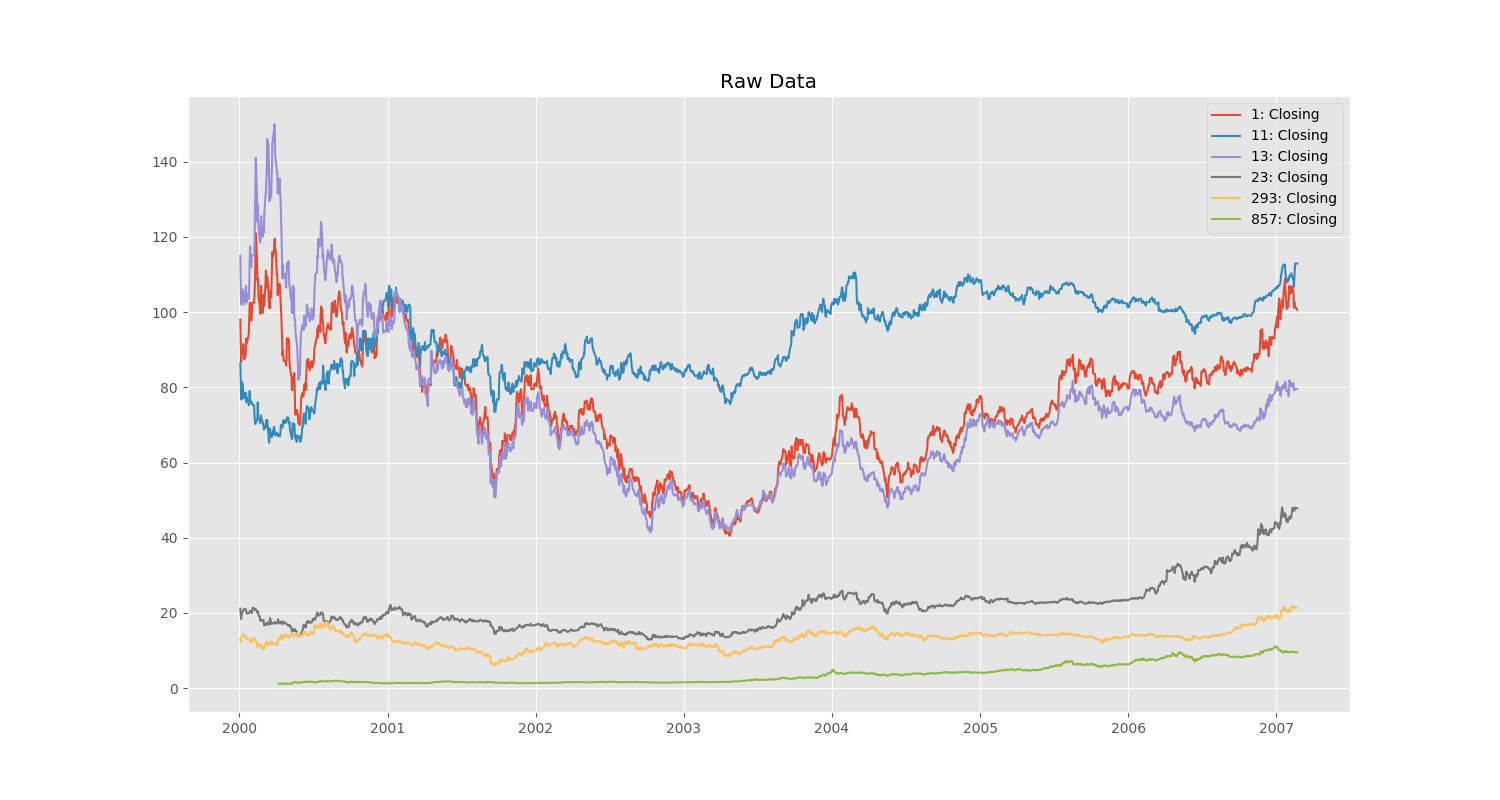
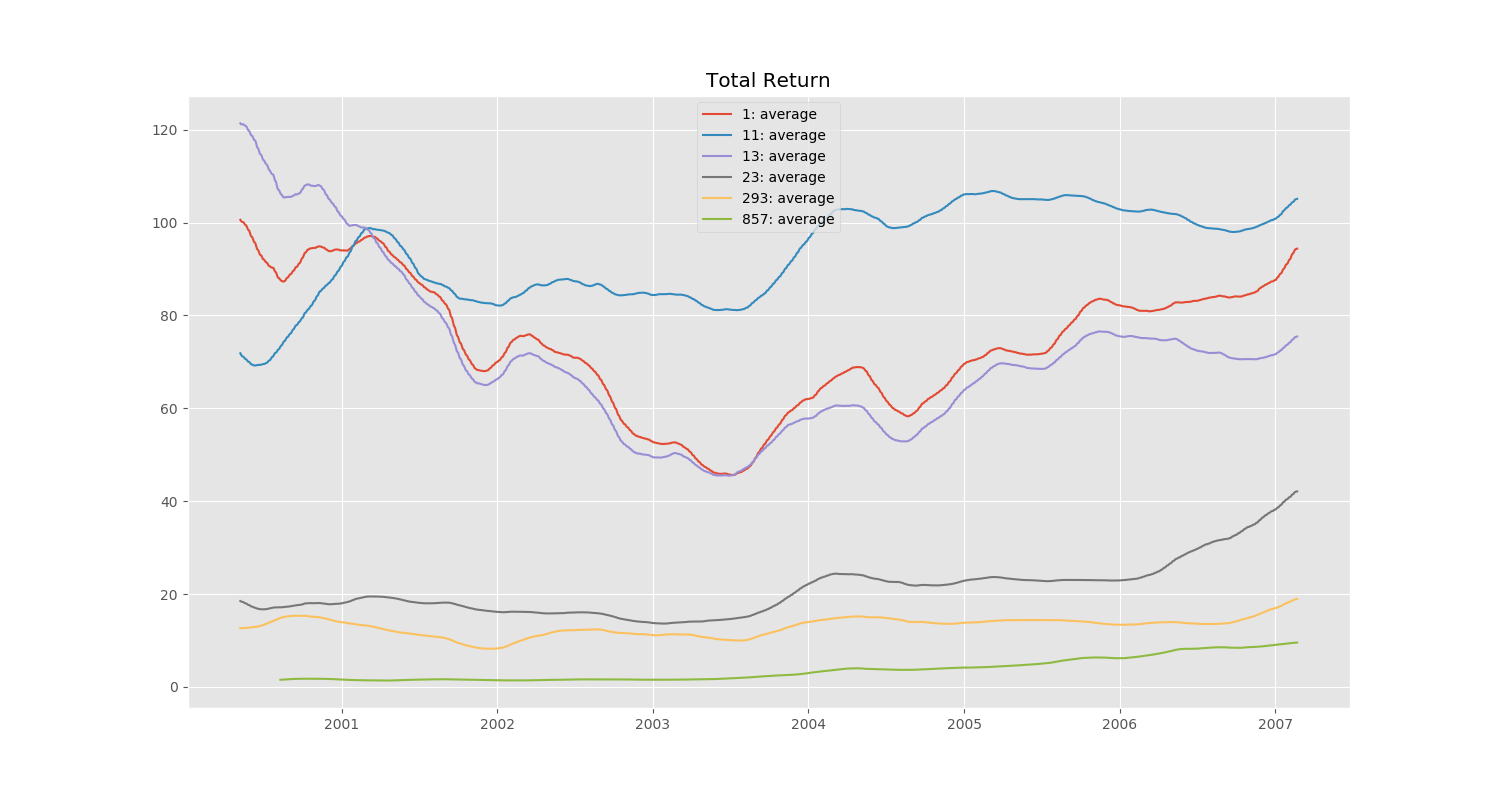
In this paper I will discuss the work I have done on using unsupervised learning on stock data. In the first section, I will discuss the pre-processing I applied to the data first. In the second section, I will begin to cover the types of unsupervised and supervised learning models I applied. The models I used are Kmeans and the Decision Tree classifier.

