day returns. A correlation cannot be formed from a comparison between all zeros and the actual returns. Further, always predicting a 0.0% future return never allows for any trading opportunities. This is why the Adjusted close correlation will be compared against the returns.

Finally, a measure of success for the use of this predictor is how well a strategy based on the predictions performs versus the market. A strategy that performs well will have an overall return greater than the overall return of the market and a higher risk adjusted return.

Methodology

Data Preprocessing

As mentioned in the Analysis section, few abnormalities existed in the raw data. Addressing the missing data was done using some of the functionality of the pandas library. *DataFrame.ffill()* addresses the gaps at the very beginning of the data can be addressed with *DataFrame.bfill(),* which uses the first known quantities and feeds them back through all preceding rows. DataFrame.ffill() is run prior to DataFrame.bfill() to avoid propagating future prices backward and incurring a look-ahead bias.

Other than the NaN values discussed above, there were some that were produced by the indicators developed in this project along with in the construction of the output data. Some indicators were constructed from a defined number of trading days into the past. This meant that for some days not all the information was available to create the indicator and NaN values were produced. These rows were simply eliminated using DataFrame.dropna().

After adding indicators to the data, many of the features were on largely different scales. To some machine learning algorithms, the size of different each feature will matter, while output data does not need to be normalized. Subsequently, the data was normalized using mean normalization. The mean of each feature was subtracted from each row, and then divided by the standard deviation of each feature respective of the feature.

Implementation

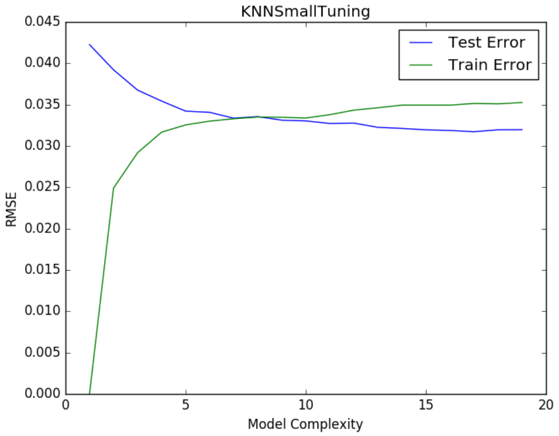
Separating data into training and test sets was completed using the earlier data to train and the later data to test. Training and test data cannot be randomly selected because it would incur a *look-ahead bias*, which simply means that the model would have access to information that it normally would not have access. For this reason, data was split using 9 months as data for training, and then the subsequent three months for testing.

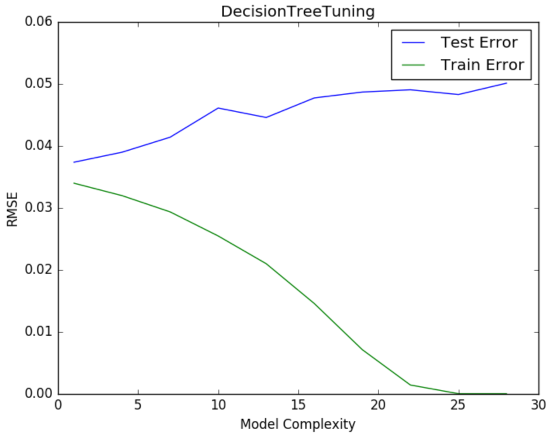
Linear regression was used to assess the efficacy of individual predictors. The final model settled on the percent change in adjusted close from the previous day, and two additional features. Those features include the percent change in volume from the previous day and the current difference between the high and low price. These results outperformed other combinations by a very narrow margin.

The below dataset is an example of the data provided to the learning algorithm. Data has been normalized to assure that no individual feature is more strongly weighted than another. What is included in the features are three details from the dataset, percent change in Adjusted Closing Price, percent change in Volume, and the high minus the low for that day. Each feature is utilized in the end model for predicting future return. After running exhaustive tests with testindicator.py, many indicators did not appear to provide clear benefit to a Linear Model's predictive capabilities. In fact, through the testing process described in the Algorithms and Techniques section, most indicators made for a worse test error than using Adjusted Close alone. The output is the right side of the table.

