ESE 499/400

Data-Driven Equity Performance Forecasting:
An Effort Towards Sound Portfolio Management Using Predictive Models

John Fordice
Bachelor's of Systems Engineering
fordice@wustl.edu

Fall Semester, 2018

Washington University in St. Louis
Department of Electrical and Systems Engineering

Advised by:

Dr. Jason Trobaugh, jasont@wustl.edu
Dr. Zachary Feinstein, zfeinstein@wustl.edu

Submitted: December 15th, 2018
Abstract:

This project sought to use rudimentary machine learning techniques to provide sound investment advice to the retail investor. Data on all actively-traded equities, sampled weekly from 2014 to 2018, was pulled from a Washington University Bloomberg Terminal. A random forest algorithm was run on this data to provide insight such as feature importance and ‘buy’ recommendations on individual equities. A k-means clustering algorithm was then be run to determine whether there is a particular subset of equities which was predicted to perform will. The results of this analysis are inconclusive. The random forest yielded unrealistically high prediction accuracy for test data and the K-means clustering yielded no discernable insight.
1. Introduction

1.1: Equity Forecasting History

Historically, the concept of forecasting future equity market prices has had mixed reception. According to Burton Malkiel, an acclaimed mid-20th century and economist, stock prices follow a “random walk”, implying that the price from market open to market close could not be accurately predicted to generate alpha. In finance, alpha refers to the amount by which a portfolio out-performs a standard index such as the Standard & Poor’s 500 Index (S&P500). In accordance with the efficient-market hypothesis (EMH), which states that market prices incorporate all publicly available information and analysis faster than any individual could exploit an opportunity for arbitrage, many in the industry say that predicting future prices of equities is a shell game; it is a lost cause. However, Burton Malkiel, and indeed the EMH itself, have industry critics who posit that the market behaves irrationally all the time. John Maynard Keynes, a universally respected economist active in the first half of the 20th century, is credited with the ubiquitous saying: “The market can stay irrational longer than I can stay solvent.” In some ways, the strategy proposed in this project could be said to take advantage of that irrationality, though the motivation is to enable investors rather than exploit them. After preliminary research it is clear that there is no open-source software suite that delivers a data-driven outlook on the equity markets.

1.2: Current Available Analysis

While the route proposed in this paper is extremely simple, there simply are not many unbiased, data-driven analytics reports available to the average retail investor.
There are platforms such as Quantopian which allow anyone with the programming background to conduct their own research on past data, but most retail investors do not have a programming background which prevents them from accessing the insights which could possibly be derived from it. A dashboard like the one proposed allows an investor to get a perspective on the forward trends of the market on a week-to-week basis unfettered by institutional bias.
1.3: Objectives

The goal of the project is to create an easily useable data pipeline which can generate ‘buy’ recommendations for retail equity investors investing in the stock market. The average retail investor does not have a background in fundamental analysis, that is, analysis which requires knowledge of financial statements or macroeconomic trends. By using only historical data on technical indicators such as AZS health grade, relative strength indices, and others, one could reasonably expect a sound recommendation to be returned from a logistic regression algorithm, which is form of statistical, or “supervised”, machine learning. In this case, a random forest algorithm will be used. This is essentially reducing the problem of picking well-performing equities to one of finding the highest correlating indicators leading to that performance. While the implementation of such a statistical learning algorithm is non-trivial, this should be thought of as a highly rudimentary approach to the use of machine learning in equity forecasting. Upon successful implementation of the random forest algorithm, significant time will be spent tuning what are called the hyper-parameters of the model. A K-means clustering algorithm will then be implemented to see whether there is a certain group of equities performing well or poorly, which should impart a big-picture view of what the markets are favoring or opposing. All historical data will come from the Bloomberg Terminal service, and all processing will be done in Spark or Python, depending on availability of distributed computing resources. The goal will be to predict equity performance within a week from each prediction. All of these data pipelines will automated as much as is feasible, and visual reporting should be presented in an easily digestible Tableau dashboard. A weekly visual report detailing the market’s favorite
category of stocks along with the performance of those stocks will be generated using the outputs of both supervised and unsupervised learning algorithms. The generation of this report will be automated as much as is feasible within the timeframe. Currently, this means an attempt to pull new data from a Bloomberg Terminal on a weekly basis which will then be imported to a Microsoft Azure database. The data will then be processed in a spark environment. I anticipate a challenge in automated data pulls from the Bloomberg terminal. The data available from Bloomberg is a product for which institutions would normally pay tens of thousands of dollars. Because of this, Bloomberg makes it very difficult to pull data using anything besides their proprietary API. I hope to find a workable solution in the coming weeks, otherwise I will have to resort to manual data acquisition.

