**DSC 478 Final Project Report**

**Shuyang Ding**

**1856131**

**Predicting Stock Market Trend by Financial Numerical Indicators and Financial News**

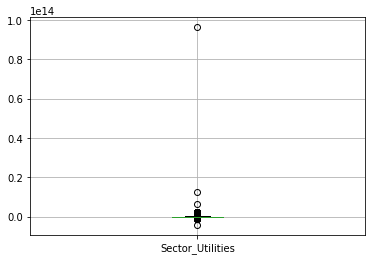
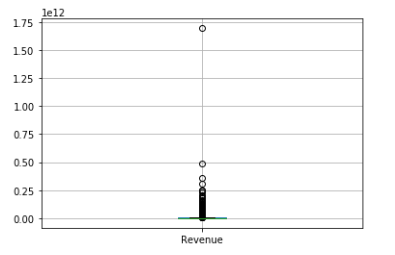
With more and more people getting into the stock market, trying to build a better model to predict stock price trend has become a crucial project for both financial institutes and individual traders. This project is aiming to use financial number indicators dataset and financial news dataset to predict stock market trends or to classify a stock to either buy or sell group.

Most of people spend majority of their time to analyze numeric indicators in company’s annual 10-K report, however, I’m wondering if financial news, which contains people’s emotion and thoughts could be another source to predict stock market trend as well. Therefore, I planned to apply various analysis methods on these 2 datasets and expect to get 2 models, one for numeric predictors, and one for news or text predictors.

The two datasets are both from Kaggle. The first dataset is traditional financial indicator, which contains 200+ financial indicators that are commonly found in the 10-K filings each publicly traded company releases annually. The other one is financial prices and news, which generates sentiment value from business news about companies and compares it with the stock market performance.

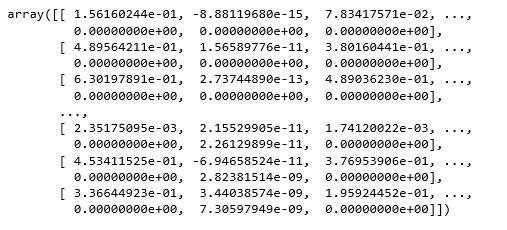
For the numeric dataset, I started with basic stats description to see if there are any missing values need to be handled. I decided to fill the missing values with average number as I assumed most companies have linear growth rate throughout the year. After filling the values, I moved to allocate dummy variables to the “sector” column, which indicates the industry that specific stock belongs to. With missing values filled and dummy variables created, I ran another stats description to make sure everything looks good, so the dataset is ready to be split and trained.

First, I split the target attribute from original dataset. The ‘Class’ column indicates whether investor should buy, a stock has positive price variance, or sell, a stock has negative variance. After separating target attribute and training attributes, I did some data visualization to see if there are any outliers. Following are some examples:

From the boxplots above, I figured that there’s couple outliers in the dataset, but as majority of the stocks are within normal range and the outlier could be a company outperformance like Amazon, so I decided to keep the outlier.

The next step I did was to normalize the dataset for clustering and classification as the indicators are in difference scale range from percent to multiple millionI went for normalization to make all data within range 0 and 1 .



Once the preprocessing is done, the dataset is ready to be used for clustering, which is an unsupervised way of classifying each data point into a specific group. In theory, data points that are in the same group should have similar properties or features. So, I planned to assign 2 clusters and see if all the stocks assigned to these clusters have similar price variance. If the result looks good, that means, we could use the clusters to predict whether we should buy the stock. For example, if a news/text with negative emotion, then investor might sell the stock and vice versa.

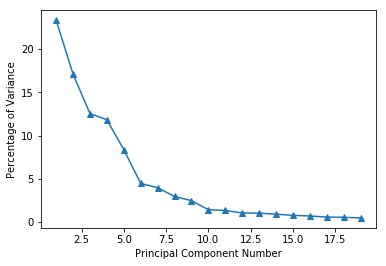
To evaluate how well each object has been classified, I calculated silhouette score, which is 0.1802. This quite low score indicates that the clustering configuration is not appropriate. In addition, I also calculated completeness score and homogeneity score, both are not in acceptable range as below.