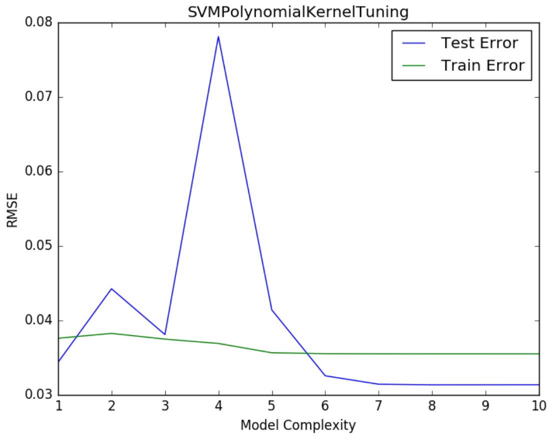
Date	AdjClose_AAPL	Volume_AAPL	HmL_AAPL	y_AAPL
6/17/13	0.236229115	-0.265297136	0.316093485	-0.068194448
6/18/13	-0.087495719	-0.795431018	0.137848733	-0.067489642
6/19/13	-1.351222747	1.41135795	1.226472517	-0.058936171
6/20/13	-0.983902335	0.24525544	1.816020978	-0.055321042

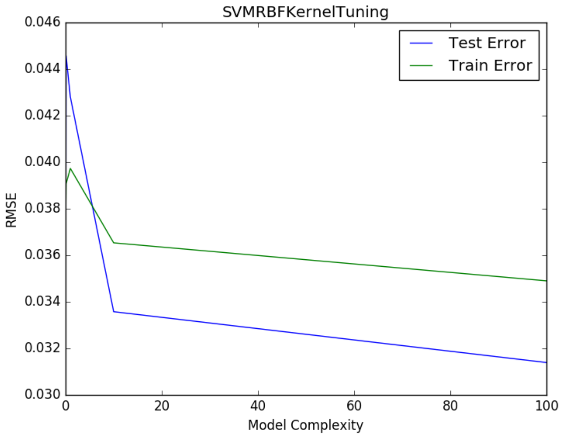
Models were tuned and evaluated using a similar process to that of the indicators. Models were trained using the above features for a nine-month period and tested on a three-month period with a regimen of increasing or changing model parameters. Their successes and failures to adapt to the data can be observed below.





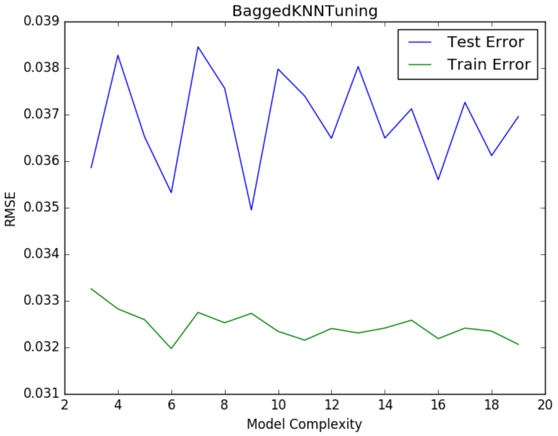
kNN appears to perform better with more nearest neighbors, but it is only converging on the benchmark error, which is the assumption of 0% returns. For that reason, when kNN is used on the end model, it will use only 10 nearest neighbors. Decision Tree tuning turns out to quickly get worse with more depth until it overfits to the training data and does worse on the test set from the beginning depth.

