Stock Trading with Machine Learning

Wenye Ouyang, M.S.

Nima Zahadat, Ph.D.

The George Washington University

**ABSTRACT**

Stock trading has been a very intriguing topic since the foundation of Amsterdam Stock Exchange (AEX) in the year 1602, when huge numbers of people started trading with different kinds of strategies. The purpose of this project was to explore how well different machine learning algorithms predicted the future trends of stock price. The models used were support vector machines, random forest, k-nearest neighbors and voting classifier. The predicting accuracy of the voting classifier on testing sets were significantly better the other three training algorithms, which is 40% accurate averaging the results from 18 stocks. Then the four models were validated on the recent data from 18 selected experimental stocks. Finally, recommendations for further research were provided.

*Keywords:* KNN, Random Forest, SVM, Voting Classifier, stock trading.

**INTRODUCTION**

For centuries, countless individuals, companies and organizations have been trying to come up with their own strategies of trading on the stock market, however, only few of those achieved considerable profits. Running with huge potential risks, most amateur traders were trading simply based on their instincts that, most of the time, inaccurately. The reason is, in order to make profits on stock market, the trader needed to successfully predict the future trend of certain stocks and made the correct moves and making predictions about future should be based on the information about the past, which, for instance, could be the movement of price in the past 20 days or the total trading volume on the prior day. These categories of information were voluminous, complex, and dirty data sets which were almost impossible for human brains to efficiently process, mine and analyze.

Fortunately, as the improvement of computing power, computers were constantly changing people’s lives by fast processing the voluminous data incompatible with humans themselves. Machine learning is a research field in artificial intelligence and statistics that aims to develop computational methods that can be used to learn from data and to predict with new data. Many machine learning methods, such as decision trees, k-nearest neighbors and support vector machine, have been developed (Ningchuan Xiao, 2017). The combination of human brains and machines learning models made the mystery behind stock market less ambiguous than before.

**LITERATURE REVIEW**

Stock prediction was attractive, extremely difficult but has been proved by many scholars to be achievable. It was extremely hard because of the numerous numbers of factors need to be considered during the work. In the study of Xiaodong Li, Haoran Xie, and Raymond Y. K Lau, based on the data of Hong Kong Stock Exchange stocks from year 2005 to year 2008, the sentiment transfer learnings can significantly improve the prediction performance of target stocks (2018). And an Indonesia research team lead by Santoso Murtiyanto, Sutjiadi Raymond and Lim Resmana published their work about how the results created by Gaussian Mixture Model- Support Vector Machines could effectively perform better than other predictive models by a sharp ration of 3.22 (2018).

All water under the bridge is not a very correct saying for the case of stock market. Ever since the early 90s, Scholars has been publishing papers about how exploring about the details of historical stock data would help discovering hints about what might happen in the future. John Y. Campbell, Sanford J. Grossman and Jiang Wang explain the phenomenon that stock return autocorrelations tend to decline with trading volume using a model in which risk-averse "market makers" accommodate buying or selling pressure from "liquidity" or "non-informational" traders (1993). They implied that a stock price decline on a high-volume day is more likely than a stock price decline on a low volume day to be associated with an increase in the expected stock return.

Feature selection was also one of the most crucial parts that could influence the performance of a model. To quote from the study of Lahmiri Salim, several studies have adopted feature selection techniques to improve the classifier accuracy in predicting financial risk. Indeed, the goal of feature selection is to identify the most informative features used as patterns in the classification task and remove redundant ones. In this regard, selected features are expected to improve classification/prediction results and help in reducing the processing time of the classifier (2016).

**RESEARCH METHODOLOGY**

Python 3.6 was the programming language that the whole analysis was based on. Pandas, NumPy, and Beautifulsoup4 were the packages used to pull and compile the raw data into csv file. Data preprocessing techniques were also used to reshape the data structures in order to make the them applicable for the model, which included filling null values, calculating correlations, creating labels, selecting features, and train test split etc. Exploratory Data Analysis was conducted both before and after the data preprocessing part, where the matplotlib package was used to visualize the historical price, volumes and correlation tables. Finally, k-nearest neighbor, random forest, support vector machines and deep autoencoder were used to construct the machine learning models which predicted the trend of one stock in the future 7 days. Grid search method was applied while tweaking the models’ hyperparameters and input features in order to achieve the best performances. Then the performances of each model were evaluated in two different ways: accuracy and profit rates. Finally, conclusion and future recommendations were provided based on the results of all the processes above.

