Stock price up/down prediction with related companies with classifying techniques

Jaeheuk Jung
jjung16@u.rochester.edu
University of Rochester
Rochester, New York

Abstract

This paper introduces a method to predict whether a company's stock price would go up or down using stock price data of other companies that are classified to be in relationship/ correlated with the company of interest. The data is a set of Comma Separated Values (csv) files of companies' stock data. This project was conducted for the final project of University of Rochester's CSC 240, 2019 Fall.

1. Introduction

Prediction of stock price has been an long unsolved project for scholars across multiple fields of studies, with no absolute method that would provide with a reliable prediction; Even though it is almost impossible to predict the stock price with high accuracy, however, this paper proposes that simple prediction such as predicting whether the stock price would go up or down in the next day or two could be possible with accuracy around 75%. This prediction has potential to be significant for trading put and call options, which is becoming increasingly popular since the early 2000s. The data mining, machine learning and prediction was based on the data from Kaggle named *Huge Stock Market Dataset – Historical daily prices and volumes of all U.S. stocks and ETFs*, which included the open, close, high, low, openInt, and the volume of trades with corresponding dates for 7193 companies. The dates vary by companies, generally spreading over the late 2000s to 2017, 2017 being the year the dataset was produced. The procedure of the prediction was divided into

- *pre-processing:*
  Parsing the csvs into data frames, removing insignificant companies' data, such as when the company being too new, or the volume of trade being too small.
- *Data mining:*
  Uses a genuine algorithm that was made for this study which returns the list of companies that are classified to be highly related with the desired company. From the list of companies, find out the dates that would be reasonable to be used for learning, and generating a data frame for learning.
- *Learning and prediction:*
  Classifying techniques from Google's deep learning library tensorflow, and other machine learning methods. Determine if this prediction is viable and make improvement unique for this prediction of this company, possibly repeating the mining process.

2. Pre-processing and Data Analysis

2.1 Cleaning and formatting

The dataset had comparably clean data, there was surprisingly NA values. To better process and get easier access for each index, I changed the csv values into data frames. Some of the csv files were not able to be read into pandas data frame. This was minor issue, however, since the number of csv files were 30 (out of 7223).

2.2 Analysis of Data of Interest

The companies were divided into two different groups on their trade volume: small and large. Since this project wants to find out whether relationships between companies could make predictions for the up and downs of stock price, the bigger companies were given more emphasis, and it was expected that the small companies would be more effected by price changes of larger companies. There seemed to be no clear correlation between the length of trading dates and the volume of trades, but instead there seemed to be some specific length of trading history that companies share in common. This data was later used in the learning section.

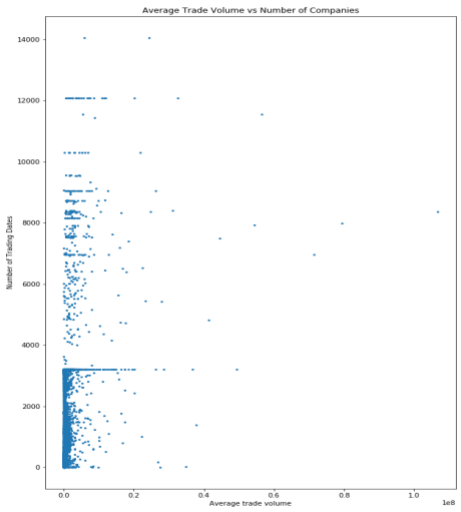


Figure 1 Scatter plot of Average Trade Volume vs length of Trading History

Another important information for this project,

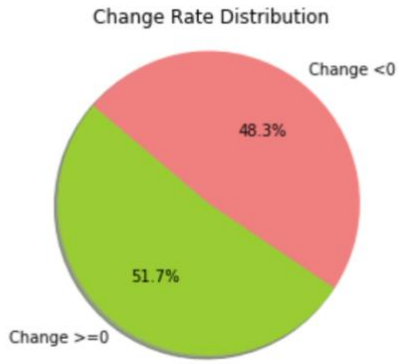
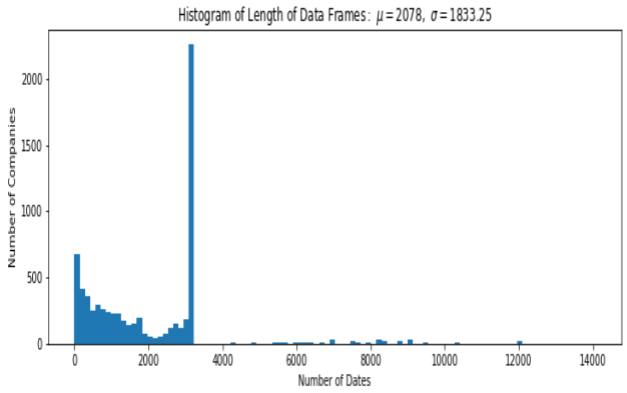