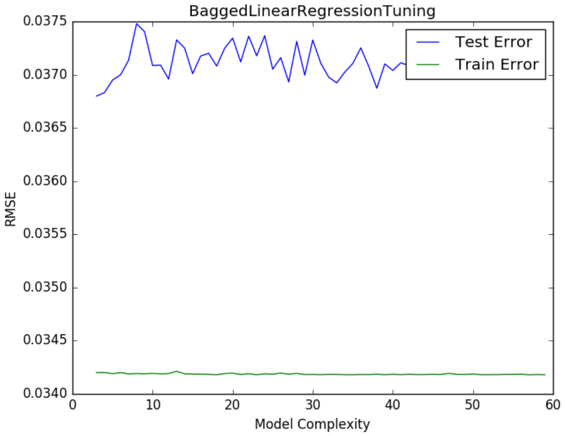




SVMs and Bagged models were also explored. Both the polynomial and RBF SVMs shown above have test RMSEs that quickly approach the benchmark as their parameters are tuned. They both consistently predict return values on the order of 10^{-5} or closer to zero, which is unhelpful.

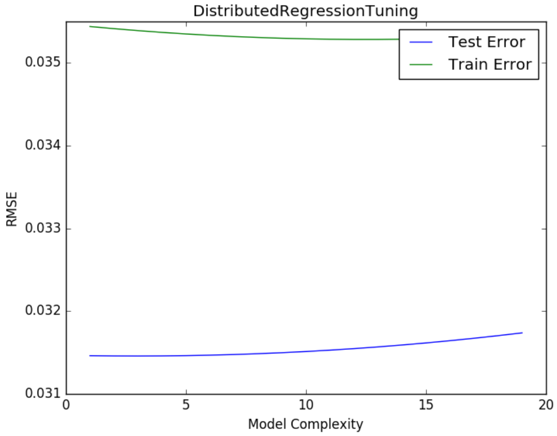
All algorithms were combined with the bagging learner and tuned by increasing the number of bags. All results look similar to the results shown in the graphs below. No defined improvements were observed with the combination of bagging.