Reporting of results will be a large component of the project. There are simple steps one can take to move outputs of data from a Databricks workbook to a Tableau dashboard for quick visualizations. Assuming a project like this was to be productionized, the weekly report would be read by those seeking to gain a bias-free outlook on the markets for the next week. This demographic would most likely be comprised of retail investors who employ many of their own trading strategies already and are simply looking for a fresh perspective.
2. Methods

2.1.1 Machine Learning – Random Forests

Radom forests are an example of *supervised learning* in that they operate based on labeled data that follows a rigid structure. Classification algorithms, which will be the main function of random forests in this experiment, operate on target variables. A supervised learning algorithm will train on a set of features and a target variable, which can be either categorical (binary in the case of this project) or continuous. A typical use case of supervised learning is to predict the outcome of an instance of the target variable based on the features for that particular instance.

The random forest algorithm was chosen for this project for two reasons: first, it is remarkably versatile in that it has a tendency to work with a small or large number of data features (which can be thought of as columns in a table) without overfitting to the training set\(^1\). Second, I have implemented them for similar data in past projects, so the learning curve will be minimal.

The primary building block of the algorithm is a decision tree. Decision trees in and of themselves are a widely used statistical learning tool. They can be used for both *regression*: predicting a continuous dependent variable, and *classification*: predicting a categorical dependent variable. They work by taking in all variables (columns) and records (rows), and predicting the output of a target variable through a series of splits:

---

\(^1\) Louppe – Understanding Random Forests: From Theory to Practice
A drawback of using singular or small numbers of decision trees are that they are prone to overfitting. The term ‘overfit’ is used to describe a situation in which a model is too specific to the training data and can no longer usefully analyze new datasets. Random forests are useful in that they combine many decision trees to avoid this, hence the term ‘forests’. If a model has $m$ input variables, each decision tree in a random forest will select $n < m$ variables randomly and perform splits based on these $n$ variables. This is done as many times as the developer specifies with more trees generally yielding greater model accuracy. Overfitting is avoided through the process of aggregating the resulting splits of all trees through a ‘voting’ process. Another advantage of random forests is that they can handle relatively large amounts of data while still yielding fast and easily digestible results.

2.1.2: Machine Learning – K-Means Clustering
K-means clustering is an *unsupervised learning* algorithm, meaning it operates on unlabeled, unstructured data with no training required. The algorithm treats each feature or column of the dataset as an axis and is essentially concerned with finding \( k \) groups, or clusters, of points within a relatively small \( n \)-dimensional distance of each other. Here, \( n \) is the number of features or columns of a flat dataset. The algorithm starts by randomly selecting a data point \( c \) with a radius \( \varepsilon \) surrounding the point:

![Figure 2: A stepwise visual representation of a two-dimensional k-means clustering algorithm.](image)

As the radius shifts in each step to include more data points, so does the center of the cluster. The radius is shifted iteratively until no more points can be added to the cluster as a result of shifting. This condition is commonly referred to as ‘convergence’. The number of clusters the algorithm seeks is determined by the developer. The most

---

2 Lavrenko – University of Edinburgh
common way to interpret the results of clustering is to output the cluster centers. Combined with the original column labels, the centers, along with the relative size of each cluster, yields a great deal of insight. In the case of this project, it might reveal that defense stocks are performing well while holding a high forward-momentum indicator. It is easy to see how this information would be valuable as one watches the market develop over the course of the week. Tangentially, this can be thought of as a simplified version of an investing strategy known as factor investing. This term is usually put under the umbrella of ‘quant’ (quantitative) investing. Firms like AQR Capital Management, Two Sigma, and Renaissance stand out as reputable players in the industry employing this class of strategies.