Figure 2 Pie chart for Change Rate of data overall

the change rate of the stocks, demonstrated the distribution of positive (includes 0) and negative was .517 to .483, which meant that if the classification prediction at least has to have better accuracy than 52%. It also had the variance of .044 which meant that the change rates are concentrated densely to each other.

2.3 Clearing insignificant companies

The important characteristic of the companies to perform learning with was that it has a *significant* amount of trade volume and long enough trading dates, not only for the sake of learning process, but since the main algorithm of mining process, *relativity*, demonstrated abnormal behavior toward companies with short length for dates.



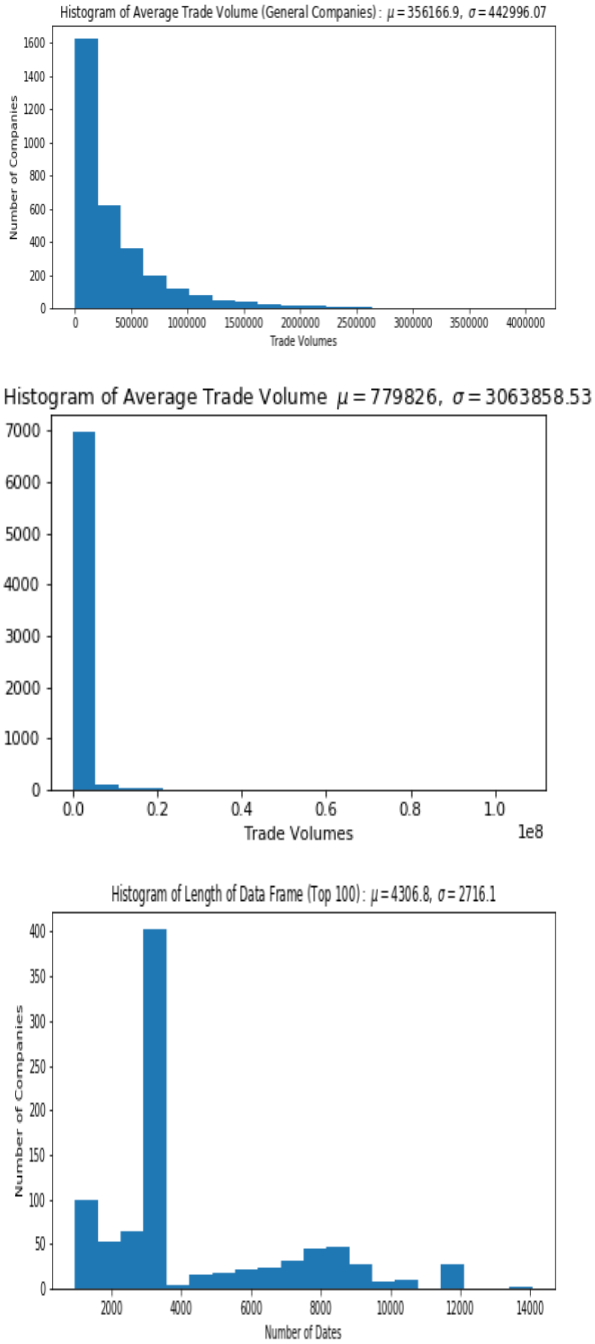


Figure 3 Histograms of Data of Interest

Therefore, a threshold for choosing which companies are significant had to be determined. The average volume of trade was observed to be extremely skewed to the left, and it was observed that there was a significant difference between the averages of general companies being 356167 and the hundred biggest companies being 4172660. Both graph demonstrating exponential decrease of company numbers as trade volumes increase. This made it difficult to determine where to drop companies that would be considered as "too small". For the sake of saving time, the threshold was determined manually each time for each trial in the data mining stage than to coming up with an empirical formula for

finding the threshold. The number of dates of the companies were Median of big companies was 3201, small companies 3175. This was very interesting, since one bin of each histogram had very distinct number of companies. This made it convenient to determine the threshold for average volume, which was 1 standard deviation around 3200. The companies filtered with this threshold were around five thousand. A further study to find out why so many companies have the similar length of history and if there are specific relationships between those companies would be interesting.