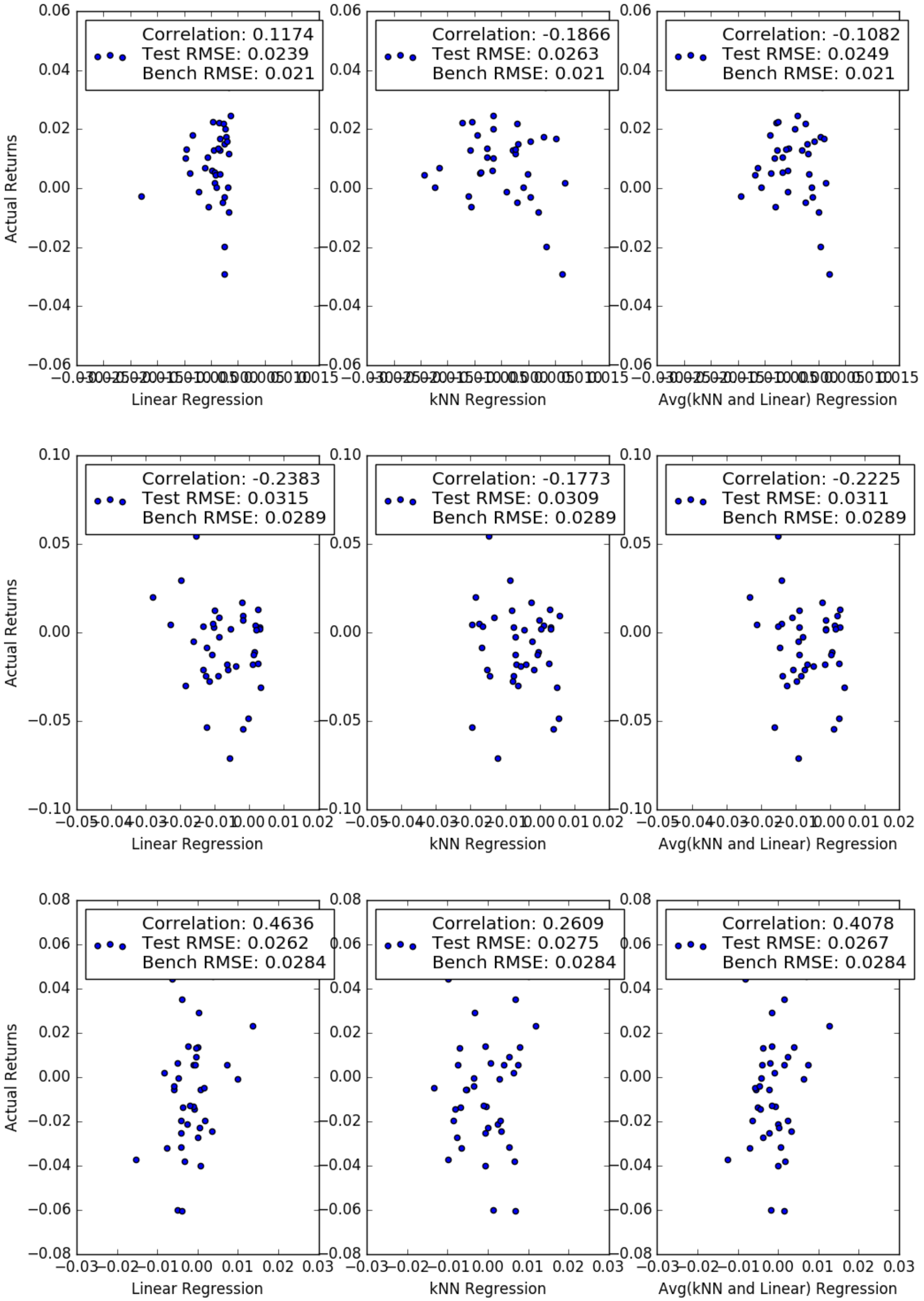


Refinement

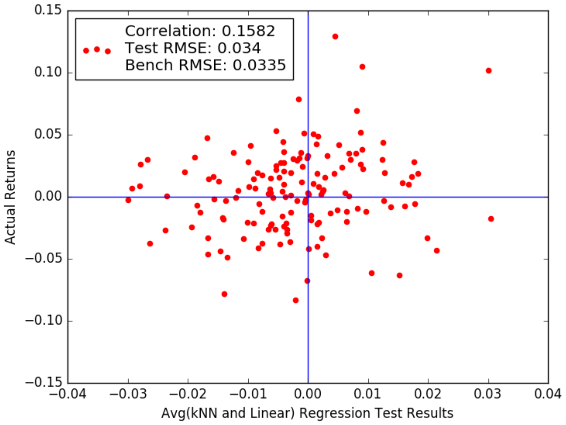
In observing the capabilities of each model in the precious section it occurred that a model with a combination of learners might be able to avoid regressing to the overall average return, 0.0%. The distribution learner shown below, uses different proportions of linear regression and k-Nearest Neighbors. The leftmost border of the graph represents 100% weight placed on the predictions from the linear regression model, while the rightmost border of the graph represents 0% weight placed on linear predictions and 100% weight placed on k-Nearest Neighbors.



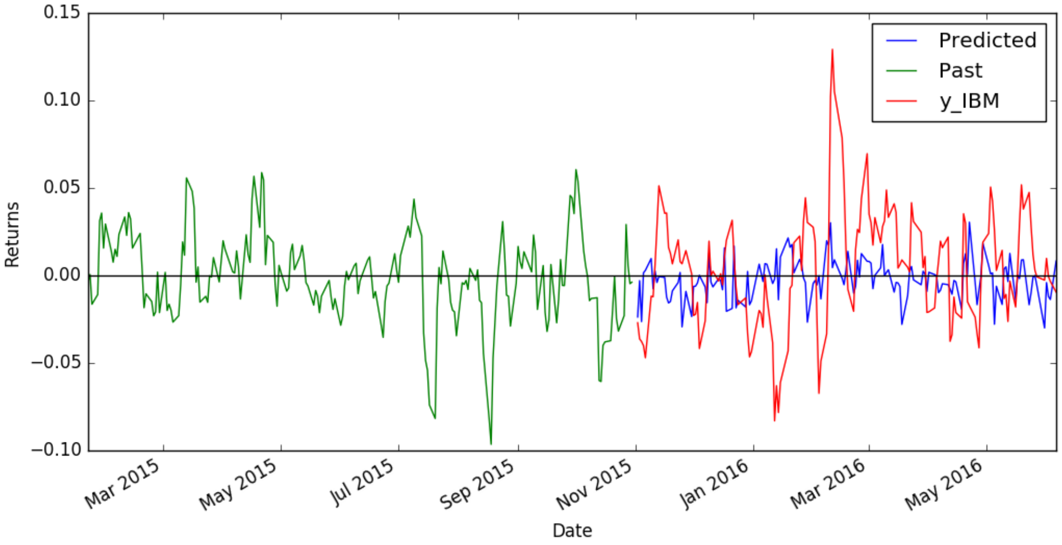
The below three sets of graphs explore the effects of averaging the output of a linear regression model and a k-Nearest Neighbors regression model. In some situations it appears that linear regression outperforms k-nearest neighbors, while not so much in others. Using an ensemble of models can be beneficial to estimating future return and is used in the final model.