2.2.1: Data Collection – Bloomberg Terminal

Bloomberg’s Terminal service was used to collect the data for this project. Using a Bloomberg Excel add-in, users are allowed to make a certain number of API calls per month. In the context of this project, one API call equates to pulling data for a certain field for a certain security at a certain date. As this project uses over 2500 listed company stocks and a range dates, this resulted in over 650,000 calls per field. As Washington University’s license to Bloomberg’s service is not of the same grade as that of an institutional investor, the school’s monthly limit was hit multiple times in the process of data collection. Many thanks to Madjid Zeggane (Washington University, Database Analyst) for his help in resolving this issue. The fields used for analysis can be seen in the table below:
<table>
<thead>
<tr>
<th>Metric</th>
<th>Time Period</th>
</tr>
</thead>
<tbody>
<tr>
<td>Call-Implied Volatility</td>
<td>Trailing 5 Days</td>
</tr>
<tr>
<td>Average Return</td>
<td>Trailing 5 Days</td>
</tr>
<tr>
<td>Share Volume Traded</td>
<td>Trailing Day</td>
</tr>
<tr>
<td>Call Volume</td>
<td>Trailing 5 Days</td>
</tr>
<tr>
<td>Put Volume</td>
<td>Trailing 5 Days</td>
</tr>
<tr>
<td>GICS Industry</td>
<td>NA</td>
</tr>
<tr>
<td>Altman's Z-Score</td>
<td>NA</td>
</tr>
<tr>
<td>Alpha</td>
<td>Trailing 3 Months</td>
</tr>
<tr>
<td>Beta</td>
<td>Trailing 3 Months</td>
</tr>
<tr>
<td>Relative Strength Index</td>
<td>Trailing 30 Days</td>
</tr>
<tr>
<td>Relative Strength Index</td>
<td>Trailing 3 Days</td>
</tr>
</tbody>
</table>

Unfortunately, due to the proprietary nature of Bloomberg’s data, it is infeasible to automate the data-collection process to occur on a weekly basis independent of analyst input while operating on a university license.

2.2.1: Data Processing – Reformatting for Analysis

Due to the way Microsoft Excel uses memory, it is not feasible to pull data in a long datasets. In the case of relational datasets, long refers to a dataset having a large ratio of rows to columns, while wide the inverse. Because most computers available to students do not have the amount of RAM required to solve this problem using brute force, a wide format must be used when pulling data. If only using the add-ins provided by Bloomberg, one is best served pulling data in a matrix format – dates as rows and
security tickers as columns – for each field used. In order to analyze this data using random forests and k-means clustering, a Python script had to be written which turns each matrix representing a particular field into a single column. If all input data is in a directory organized by field in one matrix per field in csv format, the script can automatically merge all of these csv files into one file in the correct format. It is important to note that this long schema is the format one would find in a traditional relational database.

2.3.2: Data Processing – Microsoft Azure Databricks

Microsoft Azure and Amazon Web Services are competing providers of cloud services like data warehousing and distributed computing. This project will make use of Microsoft Azure due to Microsoft’s established relationship with Washington University. Azure also offers an integration with Databricks, the premier business analytics distributed computing platform. Databricks charges in terms of Databricks Units (compute nodes) per hour. With help from Mark Bober of Engineering IT, a workspace on Washington University’s Azure account was set up. So far, the need for additional funding from the ESE department has been avoided.
2.3.1 Analysis – Random Forest (Project Context)