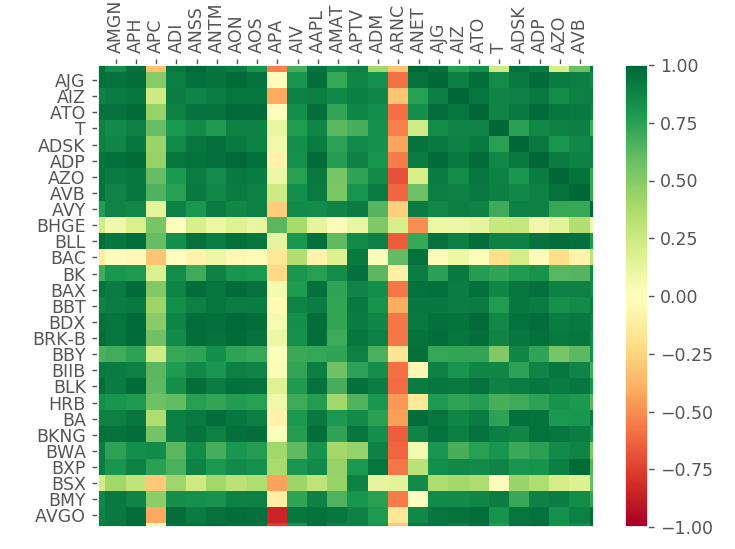
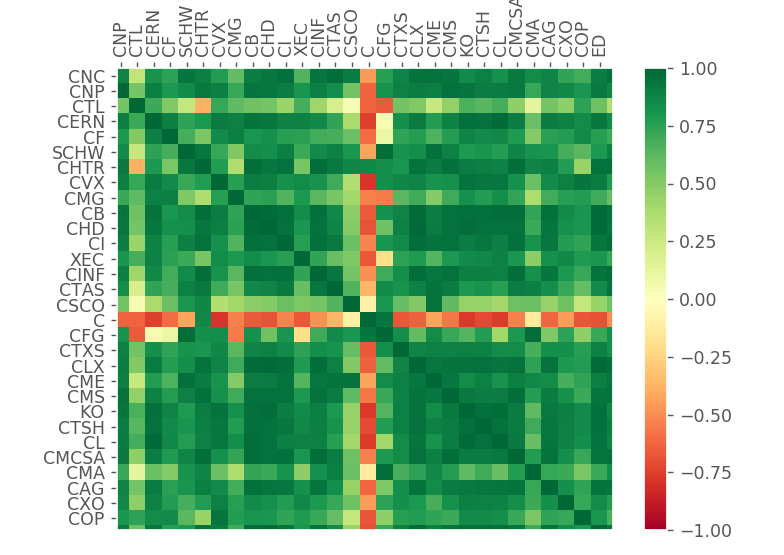
**DATASETS**

Two different datasets were used in my research. The first is a dataset I web scraped from [*https://finance.yahoo.com/*](https://finance.yahoo.com/)utilizing beautifulsoup4. It contains the historical data of all the S&P 500 from 2000-01-03 to 2019-04-10, with a total number of over 3 million data instances. Each data point has 5 features: opening price, highest price, lowest price, closing price and adjusted price, with date being the index. This dataset was used in the k-nearest neighbors, random forest, support vector machines and deep learning part.

The second one was a more detailed dataset provided by Wharton business school. Each data point in the second dataset contains 40 different features, however, some of which are obviously not very necessary for predicting the future trends of stocks. After the dataset refinement and preprocessing, 25 features were left for the following training process. This dataset was used only in the deep learning training part.

**Correlation Table**

In order to explore the relationships between the price movement of all the companies, I visualized the correlations between them using a heatmap table with the adjusted close price of each stock. The following were two sections of it after being zoomed in.

*Figure 1*. The correlation plot between stocks pair wisely.