3. Data Mining

3.1 Central method

To find out the relativities of companies, a function called *relativity* was developed. The central idea for this project revolves around this method. *Relativity* takes in two companies to compare and returns whether the companies x and y are related or not. It first adds the companies' change rate for each date in the data frame. Then with each date for both of the companies, if the dates match, it compares the change rates of the data. If they are both positive or both negative, Relativity adds weight to the possibility that the companies are positively correlated. When the change rate of both companies is different for a date, the method adds weight to the possibility that the companies are negatively correlated. If the weight positive or negative correlation adds up to pass a certain threshold, x and y (regardless the type of correlation) are determined to be related. The threshold was chosen manually, lowering the threshold if there are only a few related companies, and raising it when there were more than 40 related companies. It would be interesting to find the best threshold through machine learning afterwards. This method is yet imperfect, and it would be addressed in later section of this paper; but it is functional enough for making predictions.
The pseudo code for the function is:

```
Relativity(x,y):
X= data frame of company x,
Y= data frame of company y.
change_X=[]
change_Y=[]

for (open, close) in X:
    change_X.add(date, (close-open) as change)
    for (open, close) in Y:
    change_Y.add(date, (close-open) as change)

    for i in change_X:
        for j in change_Y:
if dates of x and y match:
            if change of x and y are both >0 or <0
                positive = positive + i/j (j/i)
            if change of x and y are different:
                negative = negative +i/j (j/i)

if positive/negative >1.3 or <.7
            the companies x,y are related.
        else:
            the companies are not related.
```

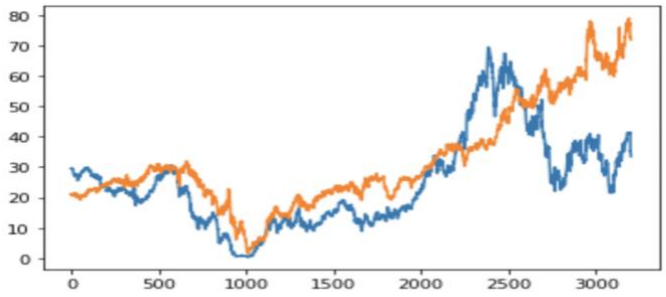


Figure 4 Two companies with ~60% from relativity function.

This method has O(d^2*N^2) complexity, d referring to number of companies and N for the length of data of each company.

3.2 Sampling

The optimal approach for the purpose of this paper would be to examine this method for every companies but was extremely costly to do so. Instead, 10 random companies from the biggest hundred companies were chosen, for each 500 random companies from the general company group were compared. Then 10 random companies were chosen from general companies to be compared with the top five hundred companies as well using the *relativity* function.

3.3 Processing

The companies that passed the threshold were appended to a list named *attributes.* For the accuracy of the prediction from the learning process, the dates were chosen that were common for most of the companies in the *attributes* list. If a company did not have enough common dates, the company was dropped from *attributes*. Then with change rates for the common dates for each of the companies, a data frame was made. The data were already mostly in between -1 and 1, and therefore normalizing was not necessary. Lastly, the classification, whether the change rate of the company for the prediction is >=0 or <0, is added to the data frame.

	931	935	430	432	438	469
2017-09-20	-0.000934	0.000000	0.008086	-0.001643	0.026444	-0.003279
2017-09-21	-0.010163	0.046791	0.000000	-0.009917	0.000740	-0.020067
2017-09-22	0.004090	0.000000	0.002670	0.040833	0.008458	0.027491
2017-09-25	0.001988	-0.015852	0.001328	-0.003205	0.004411	-0.010000
2017-09-26	0.003133	-0.050000	0.003963	0.012903	0.019226	0.013699

Figure 5 The data frame made for learning. The attributes are the company names and the values are the change rate for each date.