**Pre-Processing**

The data that we were given in this assignment was Hong Kong Stock data. We were given 6 stocks to work with and were given integer values instead of their actual names (i.e. 1, 11, 13, 23, 293, 857). In the csv/excel sheet we were given, the columns were given as “tdate”, “stock\_id”, “open”, “close”, “high”, “low”, and “volume”. However, the data was given in one huge column, so in order to extract the data properly, I used a python library called Pandas to extract all the information from the excel sheet. Once I had this data, I then grouped by “stock\_id” as the primary key in order to create a dictionary with “stock\_id” as the keys and all the other columns as a dataframe under each key. In order for the dataframe to function properly when plotted on a matplot plot, I had to reset the index for each of these dataframes too.

In order for the data to become useable, it was necessary to do a lot of data cleaning/editing. This can be seen when I plot the closing raw against dates as seen below:

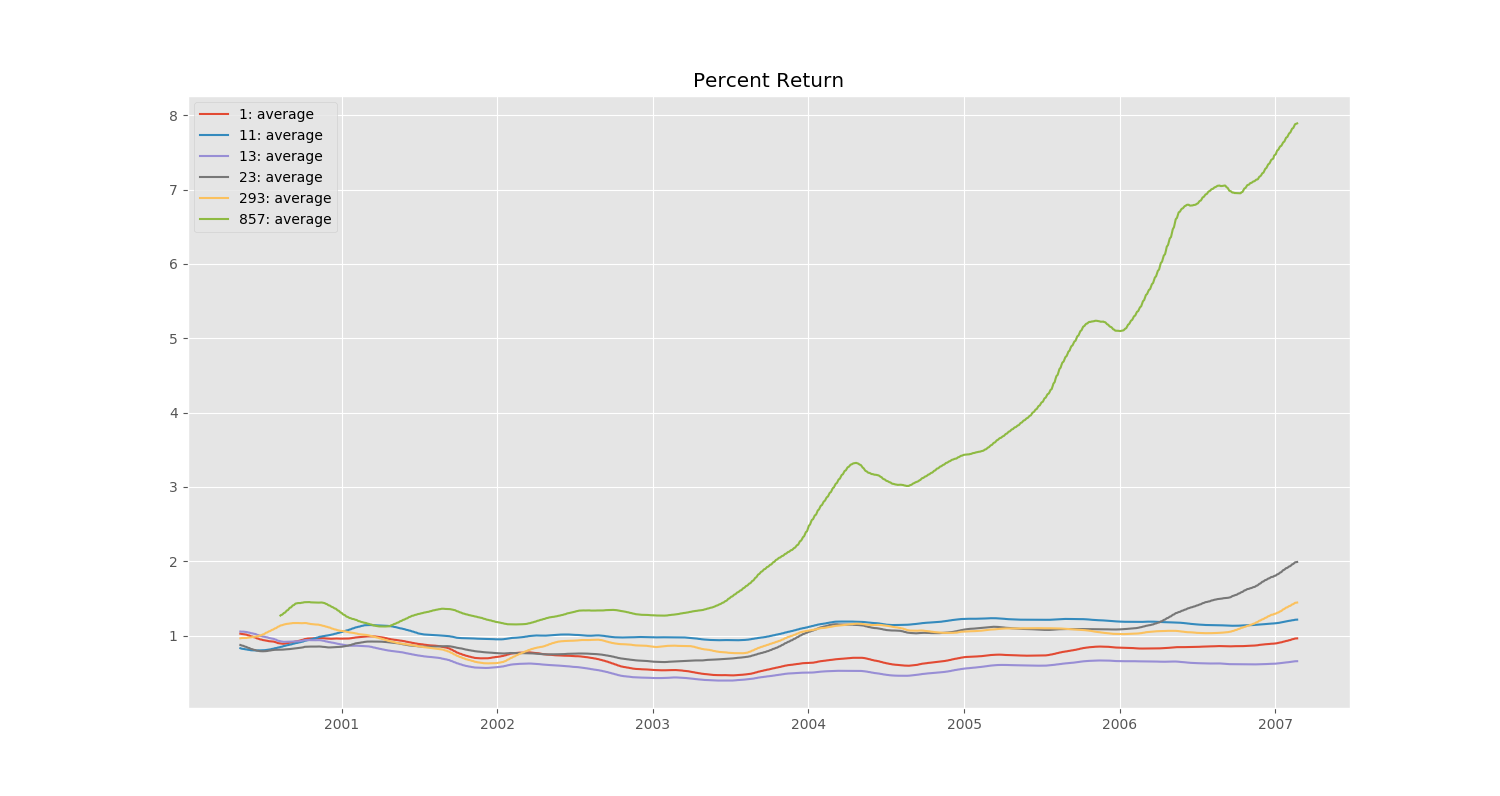
This data has too many ridges and as a result, the data would be much harder to process for good results (having all the ridges would cause a lot of models to overfit). Thus, in order to smooth out the data, I applied a rolling window of 90 in order to smooth out the data. This means that each point basically becomes the average price of 90 days. The reason for this number is also the fact that the year is generally separated into quarters in terms of finance, thus 90 gives a rough estimate into being 1/4th of the year. The resulting data is then as follows below:

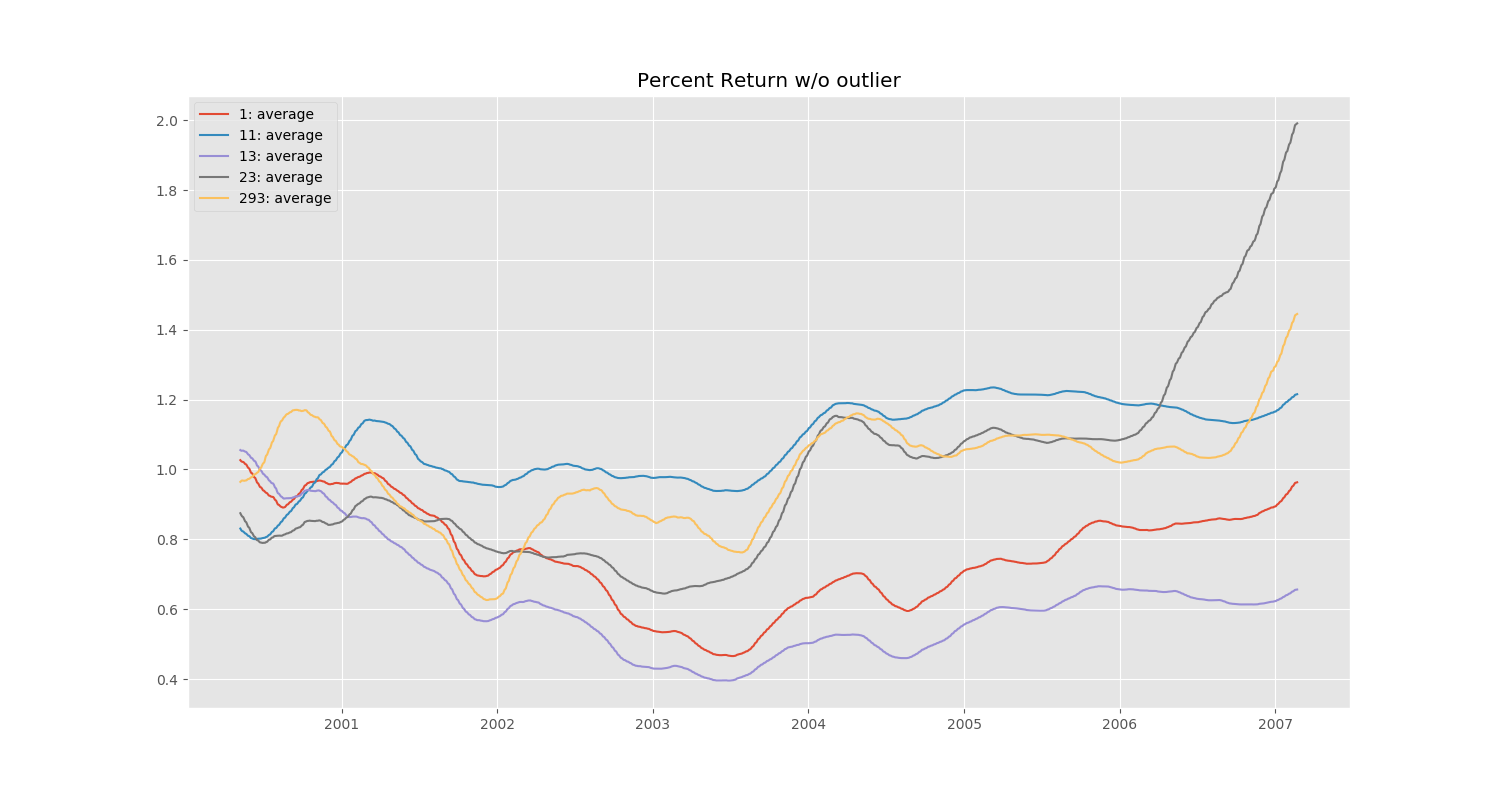


As you can see, although the data is now smooth, difference between each of the data points is too large to make sense of. Red, blue and purple are too large so the rest of the data becomes too small in comparison to see an actual trend. Thus, in order to make everything scaled relative to its original price, I calculated the percent return on the stock based on closing prices divided by the respective stock’s original price. This created the graph as seen below (“Percent Return”):