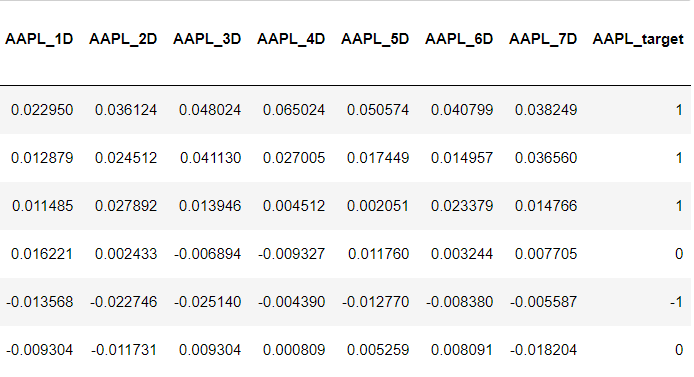
It could be discovered from the graphs that the correlations between stocks did exist, and the statistics proved that more than 65 percent of pairs of stocks had a correlation above 0.5 or below -0.5, which means they are highly correlated. As a result, it would be reasonably to predict the price of some certain stocks utilizing the price of some other stocks, including these stocks themselves, which are highly correlated with them.

**DATA PRE-PROCESSING**

The data preprocessing was one of the most crucial and tricky parts of this project. And since there were two datasets being used, I would introduce the two preprocessing parts separately.

Since some company would distribute cash dividends or issue stock splitting over periods of time, the price for recent days were not comparable directly with the price 10 years ago. That is why I choose to include only the adjusted close price for my first dataset analysis.

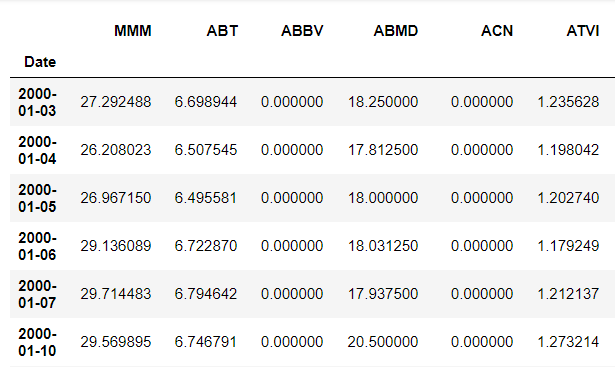
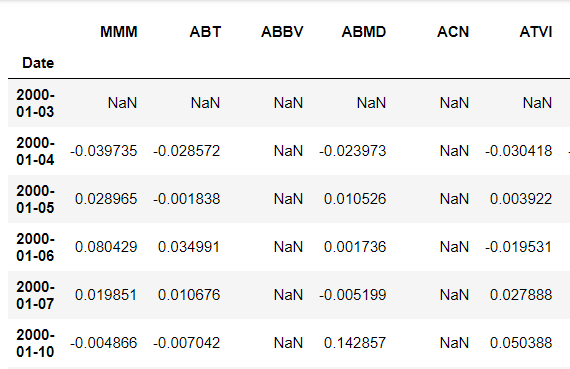
The next step was creating the training labels. The trading cycle I choose was future 5 days, 7 days and 10 days (The training results showed that labeling using 7 days trading cycle gave me the best accuracy). For the price of every date in my dataset, what I did was that calculating its price percentage change in the future 5, 7 and 10 days, and it looks like below (I used 7 days trading cycle here as a demonstration).



*Figure 2*. The demonstrating samples of labeling based on Apple Inc.

The label for every day was created based on the its future 7 days percentage change. Here I set up the requirement to be 0.02 and -0.02, which meant that in the future 7 days, if the price change of any date was above two percent, the target label would be 1, meant a move of buy. If the price change of any date was below negative two percent, the target label would be -1, meant a move of sell. If during the 7 days, none of them has a percentage change of above two percent or below negative two percent, the target label would be 0, meant holding it.

For different S&P 500 companies, comparing their prices directly was not very informative. For instance, a five dollar change in the price of Apple Inc did not mean the same thing as a five dollar change in the price of Citi Bank Group. To solve this problem, I normalized the price change into percentage change. Since the data starts from 2000-01-03, the price of that day would be the base value, and the price of future days was converted into percentage change compared to the price of its prior date. The following graphs showed the transformation clearly, with the left one being before the conversion and right one being the afterwards.

*Figure 3. Demonstrations of before and after transformations*

After the normalization, there were some null values appeared in the dataset for two reasons. The first one was that the base value would be null since it had nothing else to compare with. The second reason was that some stocks has not started trading since year 2000. My strategy of dealing with these null values were filling them all with 0, which means that the percentage change for that day was 0.