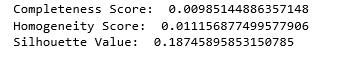


All these three scores above indicate that the clustering configuration may have too many clusters. So, I tried to include more clusters. Instead of separating cluster to positive or negative, I divided news into 4 groups from ‘negative’, ‘less negative’, ‘less positive’ and ‘positive’, to see if the score would be improved, but I didn’t get any better result.

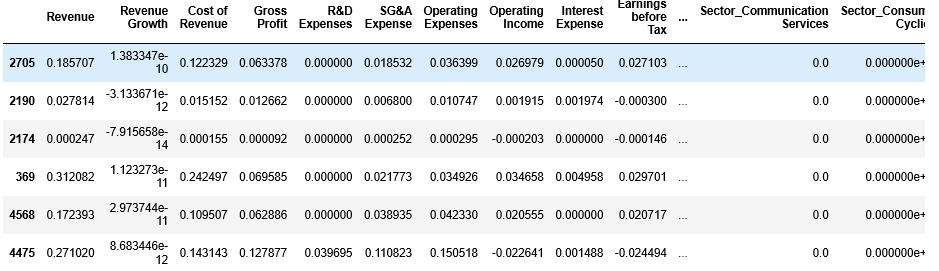
Then, I thought maybe using PCA method to reduce the dimension of the data could give me a better result. However, the final scores did not look good either, basically they were very close to the one without reducing the dimension.

Below is he PC variance plot and final scores:





As unsupervised method did not work well, I decided to go for supervised method, using classifier to predict class. First, I randomly split data into training and testing group in 80/20 proportion, and then normalized dataset so all the numbers could fit in range 0 and 1 like below.

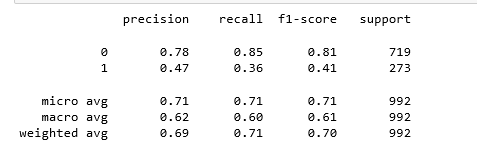


I started with KNN classifier. The accuracy Score was 0.7127, which is very good comparing to unsupervised clustering. Also, from the confusion matrix and classifier report below, the KNN classifier’s performance is decent to be used to predict a stock’s class, whether to buy or sell.

KNN Classifier – Confusion matrix:

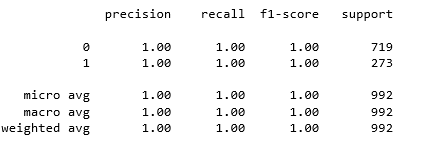


KNN Classifier – classifier report:



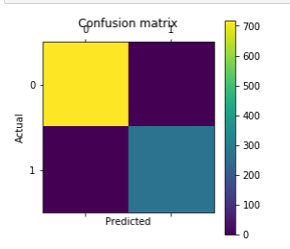
Then, I tried to build model with decision tree classifier. The result of decision tree classifier was surprisingly good with accuracy score as 1, and all other validation have same result as well.

Decision Tree Classifier – classifier report:

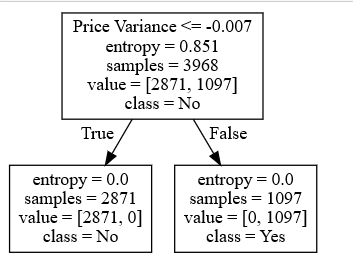


Decision Tree Classifier – Confusion matrix & Visualization:





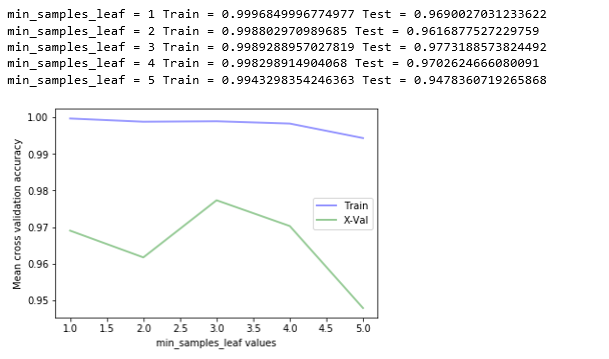
Decision Tree Visualization:



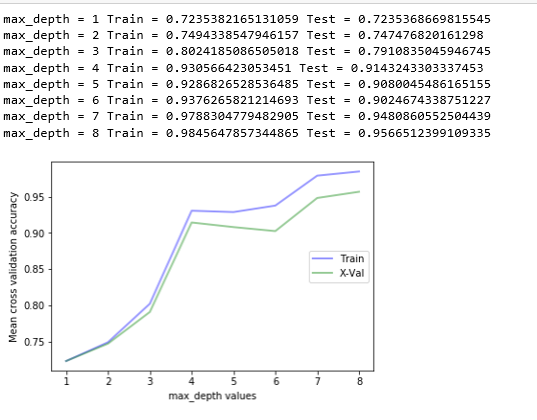
After getting a great score from decision tree, I tested Bayes classifier as well, just wanted to see how well the classifier would perform. However, I only got accuracy score as 0.31, which is even lower than KNN classifier. Therefore, I do not think Bayer classifier is good for this prediction project.

I then went for random forest, which is an ensemble method. Each classifier in the ensemble is a decision tree classifier and is generated using a random selection of attributes at each node to determine the split. The accuracy number for my first model was 0.957, which looks good. Then, I used “calc\_params” function to explore the impact of individual parameters using cross-validation. From the plot below, it seems that min\_sample\_leaf = 3, max\_depth = 8 and n\_estimator = 255 work well.

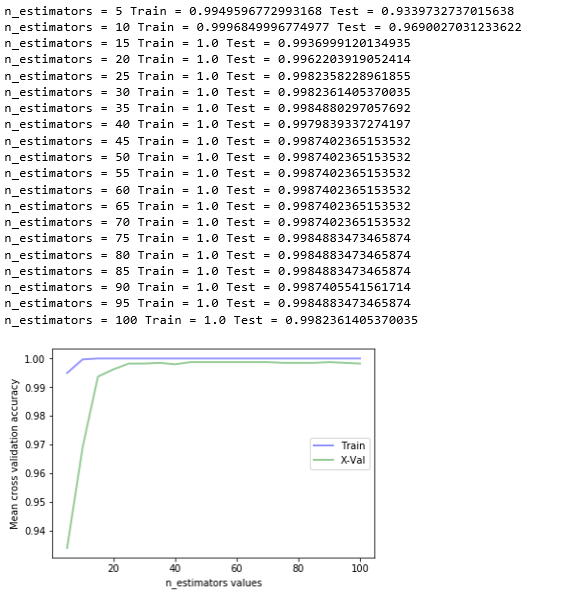
Min\_sample\_leaf:



Max\_depth:



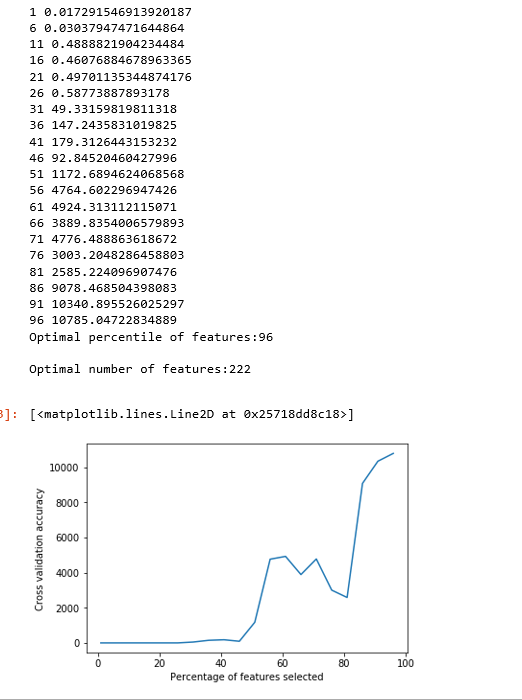
n\_estimator:



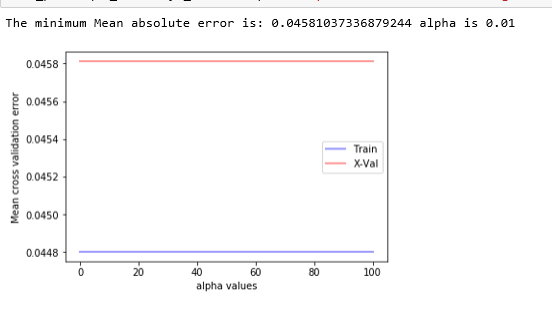
After applying these parameters back into the original model, my accuracy score was increased from 0.957 to 0.961, a good progress.