However, this data was still slightly hard to read due to one stock having a much higher return than all the other stocks, as a result of this, I decided to remove the stock 857 in order to more properly scale the results. The graph of the resulting plot can be seen under (“Percent Return w/o

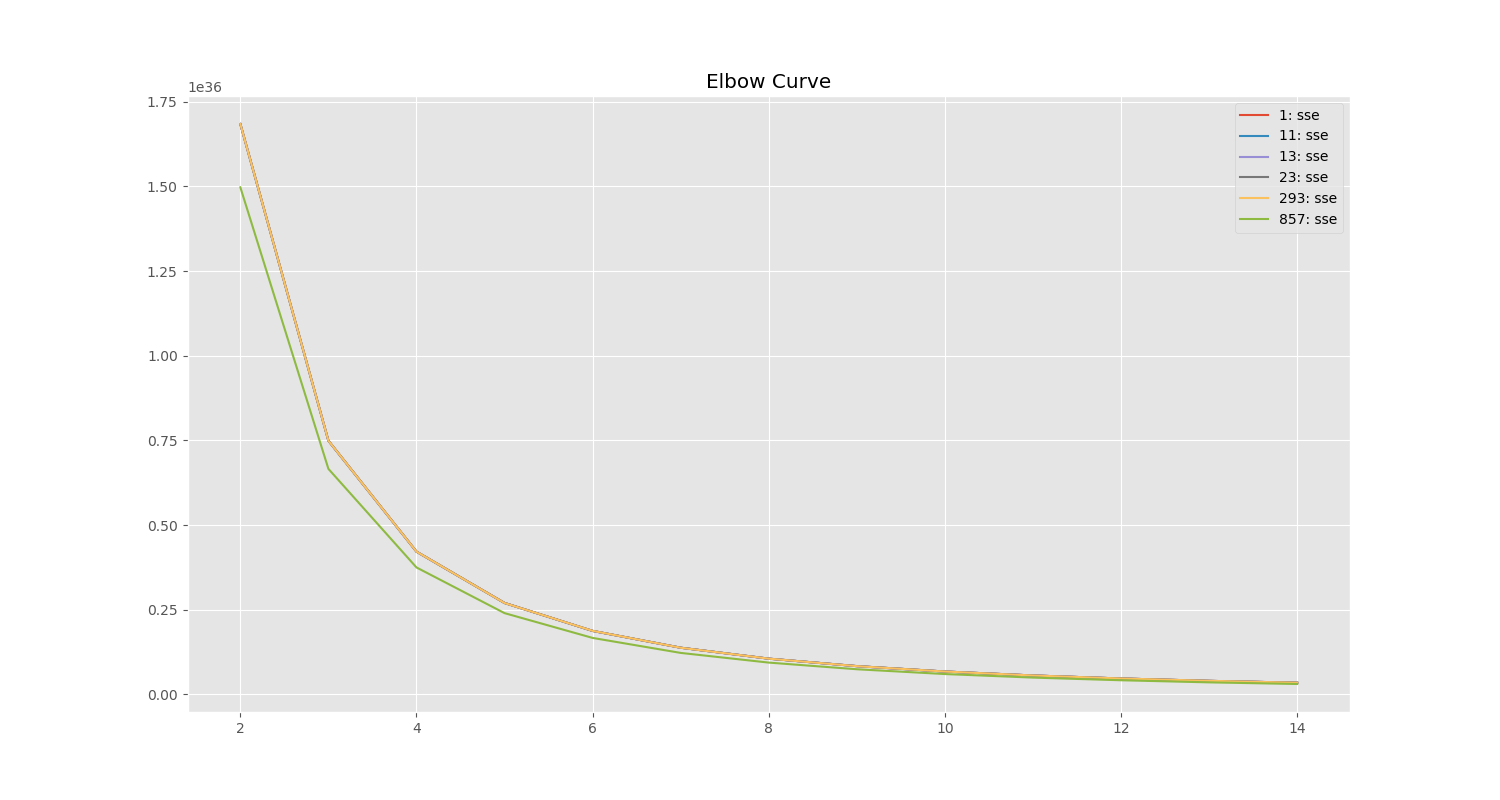
outlier”):