It was a common sense for machine learning engineers that if too many features were included during the training, the performance of the model was always poor. The features of my data were the prices of all the other stocks on the same day. My feature selection was based on the correlation values visualized within the previous section. The final features I selected, after several training experiments, was the top 30 stocks that had the strongest correlations with my target stock. Therefore, for each target stock, there would also be a correlation score which was calculated by adding up the absolute values of correlation values from the top 30 correlated stocks. The correlation score would also contribute to part of my analysis during the following training step.

The second dataset had 47 features originally. After the deletion of unnecessary ones, 25 features were left. The percentage change calculation and labeling were very similar to that of the first dataset, however, the processing for features like cash dividends and stock split factors were trickier than I thought. For instance, apple issued cash dividend of 2 dollars and 3 dollars per share on March 1st and June 12th, for the time stamps between these two dates, the entity for cash dividends was 0. However, the cash dividends should still be considered as an important factor within this time period. My strategy of dealing with it was filling all the 0 entities under cash dividends with the value of its prior dividend value.

**DATA ANALYSIS**

The datasets were ready for training after the previous preparation step. 75 percent of data was selected as the training set and 25 percent of data was selected as the testing set. One important thing to be noted here was that the selecting was not pure random. The reason was that if the dataset was shuffled randomly, then there would be a very high chance that the training set was highly unbalanced, which would cause a disaster for my model. If the training dataset was highly unbalanced, we would finally get a seemingly very high accuracy rate on the training set but turns out our model to be overfitting. For example, if 80 percent of our data were with the label ‘Buy’, then our model would stop learning and simply predicting every new data to be ‘Buy’ without any considerations. As a result, the selection of training set should not be pure random, but half random half manually interfered. What I did was first randomly split 75 percent to be training set then check the data spread within. If it was highly unbalanced, swap some of it with the testing set until the percentage of each category lies within +-5% of 33%.

K-nearest neighbors, random forest and support vector machines were trained on the first dataset. The hyper parameters were set up by comparing the accuracy rates utilizing the grid search method, and the final hyper parameters for k-nearest neighbors: neighbors = 20, random forest: max-leaf = 20, support vector machines: L1 regularization, one vs rest and C = 10. The hyperparameters for every model were chosen using grid search based on the averaged testing accuracy of ten times.

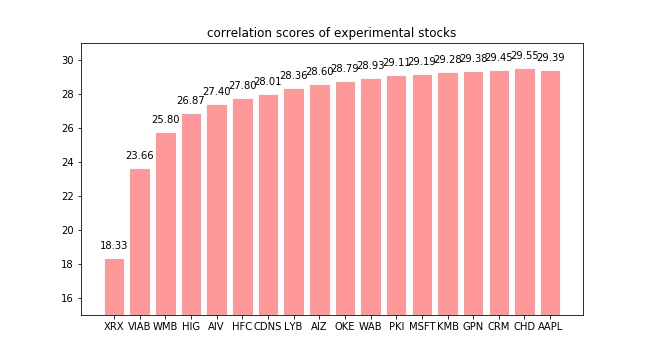
**KEY FINGDINGS**

The training results were interpreted in two different ways. The first was the accuracy based on my 25 percent testing set. After averaging the results 18 times trained on different splitting of training and testing sets, I got the final accuracy of every model. For k-nearest neighbors, the averaged accuracy was 0.3523; For support vector machines, the averaged accuracy was 0.3605; For random forest model, the averaged accuracy was 0.3547; And finally, for the voting classifier, the averaged accuracy was 0.4092, which was the strongest among all the models.

If we were investing based on random guesses, the probability for correctly predicting the future 7 days price value would be 0.3333, which meant that based on the results of all our models, the performances were only slightly better than the random guessing by around 2 to 5 percent. My machines were learning something, but not very much.

On the other hand, things would be very different in practice. The accuracy was just the numbers of correct predictions. Those wrong predictions, which were 20 percent more than the correct ones in numbers, might hurt the profits more than we thought. As a result, if traders were trading completely based on the predictions of my model, the risk would be huge because what the model predicted wrong could possibly make the traders lost big amount of money.