The final two graphs are used to express how well the final model has performed on the task. The first graph is a scatter plot that expresses the correlation between the predicted and actual results for the test set. The legend shows the correlation, which exceeds that of the benchmark by being both larger in magnitude and positive. On the other hand, RMSE for the predicted is still slightly greater than that of the benchmark. Still, having a better correlation implies that trading opportunities can be recognized at all, while a prediction of 0.0 returns will never produce recognize any opportunities.



Statistics concerning the accuracy of the model express how often the predicted and test returns were above zero and also how sensitive the model is to predicting positive and negative values. Only 40.27% of the predicted returns were greater than 0.0, while 56.38% of the actual test data returns were greater than 0.0. 44.05% of the time the return was positive, the model predicted it would be positive. 64.62% of the time the return was negative; the model predicted it would be negative. It appears that the model performs better at predicting negative values than it does at predicting positive values. This information can be utilized in a trading strategy to short stocks more accurately than random chance.



Lastly, the bottom graph shows the 5-day returns recognized over the training data on the left side in green. In the fourth quarter on the right of the graph are the predicted returns over the test data and the actual 5-

day returns. Future explorations of this type can search to classify whether or not the return will be greater than or less than 0.0 returns.

Results

Model Evaluation and Validation

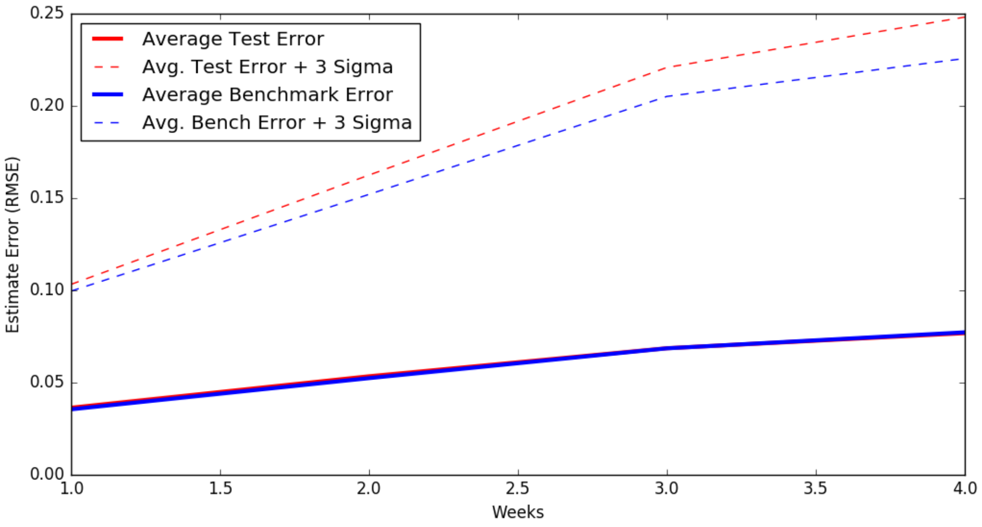
The table below is used to describe predictions made for all companies in the project list. The model used in making the predictions was an ensemble of regression models. Specifically, there are two types of regression models, k-Nearest Neighbors and Linear Regression. The k-Nearest Neighbors parameter k was chosen to be 10 to avoid the bias that is addressed in the model tuning shown in the Implementation section. Linear regression was tuned by exploring the efficacy of each feature to predict the outcome., also discussed in the Implementation section. As in the previous section, the model is run for all ticker symbols in spy_list.csv and the results are analyzed and presented using the below statistics.