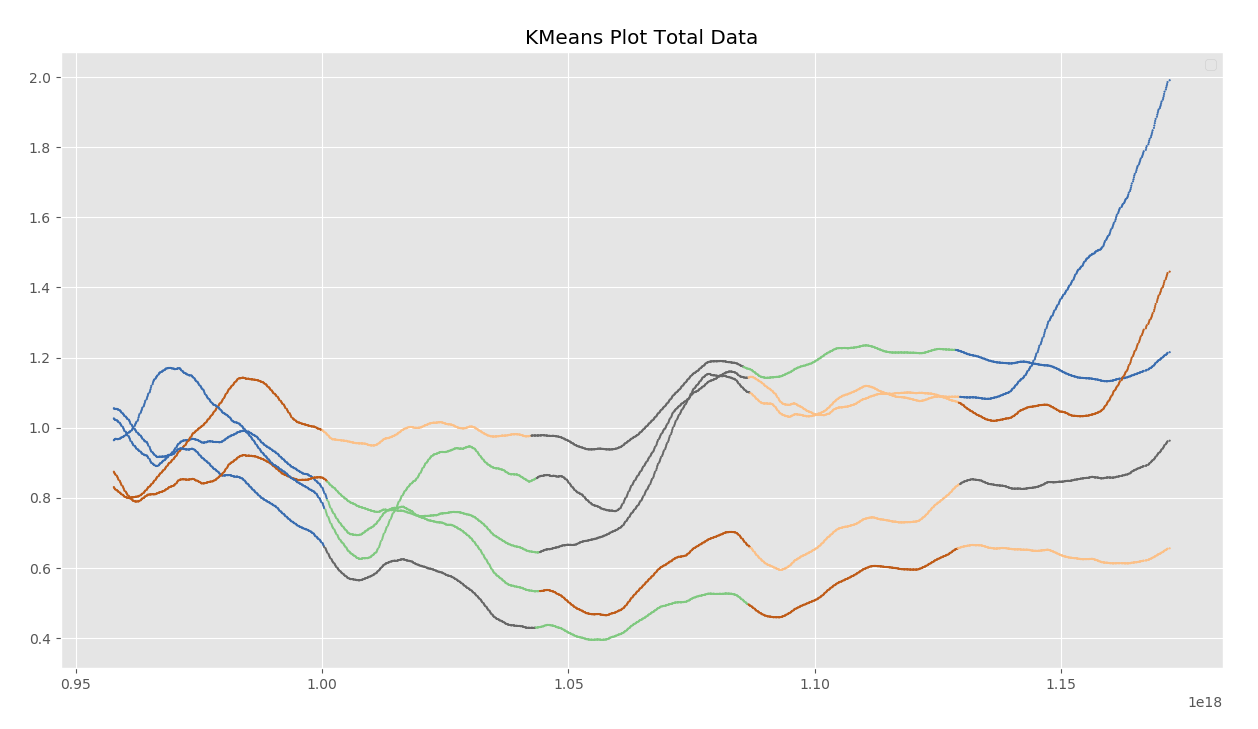
This data is much more readable than the others and shows more clear trends than before and scales the data properly according to starting price. Thus, from this point on, I then used the percent return to apply the unsupervised learning models.

**KMeans**

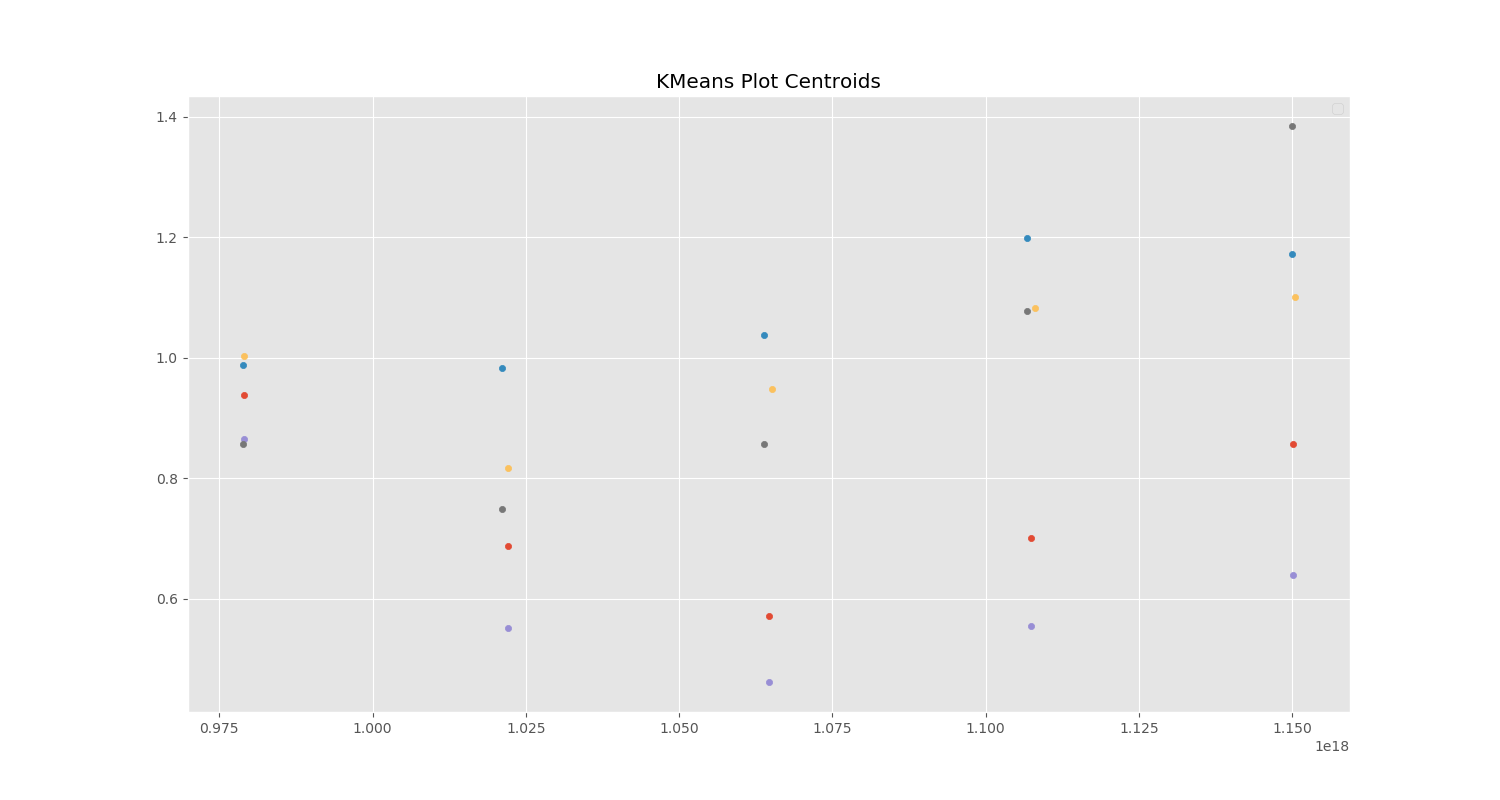
 After pre-processing the data I used the Percent Return data to apply Kmeans on the dataset. In order to find the optimal number of categories (or K) to apply the data to I then used Kmeans on a range from two to fourteen and then calculated Sum of squared errors within each cluster. The lower this value, the more accurate the classification of each cluster was. Here is a chart depicting the Sum of Squared errors which follows an Elbow Curve diagram (Sum of Squared Errors on the Y axis, and cluster count on the X axis):

As can be seen in this diagram, after 5 clusters, the error rate begins to smooth out. This signifies that at around 5 clusters, the model begins to classify things more accurately. The reason for not choosing more clusters is in order to avoid overfitting for this data/keeping the model simpel enough for us to interpret.