4. Classification and Prediction

4.1 Learning methods

The method chosen for prediction were deep learning with artificial neural networks and random forest classification, since random forests are known to show descent performance with smaller number of samples and deep learning perform better when the sample size is large.

4.2 Random forest

For random forest, 80% of the data was set to be training set, the rest 20% being the test set. Testing with 30% or 10% of the data generally decreased the accuracy to around 3%. The model implemented 10 or 15-fold cross validation, chosen differently for each company based on their performance. The standard 5 fold cross validation demonstrated around 5% less accuracy in general. Scikit-learn library's ensemble- *RandomForestClassifier* was used. Confusion matrix and Classification report Scikit-Learn's Metrics library were implemented to evaluate the prediction accuracy.

4.3 Artificial Neural Network

Tensorflow's Keras library was implemented for the neural networks, with four Dense layers including the output layer. The activation method used was *relu (rectified linear unit)*, kernel initializer as *uniform*. The loss function used was *binary_crossentropy*, which works best for two value classifications. This could be only used in special cases such as this project, when there are only two values for classification. If other loss functions were used, the accuracy most of the times did not improve throughout the learning process regardless the changes on batch size, layer number, or the train/learn data proportion. Lastly, the optimizer applied was *adam (adaptive moment estimation)* which provides the best results for most of the deep learning cases as of now.

$$\mathcal{L}(\theta) = -\frac{1}{n}\sum_{i=1}^{n}\left[y_i \log(p_i) + (1-y_i)\log(1-p_i)\right]$$
$$= -\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{m} y_{ij}\log(p_{ij})$$

Figure 6 binary cross entropy loss

4.4 Results

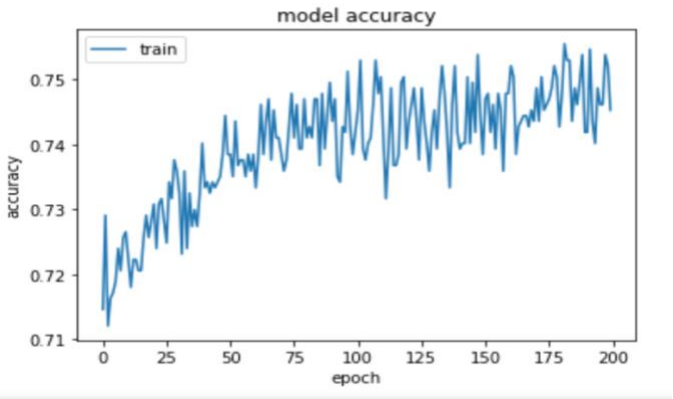
Case 1: 17 companies of reference (related companies), 1468 dates.

```
=== Classification Report ===
              precision    recall  f1-score   support

           0       0.64      0.64      0.64       147
           1       0.64      0.64      0.64       147

    accuracy                           0.64       294
   macro avg       0.64      0.64      0.64       294
weighted avg       0.64      0.64      0.64       294


=== All AUC Scores ===
[0.5959596  0.53535354 0.58585859 0.55555556 0.68686869 0.65656566
 0.63265306 0.68041237 0.69072165 0.57731959 0.67010309 0.65979381
 0.59793814 0.57731959 0.57731959]


=== Mean AUC Score ===
Mean AUC Score - Random Forest:  0.6186495008291768
```

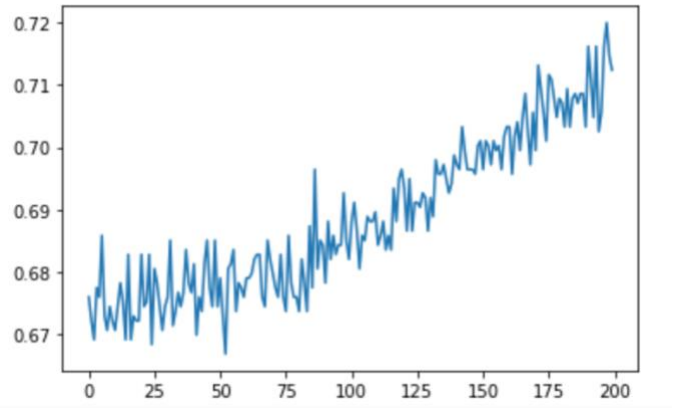
Random Forest for case 1. Test size : 20%

```
=== Classification Report ===
              precision    recall  f1-score   support

           0       0.57      0.68      0.62        69
           1       0.66      0.55      0.60        78

    accuracy                           0.61       147
   macro avg       0.62      0.62      0.61       147
weighted avg       0.62      0.61      0.61       147


=== All AUC Scores ===
[0.58783784 0.55405405 0.59863946 0.61904762 0.62585034 0.6122449
 0.60958904 0.63013699 0.56849315 0.50684932]


=== Mean AUC Score ===
Mean AUC Score - Random Forest:  0.5912742697967747
```