	Average	Median	Standard Deviation
Test_Error(RMSE)	0.036385	0.030461	0.022293
Bench_0(RMSE)	0.035533	0.030249	0.021356
Test_Corr	0.215578	0.229058	0.198156

The parameters, as stated in the implementation section are percent change in Adjusted Closing Price, percent change in Volume, and the high minus the low for that day. The input parameters for the program itself include: horizon, stock symbol, learner, and if they would prefer to predict prices. The default output of the model is a 5-day return for all stock symbols in the project list using the learners previously stated. This output is equivalent to a one week investment period. Predictions can be made over longer or shorter periods of time depending on the user's request. The user can also adjust the request to only use a specific company. If the user would prefer, they can also predict future prices, if they choose, though the main concern is what they would gain by investing and that is calculated as the return.

Justification

The results, as shown below, are comparable to the benchmark over a number of weeks. Unfortunately, when the test error beats the benchmark, the overall error is too substantial to consider as a feasible predictor of returns.



It is important to understand that the market is a very noisy collection of data that is a conglomeration of a variety of factors. Certainly, not all indicators are known about and most investors underperform against the market. This statement is make clear that predicting the returns of any stock using only historical pricing data has a very low success rate.

This model uses the most readily available information, historical pricing data. Companies today use metadata from new feeds, twitter, etc. to explore the value of public sentiment as a feature in predicting price movements for a given company. However, the predictions calculated by this model can help in a trading strategy. There is a moderate positive correlation between the predictions and the actual results. The information provided in the predictions may be used to develop a trading strategy by buying stocks for the specified horizon when the prediction is strongly positive.

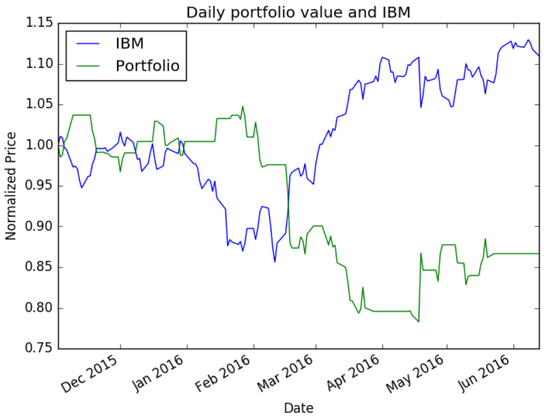
Conclusion

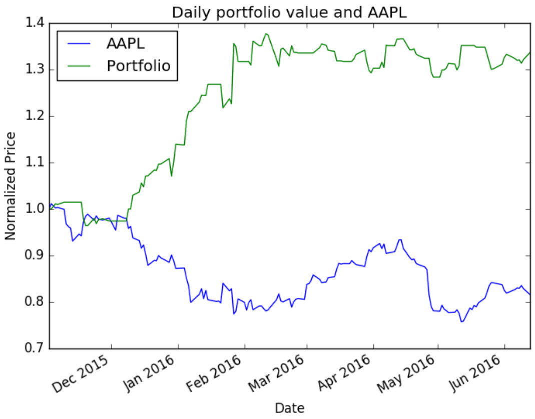
Performance overview

The IBM predicted returns are explored for the following trading strategy. The same horizon that returns are predicted for is used to buy and sell stocks over the same time. The starting value as well as the number of shares to buy and sell is defined so that someone does not have to be very wealthy to see the potential benefits of using these predictions. The threshold is used to maximize the number of trades made while not simply following the market.