The next step was then to apply the KMeans model with 5 clusters, and when plotting these clusters onto a scatter plot (the dots make up what looks like a line) we get the following result:



Each color represents a different cluster in each model (same color does not mean part of same cluster if not in the same line), and thus it is clear that the KMeans algorithm classified all the data similarly with similar break points. To further prove this, I then plotted the centroids of each cluster on another graph:



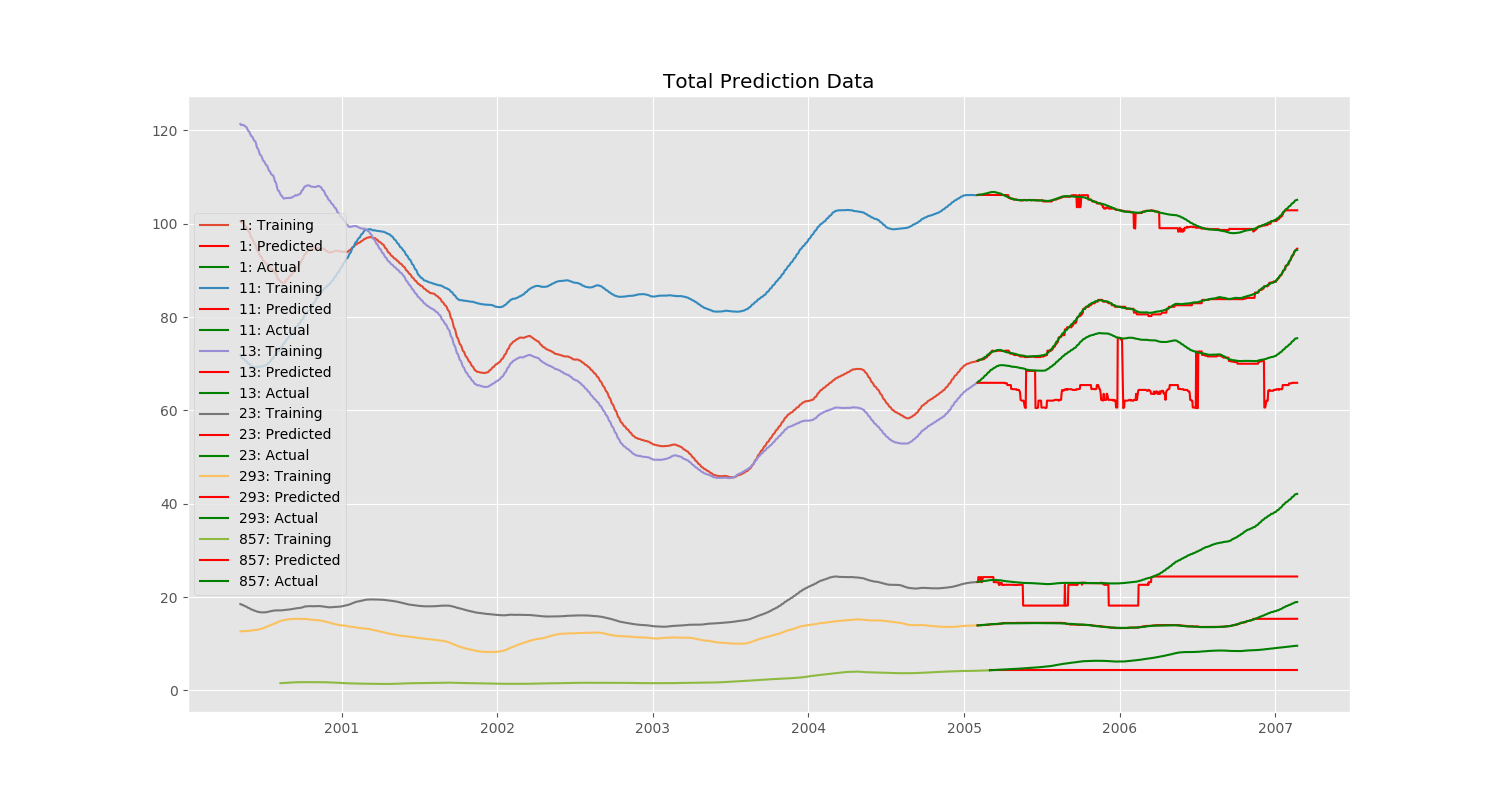
This helps to show how Kmeans classified the data in very similar intervals/segments based on the date of the stock and that the data for each of these stocks followed very similar trends.