Random Forest for test 1. Test size: 10%

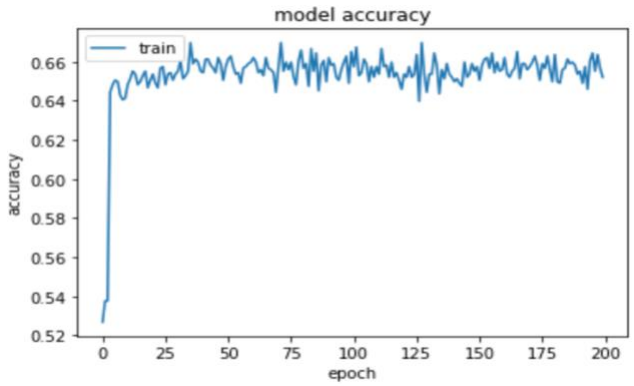


Deep learning with ANN for case 1. Test size: 10%, Final accuracy: 74.5%

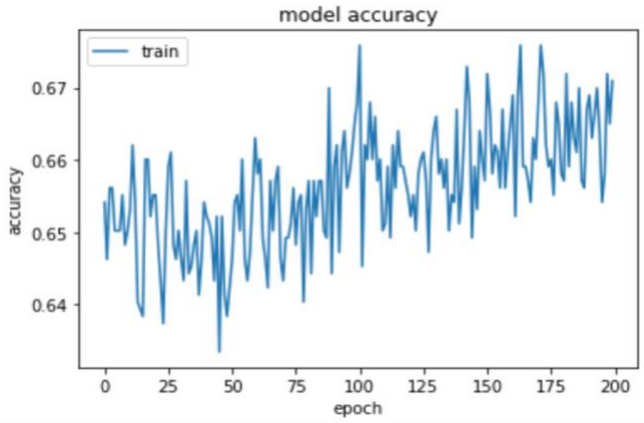


Deep learning with ANN for case 1. Test size: 20%, Final accuracy: 70.8%

Case 2: 16 companies of reference (related companies), 1447 dates.