The brunt of the predictive work accomplished by this model will be done by implementing a random forest algorithm to determine which indicators are the most important for any given week. I predict that in times of high market uncertainty, such as the recent episode of decline in technology firm stocks, indicators like volatility and call/put-to-volume ratios will be key. In the case of stable bull markets, I expected these important indicators to be average volume and number of call contracts bought, which would agree with traditional thinking.
There are many possible performance thresholds one could set for determining whether the binary success variable for training will be a one or zero. For instance, one could create a calculated column from past data called ‘performed’ where the record indicates a 1 (performed) if the average trailing five day return was over 3% and a 0 (did not perform) if otherwise. In this case, a predicted ‘1’ for a given equity by the random forest indicates that the price is expected to rise by at least 3% one market week from the time the prediction was made. This is a simple approach which is attractive because of its ease of implementation. I expect I will need to test multiple thresholds in the hyperparameter-tuning phase to see which yields the most accurate results.

The random forest will not only reveal which indicators were most important in the previous week of trading – it will also predict the binary target variable of ‘performance’ for the next week, which results in ‘buy’ recommendations. After this was implemented, the idea was posed predict the actual performance of individual securities in terms of percent returned.

2.3.2 Analysis – K-Means Clustering (Project Context)

Unsupervised learning is primarily used as a tool to approach datasets with which an analyst or institution has no familiarity. It serves as a first step to see if there are any analyzable patterns within the data. This project does not employ it as such – its output will serve as increased contextual awareness for a hypothetical small-scale investor for when they consume other financial analytics. Many research teams have attempted this in the past, which is why it is in the workflow of the project\(^3\).

\(^3\) Suganthi, Kamalakannan2 - An Approach to Analyze Stock Market Using Clustering Technique
K-means clustering is commonly evaluated via sse (sum of squared error). In the context of clusters, this is the sum of the squared differences between each observation and its group’s mean, the group’s mean being the average distance of all points in a given cluster from that cluster’s centroid. For one dimension, this is written mathematically as:

\[ SSE = \sum_{i=1}^{n} (x - \bar{x}) \]

(1)

Multidimensional SSE is calculated using the Euclidian distance. SSE is a useful metric for evaluating the validity of our clustering results. For evaluation on clusters whose data points hold more than three dimensions, one has trouble using visual inspection to evaluate the results. This is where SSE is helpful – a relatively high SSE can help us to see how ‘tight’ or ‘loose’ a cluster might be. For reference, if all data points in a cluster were identical, then SSE for that cluster would be zero as the terms in (1) would be identical. Therefore, it is valid to think that the more diverse the data points in the cluster, the higher that cluster’s SSE will be.

The output of the K-Means clustering algorithm will ideally be a set of stocks which were predicted to perform well by the random forest in the next week. As K-Means is an unsupervised learning method, this is not guaranteed. Regardless of the interpretability of the output, the cluster which is predicted to perform well (if it exists) will be reported on the same dashboard as the buy recommendations. The consumer of this report would then be able to take information about the markets from more traditional sources like the Wallstreet Journal or Bloomberg Media in context. One could say this method of reporting provides a bias-free outlook on the markets.
SSE is also useful in determining the number of clusters (the $K$ in K-means) one should seek to find in the dataset. This can be done by running a *compute cost analysis* on the entire dataset to which one wishes to apply K-means. Compute cost as a term is often interchanged with SSE and is the name of the Spark method used to measure it. In our case, this is done using the “elbow method.” The idea behind this method is to choose a $k$ (the number of clusters) such that the model would not perform significantly better if it were any higher, thus giving a favorable tradeoff between information extracted and computation time\(^4\).