**Decision Tree Classifier:**

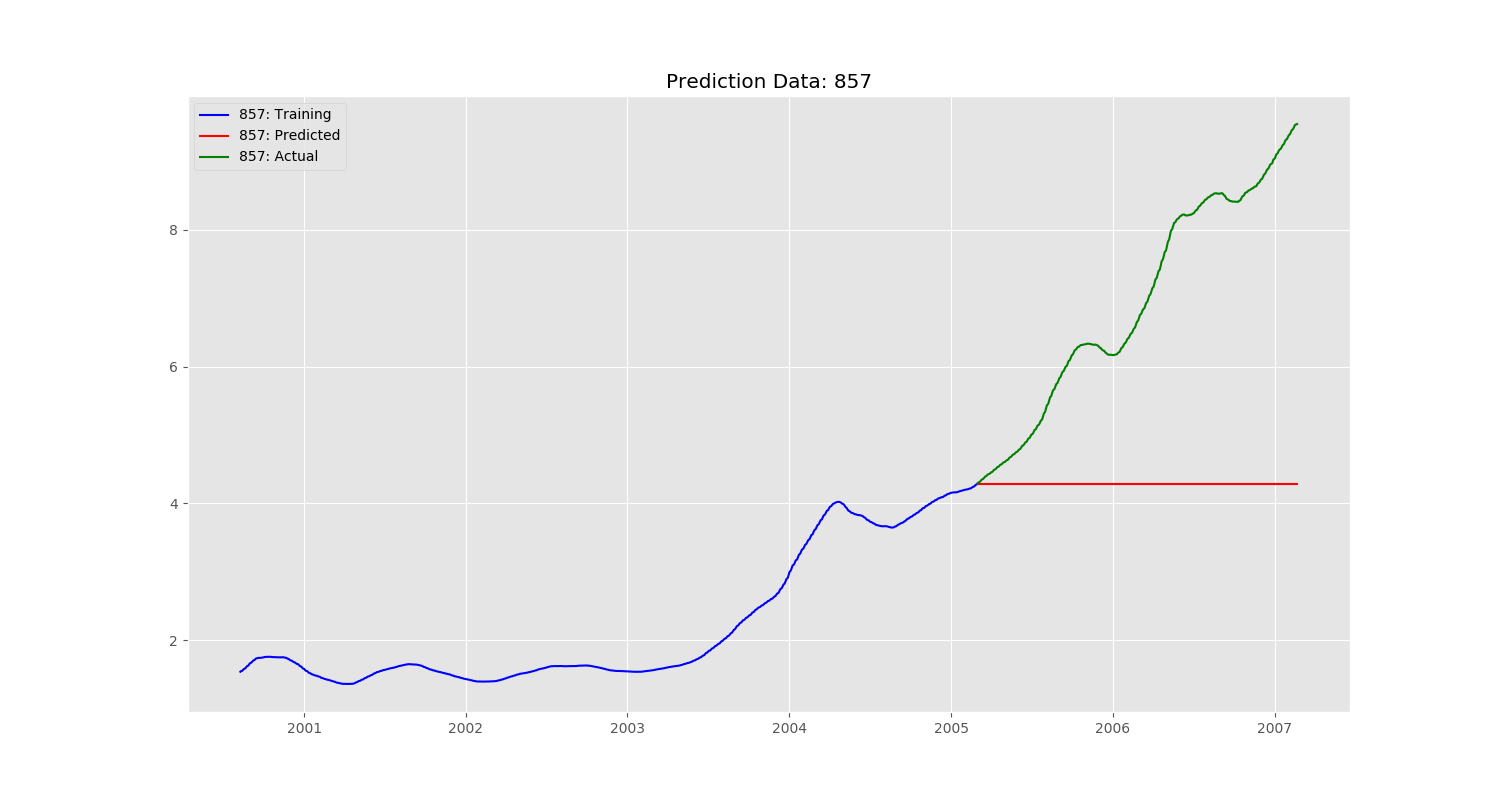
The Decision Tree Classifier can be considered part of ARM (associative rule mining) and a classifier. Thus, for our data it helps to classify/predict new stock prices based on association with previous values. The data that I used for this part of the project is the smoothed out data of stock prices with the rolling window of 90 days. I did not use the percent return as I was not comparing actual prices or using one model on another stock to predict the other.

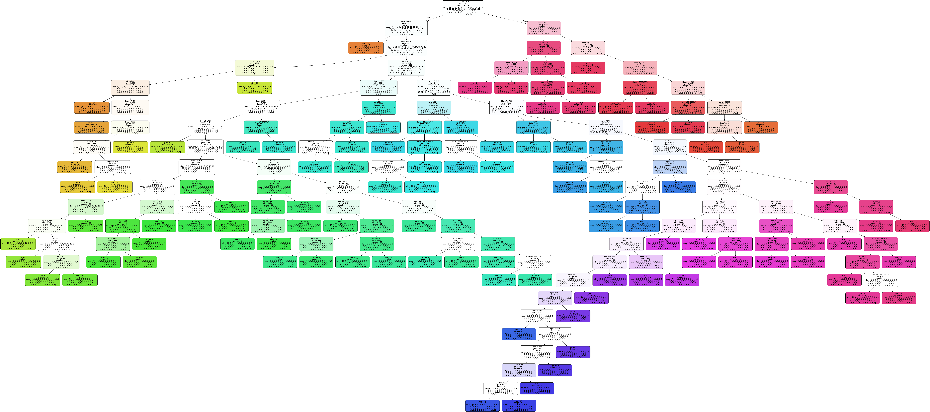
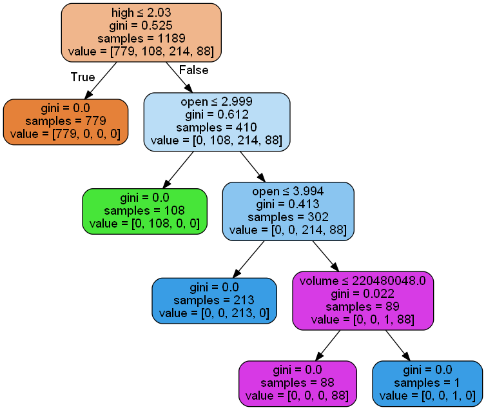
|  |  |
| --- | --- |
| ID | Accuracy |
| 1 | 99.79% |
| 11 | 99.44% |
| 293 | 98.41% |
| 13 | 91.22% |
| 23 | 84.18% |
| 857 | 65.52% |