Not only could be used for classifying, this dataset could also be used for predicting future value of a stock. Therefore, I also tried to apply regression analysis to the dataset. Frist of all, I needed to separate new target “price variance” from original dataset. Then, I performed standardization and split standardized data into train and test set. So far, the preparation was completed, and the dataset was ready to be analyzed by different regression models.

I first started with simple linear regression with feature selection applied. Feature selection is the process of reducing the number of input variables when developing a predictive model. After running the function, I got 96 as optimal percentile of features and 222 as optimal number of features.



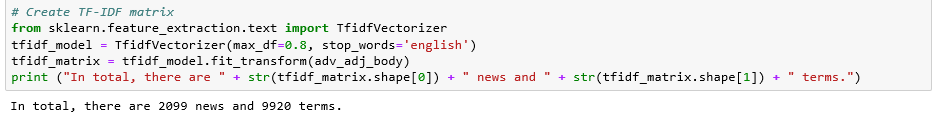
I used these two features to re-run the model, and then validated model by calculating mean absolute error, which is 0.3845, did not look bad. I then tried ridge regression and got RMSE on training: 0.0587 and RMSE on 5-fold cross validation: 1.1348. From the numbers, it looked like my model was probably done a good job at generalizing. In addition to these 2 regression models, I tried Lasso Regression as well. The result I got did not have much difference from the above two models.



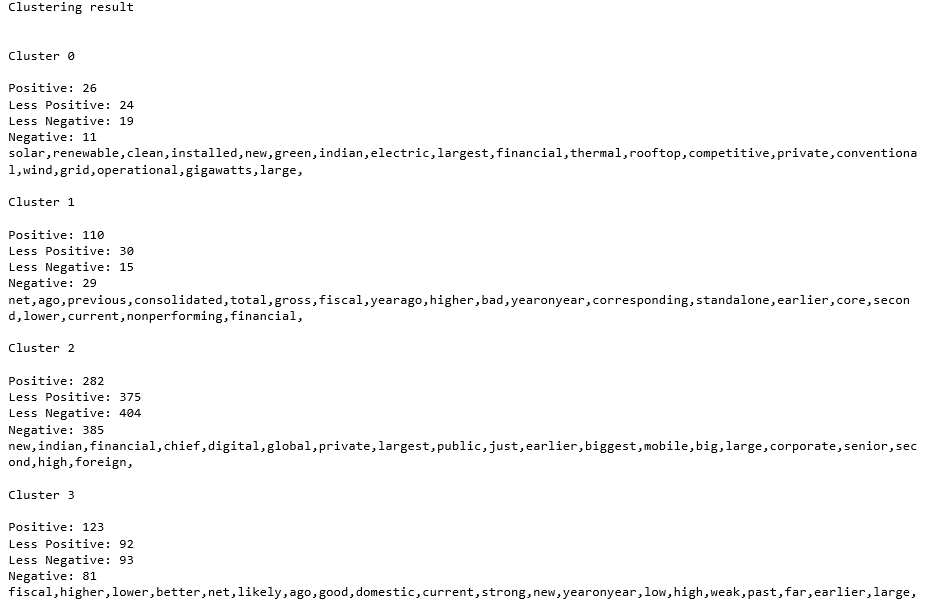
Finally, I attempted to use SGD regression method to train my data to see if I could get similar result. As SGD regression is very sensitive with data scale, I normalized original data to make all data are in rage 0 and 1. After couple test, I came up with a final model with l1\_ratio = 0.08508508508508508. However, when I use that model to calculate mean absolute error of set-aside test data, I got an extreme large number, which is over billion. I thought I did something wrong on my model, so I went through everything, but everything looked fine and worked well. I then did some research online and found if the target data and predictors have huge difference, then mean absolute error should be expected to be large. As I normalized dataset with predictors only, but did not preprocess on target data, I would think that is the reason caused huge mean absolute error. I then tried to normalized target data and got a mean absolute error very closed to the other three regression methods.

So far, I am basically done with analyzing financial indicator dataset. So, I then used similar methods on Financial news and price dataset. The difference between two datasets is that financial indicators are pure numbers, while financial news are texts and sentiment score the news belong to.