![Image](image.png)

**Figure 4:** Sample plot demonstrating elbow method

Note the inflection point of the curve in figure four – by the elbow method, three would be the optimal value of $k$ to use when applying K-means to this dataset. Recall

that K-means is an iterative algorithm that goes until it reaches convergence. Thus, the tighter the tolerances of each point relative to the cluster centers surrounding it, the more iterations it will take to reach convergence.
3. Results and Discussion

Initial results from the random forest have shown signs of rampant overfitting. This can be diagnosed by simply taking comparing the number of correct predictions to the number of incorrect predictions. Achieving 0% error is a clear sign, which is was present in the first set of results across all numbers of estimators (trees) ranging from 1 to 1001. This was due to multiple factors: Firstly, the initial random forest model ran on the entire dataset rather than weekly chunks of data. Secondly, the indicator `perc_chang_5d`, which describes the price movement of the security in the past week, was included in the initial run as a feature. This oversight led to a faux predictive nature of the model – if the parameter on which the label is based is present as a feature to be observed, there is nothing to predict, as the data is already there. After fixing these mistakes, the resulting drop in model accuracy was encouraging.

![Figure 5](image)

**Figure 5**: A model trained on 52 weeks of data from 2014 predicts weekly recommendations in 2015.

This level of accuracy is still unrealistically high – predicting a three percent return week with an average accuracy of over 80% is unheard of even among the most advanced hedge funds. The implications of a model like this functioning in the real world
would mean that a individual's portfolio could theoretically never perform worse than the market, and in fact, could only ever beat its return. See the appendix for weekly predictive accuracies over additional years.

There are multiple possible explanations for this behavior. Relative strength indices (RSI), both trailing three and thirty day variants, are used as features of the dataset. RSI is a measure of price momentum between 0 and 100 with sideways movement (no increase or decrease) being 50. The greater the increase in price over the given time period, the higher the security’s RSI. The inclusion of this feature could be affecting the model accuracy in the same way the inclusion of the perc_chang_5d, as its value is related to increase or decrease in a securities price. It is, in a sense, still including the information provided by perc_chang_5d. Indeed, when we examined the feature importance ranking for a model trained on data from 2014, our suspicions are confirmed:

<table>
<thead>
<tr>
<th>Feature</th>
<th>Importance</th>
</tr>
</thead>
<tbody>
<tr>
<td>rsi_3d</td>
<td>76.16%</td>
</tr>
<tr>
<td>rsi_30d</td>
<td>8.93%</td>
</tr>
<tr>
<td>px_last</td>
<td>7.12%</td>
</tr>
<tr>
<td>day_shares_volume</td>
<td>5.14%</td>
</tr>
<tr>
<td>alpha</td>
<td>1.27%</td>
</tr>
<tr>
<td>beta</td>
<td>1.01%</td>
</tr>
<tr>
<td>azs</td>
<td>0.29%</td>
</tr>
<tr>
<td>num_call_contracts</td>
<td>0.04%</td>
</tr>
<tr>
<td>call_imp_vol</td>
<td>0.03%</td>
</tr>
<tr>
<td>num_put_contracts</td>
<td>0.02%</td>
</tr>
</tbody>
</table>

We see that the RSI trailing by three days is far and away the most important feature, while the second most important feature is the very same metric, different only in the period over which the metric is calculated. It follows logically that the three-day variant
of the metric is more important than the thirty-day variant, as the measure which determines the label trails by five days. All other features aside from the last closing price and trading volume are overshadowed to the point of not being relevant in the predictive model. In statistical terms, these top three metrics would be considered the principal components of the dataset.