Deep learning with ANN for case 2. Test size: 10%, Final accuracy: 65.2%

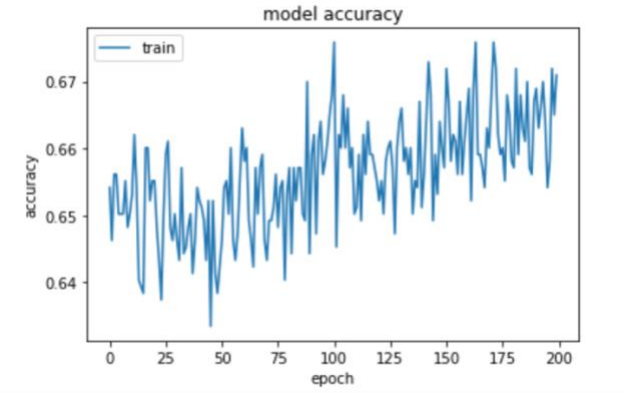


Deep learning with ANN for case 2. Test size: 30%, Final accuracy: 67.1%

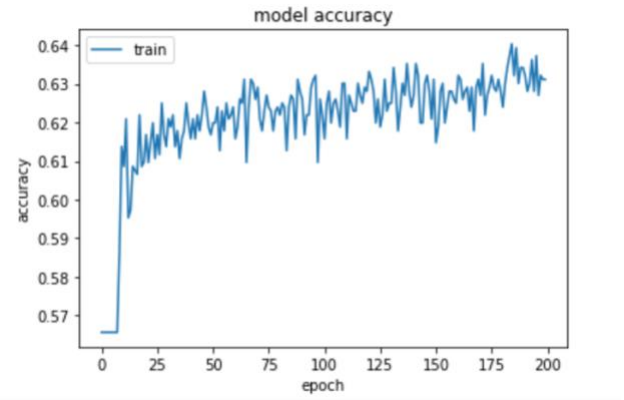
```
=== Classification Report ===
              precision    recall  f1-score   support

           0       0.56      0.64      0.60       205
           1       0.63      0.56      0.59       230

    accuracy                           0.60       435
   macro avg       0.60      0.60      0.60       435
weighted avg       0.60      0.60      0.60       435


=== All AUC Scores ===
[0.67586207 0.51034483 0.60689655 0.65517241 0.57931034 0.64137931
 0.57931034 0.59027778 0.56944444 0.52777778]


=== Mean AUC Score ===
Mean AUC Score - Random Forest:  0.5935775862068966
```

Random Forest for test 2. Test size: 20%

Case 3: 18 companies of reference (related companies), 1085 dates.



Deep learning with ANN for case 2. Test size: 10%, Final accuracy: 67.0%



Deep learning with ANN for case 2. Test size: 20%, Final accuracy: 62.8%

```
=== Classification Report ===
              precision    recall  f1-score   support

           0       0.47      0.75      0.58        53
           1       0.43      0.18      0.25        56

    accuracy                           0.46       109
   macro avg       0.45      0.47      0.41       109
weighted avg       0.45      0.46      0.41       109


=== All AUC Scores ===
[0.55045872 0.55963303 0.53211009 0.55045872 0.6146789  0.56481481
 0.57407407 0.61111111 0.62962963 0.51851852]


=== Mean AUC Score ===
Mean AUC Score - Random Forest:  0.5705487597689433
```

Random Forest for test 3. Test size: 10%

```
=== Classification Report ===
              precision    recall  f1-score   support

           0       0.58      0.81      0.68       117
           1       0.59      0.32      0.42       100

    accuracy                           0.59       217
   macro avg       0.59      0.57      0.55       217
weighted avg       0.59      0.59      0.56       217


=== All AUC Scores ===
[0.53424658 0.45205479 0.50684932 0.60273973 0.61643836 0.61111111
 0.65277778 0.58333333 0.66666667 0.58333333 0.55555556 0.66666667
 0.625      0.51388889 0.54166667]


=== Mean AUC Score ===
Mean AUC Score - Random Forest:  0.5808219178082191
```

Random Forest for test 3. Test size: 20%

4.5 Result Analysis

The deep learning method overall produced better predictions, with the average accuracy of 67.9% compared to 59% of Random Forest classification method regardless the size of the samples or the number of related companies. This may be due to the fact that all the cases had arguably large samples, with all the trials having greater than 1000 samples. Random Forest method demonstrated questionable performance for this approach, showing "capped" accuracy under 65% of any cases.

For neural network method, some interesting characteristics were found, such as the increase unit size, or the dimensionality of the output space, for the layers shown to improve the prediction in some cases. It showed better performance when the training set was larger, as shown in case 1 and case 2 compared to case 3. Also shown in setting the training set proportion to 90% getting the best results. The number of related companies seemed not to affect the accuracy of the prediction but would need a more testing for an adequate conclusion.

5. Conclusion

The approach to classify the up and downs of a stock price relativity was shown to have better than guessing (52%) accuracy, with near 68% accuracy with deep learning techniques. The results showed that some manual adjustments to the neural networks could improve the predictions, so development an empirical standard for the networks would surely improve the outcome of this project. Some improvements on the relativity function is needed as well, since 68% accuracy would not always be considered a good enough accuracy statistically. This project would be a good source for put and call investors, since their primary interest is to predict whether the stock price would go up or down in the near future. Further study of predicting price up and downs of dates of further future would be the ultimate goal of this approach. As well, applications such as CUDA would increase the completeness of this project, since the *relative* function is time-costly.

Figure 6 citation
https://stats.stackexchange.com/questions/260505/machine-learning-should-i-use-a-categorical-cross-entropy-or-binary-cross-entro