Stock	IBM	AAPL
Portfolio Start Value	$1,000.00	$1,000.00
Portfolio Final Value	$866.62	$1,336.52
Date Range	2015-11-02 to 2016-06-13	2015-11-02 to 2016-06-13

Sharpe Ratio of Fund	-0.805583699	2.095602619
Sharpe Ratio of IBM	0.897396864	-1.127051499
Cumulative Return of Fund	-0.13338301	0.33652442
Cumulative Return of Stock	0.109549461	-0.183978084
Standard Deviation of Fund	0.015968733	0.015228874
Standard Deviation of Stock	0.013671679	0.016733404
Average Daily Return of Fund	-0.000810365	0.002010372
Average Daily Return of Stock	0.000772869	-0.001188031





The plots above show both the value of the stick and the portfolio normalized by the starting cash value, $1000, of the investor. The strategy utilized in this portfolio is to make a trade for the number of shares, either Buy or Sell 10 shares, if the absolute value of the 5 day return prediction exceeds the threshold value, 0.01, and then do the reciprocal, either Sell or Buy, after the horizon, 5 trading days, has passed.

In this instance, the IBM model, shown on the left, achieved a -13% return while the actual value of the company rose 11%. Furthermore, the Sharpe ratio for the portfolio following the prediction-based trading strategy is negative, as expected, and just as risky an investment as investment in the stock itself.

On the other hand, the AAPL model, shown on the right achieved an overall return of 33% over size months. That is a substantial increase, especially considering that the actual stock fell in value by almost 20%. The volatility of both the portfolio and the stock are comparable, but due to the returns, the portfolio outperforms the stock with Sharpe ratios of 2.1 and -1.1, respectively.

The analysis of these trades occurs over the testing data, which means the model is operating on data it has not been trained on. According to this representation of how the model can be utilized, it appears that a successful trading strategy is possible even with the formerly stated flaws in model performance. The above results only show the results of two stocks. When run over the entire list of approximately 500 stocks, the results are slightly more revealing.

The average portfolio return during the testing time period was -0.02439911804, or roughly a 2% decline in portfolio value. This was while the market increased an average of 0.00989090412063, or an average 0.9%

increase over all stocks in the list used in this project. There was also a poor average Sharpe ratio negative 0.176117617031, while the overall average Sharpe ratio was 0.283468337271.

Reflection

The process of developing this program began with retrieving the data from Yahoo Finance. After that, there were many potential indicators to explore, but the information that proved most useful was the information almost immediately provided by the data retrieval. The data needed to be processed so that it was on the same scale given the dynamic nature of the market. This meant changing day-to-day prices to percentage changes. Given the processed data, predictions were improved by observing the combination of two regression models, Linear and k-Nearest Neighbors. While results occasionally proved useful for certain stocks in a trading strategy, the overall value of these predictions does not appear to be positive. he results did not immediately appear useful, when observed in the context of a trading strategy, the results proved that they could function as a method to achieve greater returns than the market in the instance explored in this project.

Unfortunately, while a great amount of time was used to explore the effectiveness of indicators, none proved to be very useful. This is not to say that indicators do not serve a purpose in a trading strategy, but they were not useful for predicting future returns. The most difficult aspect of this project is how noisy the returns are well known to be in the short and long term. It was difficult to find meaningful features that improved the accuracy of the model.

As a disclaimer, stock trading is a very volatile way of making money. In other words, it is just as likely that a trader will lose money, as a trader will make money. In short, stock trading can be very much like gambling, and all investor can engage in trading at his or her own risk.

Improvement

There are a myriad of improvements that can be made to this product. There are hundreds, if not thousands, of technical indicators used in inter- and intra-day trading. There is continuous analysis and debate over what indicators are most effective, just as there is the same analysis and debate over which models are most effective in this realm.

In the same vein, sentiment analysis can be explored as a feature to better predict stock returns. There is not enough storage space or computing power on one computer to explore that effectiveness in a reasonable amount of time. Information from a large number of sources would need to be stored over time to explore sentiments effectiveness in predicting returns for even one company.

Another way to enhance the usefulness of this program is to use it in conjunction with a reinforcement learner. A reinforcement-learning agent that can adjust to learn an optimal trading strategy and maximize returns can control the trading strategy.