There are also interesting implications of the lack of importance given to the number of call and put options bought. A call option is a contract giving the holder the right to buy a given amount of shares at a given strike price. The value of this contract increases as the security’s actual price gets closer to or exceeds the strike price, and it can be sold before expiration if the holder does not wish to exercise the option. It is essentially a bet that the price of a security will rise to a given price by a given time. A put option is similar contract giving the holder the right to sell a given amount of shares at a given strike price. It can be thought of as a bet that the share price will decrease and is akin to a short position. The amount of call and put options bought for a given security are often used as a measure of market sentiment. If investors believe the stock price will increase, this is reflected in a higher number of call options bought. Inversely, a higher number of put options bought reflects a that investors believe the share price will decrease. It is interesting that these metrics are so unimportant to the predictive model, as investor sentiment is exactly what this predictive model was meant to disregard. This idea should be taken with a grain of salt, as it is unclear where these metrics would fall in terms of importance without the dominating presence of RSI metrics.
Another possible explanation for the unrealistically high accuracy is that the markets have remained somewhat consistent over the past four years. The 2014-2018 run has been one of the strongest bull markets in recent history. It is naïve to say that the historical span and scope of the metrics gathered for this project perfectly encapsulate all market forces and conditions, but there are still interesting ideas to be explored. Momentum trading is school of trading strategies which aims to take advantage of market conditions like those seen in the past four years. It relies heavily on data from metrics such as RSI, which is a direct measure of price momentum, as predictive model apparently does. The most salient downside to momentum trading is that it does not respond well to crashes following a period marked by consistent upward trends. This is relevant to this model because it was not tested on data from late October and November of this year. These two months have hosted the largest market downturn since September 2011 when the country was still recovering from the fallout of the 2008 financial crisis⁵.

In terms of market performance, the predictive model was set to train on data from 2014 and run against weekly datasets from subsequent years, ending in early October 2018.

⁵ AP News - October marks worst month for US stocks in 7 years (October 31, 2018)
The above result was obtained by using a calculated label of 1 for weeks in which a given security attained a threshold 3% or greater return, and 0 otherwise. Again, we see unrealistically high performance against the market. The advised portfolio contains almost zero poorly performing stocks as they are filtered by the predictive model. This is a direct result of the unrealistic accuracy, likely due to the inclusion of the RSI metrics in the training data. This behavior is amplified when we increase the threshold from 3% to 10%. As it is known that the model will perform unrealistically well, we expect the disparity in returns between the recommended portfolio and the overall market to be even larger, which is, indeed, the case:
The average return is much higher given a higher return threshold. See appendix for additional results comparing recommended portfolio to overall market.

This was followed by treating the resulting of predictions using K-means clustering. In order to carry out clustering, it was necessary to use the elbow method to know which value of k would yield the most interpretable results. Running the compute cost method on the entire dataset generated the following curve:
This does not result in a clear inflection point. This could imply that the clustering is not a suitable method for analyzing this dataset, which very well may be the case. This was not the end of the project as the objective was not to cluster all the available data at once, but to cluster weekly slices of it. The first week of data from 2015 generated the following SSE curve:
The curves for ten other weeks looked similar with a sharp decline in SSE tapering off after $k = 5$. Running K-means on this data yielded the following cluster centroids:

<table>
<thead>
<tr>
<th>Cluster</th>
<th>performed</th>
<th>px_last</th>
<th>alpha</th>
<th>azs</th>
<th>beta</th>
<th>call_imp_vol</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.39</td>
<td>21.76</td>
<td>0.07</td>
<td>21.73</td>
<td>0.77</td>
<td>488.87</td>
</tr>
<tr>
<td>1</td>
<td>0.00</td>
<td>46.99</td>
<td>0.44</td>
<td>46.99</td>
<td>1.04</td>
<td>408915.17</td>
</tr>
<tr>
<td>2</td>
<td>0.20</td>
<td>30.23</td>
<td>0.15</td>
<td>30.23</td>
<td>1.13</td>
<td>112883.13</td>
</tr>
<tr>
<td>3</td>
<td>0.00</td>
<td>62.33</td>
<td>0.17</td>
<td>62.33</td>
<td>0.89</td>
<td>1330319.33</td>
</tr>
<tr>
<td>4</td>
<td>0.29</td>
<td>48.21</td>
<td>0.14</td>
<td>48.21</td>
<td>1.25</td>
<td>15243.73</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>day_shares_volume</th>
<th>num_call_contracts</th>
<th>num_put_contracts</th>
<th>rsi_3d</th>
<th>rsi_30d</th>
</tr>
</thead>
<tbody>
<tr>
<td>285526.08</td>
<td>488.87</td>
<td>332.11</td>
<td>41.20</td>
<td>3.32</td>
</tr>
<tr>
<td>39503413.70</td>
<td>408915.17</td>
<td>233298.00</td>
<td>48.07</td>
<td>4.08</td>
</tr>
<tr>
<td>14347228.80</td>
<td>112883.13</td>
<td>79009.13</td>
<td>39.29</td>
<td>4.01</td>
</tr>
<tr>
<td>62126447.00</td>
<td>1330319.33</td>
<td>726092.00</td>
<td>51.37</td>
<td>4.12</td>
</tr>
<tr>
<td>4582951.49</td>
<td>15243.73</td>
<td>9854.43</td>
<td>46.46</td>
<td>3.82</td>
</tr>
</tbody>
</table>