The column that I was trying to predict was the “closing” value of the stock which I converted to integer types in order for the classifier to work. The columns that I used to predict this value were the “open”, “high”, “low”, and “volume” columns. I used 70% of the data for training and the remaining 30% for testing. The results in numbers can be seen to the left. The way that I calculated the accuracy was to take the absolute value of the difference between the predicted value and the actual value and then divide this percentage by the actual value. I then summed up all these percentages and divided by the length of the data in order to get the average. Below, I have plotted all the data for this project onto a single graph:



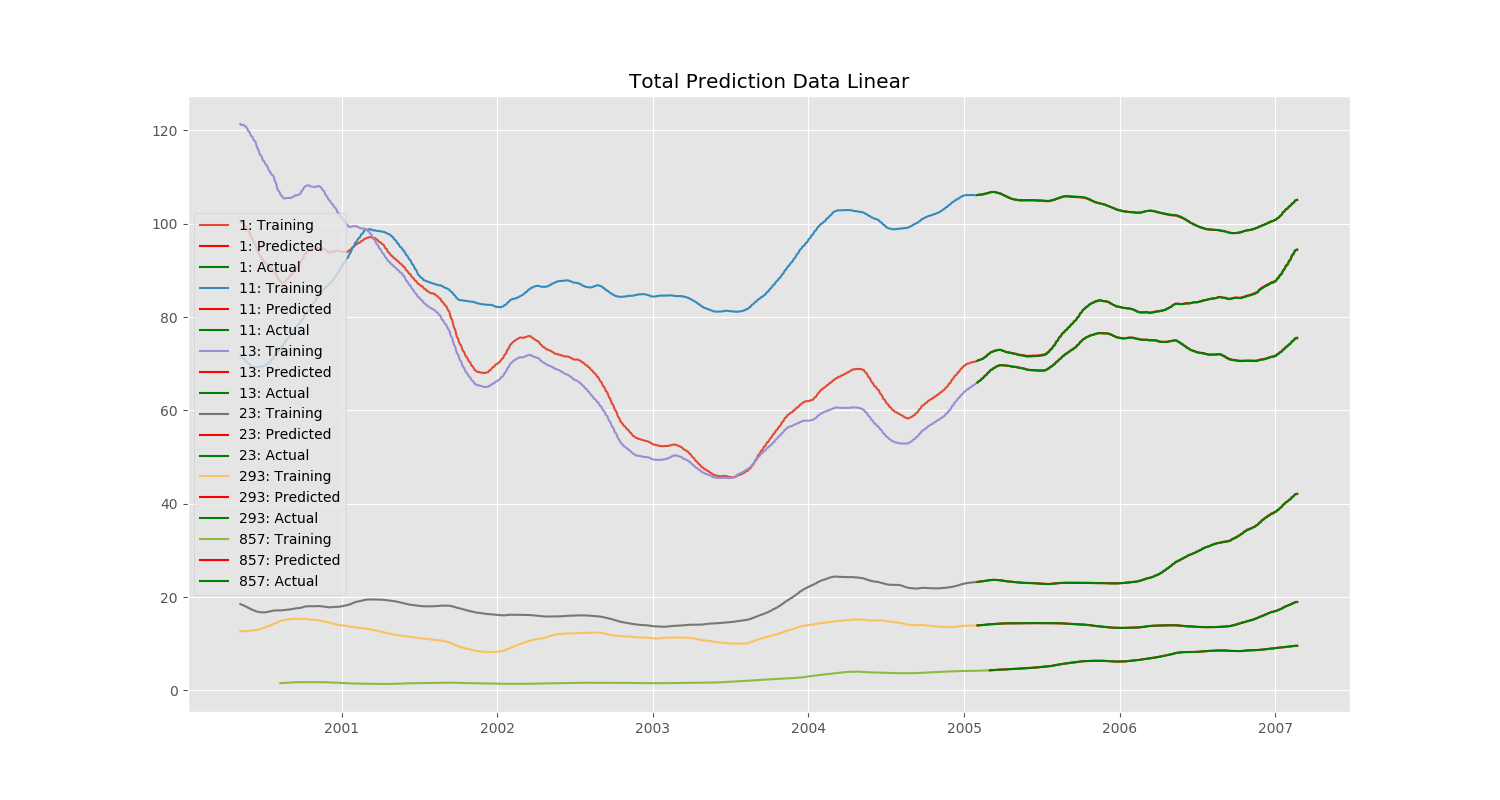
Here the model for ID 1 (orangish-red) nearly perfectly fits the training data, while the model for ID 857 (bright green) misses the predictions completely. As most of the other datasets had data that went up and down with more random trends, the predictions were generally more accurate. Taking a closer look at ID 857 reveals that the data only follows a major single upwards trend, where none of the data had been previously seen in that range and as a result the classifier does not know what to do with the data:



The most major factor for this is that since the model used was a classifier, it was unable to predict values it had never seen before. More evidence of this can be seen in the decision tree itself (these trees can be seen in the “img” folder of my project): 

The decision tree to the left is for stock ID 857, and the decision tree to the right is for stock ID 1. Thus, the complexity of each tree can also help to be a good determining factor for each stock.

 In my opinion, using a classifier to predict stock data is meaningless though, as if the stock goes above the expected range (i.e. stock ID 857), the data cannot be predicted since it is not part of the classification tree. Thus, I also applied a linear regression model which scored much higher accuracy wise and also visually. Here is the accuracy table and graph showing the plot:



As can be seen, the predicted data almost nearly completely overlaps the actual data. Thus, using a classification tree was not the most elegant or simple solution.

**Conclusion**

Overall, I learned various techniques from this project for data extraction from a dataset. A lot of what I had learned involved pre-processing data and the majority of my difficulties involved trying to visualize the data properly and deciding what to do for the project overall. I also learned the weaknesses of classification models as they cannot predict values they have never seen before. Thus, it would make more sense to use a different model in this project if we were to attempt to “accurately” predict the stock data.