In pre-processing phase, I group those sentiment score into 4 groups base on percentile. Therefore, instead of hundreds of different scores, I eventually had 4 groups as “negative”, “less negative”, “less positive” and “positive”. Next, I cleaned text and title by removing all special characters and punctuations. As I wanted to group all news into those 4 groups, I thought adverbs and adjectives would be more representative in expressing emotions than other neutral nouns and verbs do. I extracted adverbs and adjective by using NLTK package. Finally, the most important step, I created td-idf matrix for the dataset, as of now, the dataset could be counted as a clean dataset and ready to be analyzed.

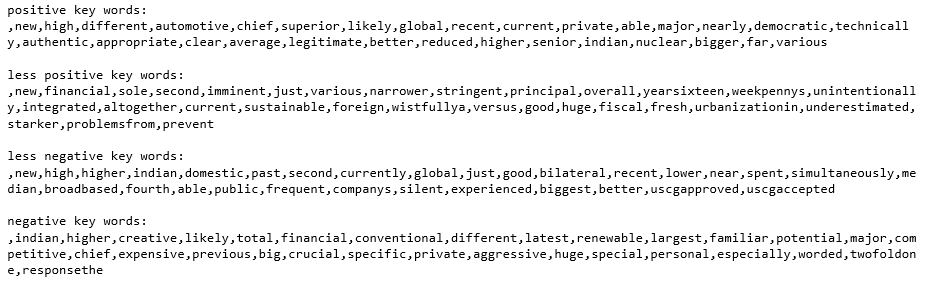


Like what I did before, I started analyzing with K-means clusters. My though here was to count frequency of positive, negative, and neutral words in each cluster, see if there is any obvious pattern, for example, among all clusters, which cluster has the most positive counts. Then that means, if an article has those words appeared a lot, then the article could be categorized into positive group.



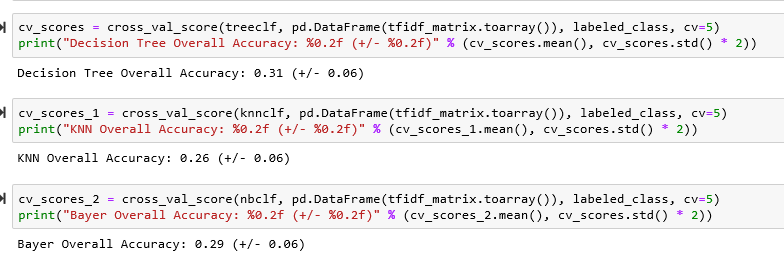
However, from the result above, the cluster did not uniformly distribute in each of cluster, and there's no obvious pattern.

So, I tried to count top 10 key words for each score group to see if all negatives words fall in negative group, and more neutral words land in less negative or less positive group. Below is the result:



As you can see, there is no obvious pattern, word “good” landed in both positive and negative group. Therefore, even though the news is negative it could still contain tons of positive words. So, I thought maybe using news body brought too much information, so I changed my direction to just use title to do clustering. Most of the time, new title summarizes the emotion of the entire news. However, the result still did not seem good. Positive and negative group seem have their related emotion words, but less negative and less active are hard to tell as most of the words were tended to be neutral.

So far, I though unsupervised method might not work, therefore, I went for supervised classification and expected it could bring me better result. Same as before, I used KNN classifier, decision tree and Bayes classifier. However, the result did not come in the way I expected. Here are the accuracy scores for all three models:



After this, I tried to use Ada boos classifier to improve the model. From the result, I had n\_estimators equal to 10 and learning\_rate as 0.6. However, the overall accuracy score still didn’t look good, which was only increased to 0.338. Similar result was given by random forest method. At this point, I do not think there is any way I could to improve the model. Therefore, I stopped there.

From the analysis above, it could be concluded that the classifier model for financial indicator dataset worked the best, the regression models were in the second place. I’m pretty confident in the classifier model and think it could be used to predict whether I should buy or sell stock going forward by using recent data and see if I could beat the market.

Compared to these two, the classifier models applied on financial news dataset did not work well. I think the main reason could be I did not create tf-idf matrix in a correct way. I might need to revisit my code see if I missed something. Also, it could be the sentiment score in the original dataset is not reliable as it was pre-processed by some other users and was not validated. Therefore, next step I could start with validating original dataset first. If everything looked fine, I would go for next step to check my work on tf-idf convention.