After running on many sets of weekly data, it was clear that K-means would not yield any of the insight on common trends that I had hoped it would. It was interesting to see that cluster zero contained most of the population:

If we examine the cluster centers, there is a possible explanation. Cluster zero is unique from the others in that its center indicates lower relative activity. Volume, number of call
and put options bought, and volatility are orders of magnitude lower than for all other clusters. This is likely due to the fact that, while a large number of securities are listed on the various exchanges, most are not actively traded.
4. Conclusions

The main findings of this project were that, when predicting price momentum, measures directly pertaining to price momentum will override all other features of a given dataset. I do not think this report contributes either to Burton Malkiel or his detractors in the debate of whether or not a security’s future price can be predicted. It is clear that further steps are necessary to glean information from this model. The first should be to gather results in the same workflow excluding the RSI, as it is highly possible that it led to a faux predictive behavior in the model leading to unrealistic levels of accuracy.

Early problems with model overfitting were solved as indicated by varying levels of high accuracy. If overfitting had been present, we would see far lower levels of accuracy reported as well as higher variance in outcomes.

In retrospect, this project would have been impossible without the use of the Databricks workspace. In terms of runtime alone, it has saved multiple days of man hours.
5. Deliverables

The status of deliverables for this project is a mixed bag. Data was extracted from a Bloomberg Terminal, preprocessed and analyzed in Python, and analyzed in a Databricks cloud workspace using Apache Spark. Both random forests and K-means clustering models were successfully implemented with somewhat unsuccessful results. In this way, the objective of providing the average retail investor with a data-driven alternative was also failed, but I believe I have made significant progress towards that goal. Results were visualized in Tableau and the dashboard is available on the course website. In terms of the automation, automated movement of data from Databricks to Tableau has been achieved. Unfortunately, due to the proprietary nature of Bloomberg’s data and daily data limits on educational licenses, automation was infeasible for that portion of the project, which made automation of weekly analyses infeasible as well. There is certainly a straightforward path through the Databricks command line interface and integration with a personal computer’s local filesystem, but, unfortunately, that will have to be explored by someone with access to an institutional Bloomberg License.
6. Timeline

[Diagram of a timeline with specific dates and events marked]
The above timeline was followed quite closely at the beginning of the semester. Unfortunately, due to an increase in job interviews late in the semester, the period in which I was supposed to be collecting results fell into the week before the due date of December 15th. Had this not been the case, additional results excluding RSI could have been collected and it is possible the model could have been salvaged for more interesting findings than the ones presented. I do not think that this was due to any mistakes in time management, rather it was due to there simply not being enough time.
Appendix

Model Accuracies

2014 Training - 2015 Predictions

2014 Training - 2016 Predictions
Comparison vs. Market

2016 Performance vs. Market - 3% Threshold

Average Return

-10% to 20%

Full Market | Recommended

Week

2017 Performance vs. Market - 3% Threshold

Average Return

-4% to 10%

Full Market | Recommended

Week
Tuning the Random Forest Model

Random Forest Tuning

Model Prediction Accuracy vs. Amount of Training Data

Impact of Number of Trees on Model Error

Model Error vs. Number of Trees