To validate what would be happening in practice, I gathered the most recent stock price data for all the S&P500 companies from 2019-04-10 to 2019-04-30. Also, I picked 18 experimental stocks from different correlation scores regions and visualized the profits if I invested on them based on my models. Their correlation scores were shown in the following graph:

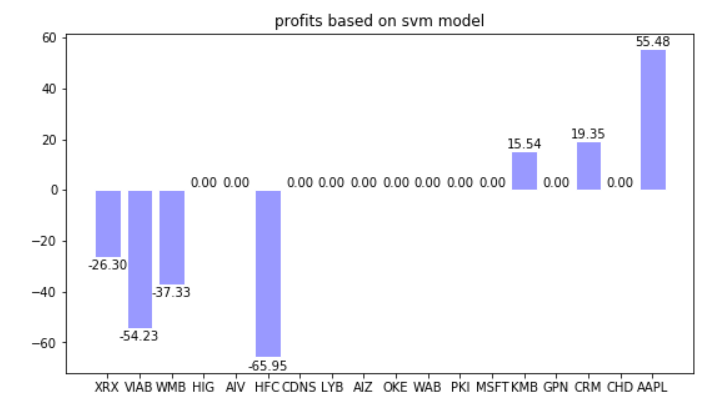


*Figure 4*. The correlation scores of the selected experimental stocks

In order to compare whether there were some relations between the performance of my models and the correlation scores, the 18 experimental stocks were selected from different correlation scores regions. My way of selecting was first ranking all their correlation scores from low to high and then randomly pick one or two from every 30 stocks.

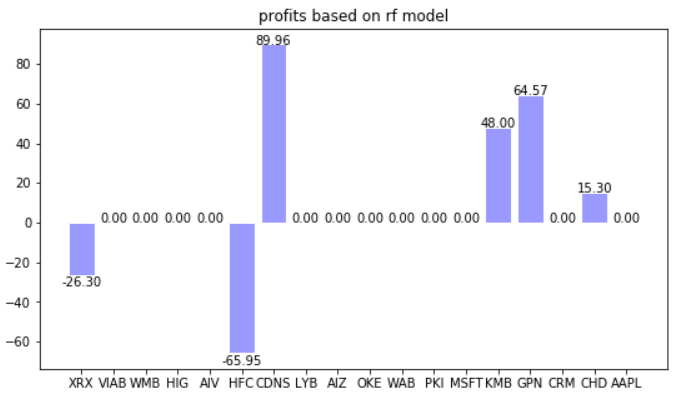
The theatrical range of correlation scores would be from 0 to 30. However, it is impossible for any individual stock to be 100 percent correlated or 0 percent correlated with the other ones. As a result, the actual range was from 18.33 to 29.55. My hypothesis here was that for those stocks which were highly correlated with other ones, it might be easier for my models to capture their movements and provide good performances. And for those which were less correlated with other ones, it might be more difficult to capture their movement.

The following graphs showed the profits I got if I invest 1000 dollars on these 18 experimental stocks started on April 10th and ended April 30th based on my models.



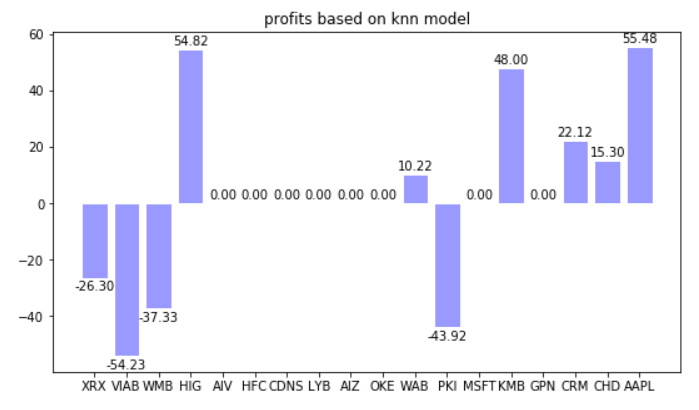
*Figure 5*. The profits of 1000 dollars based on the SVM model

Combined with the graphs of correlation scores, it was very clear that the performance of SVM model was related to the correlation scores of our target score. The model successfully captured the up trending of AAPL, CRM and KMB but failed to capture the down trending of XRX, VIAB, WMB and HFC, which were less correlated with the stocks model used to predict their trends.



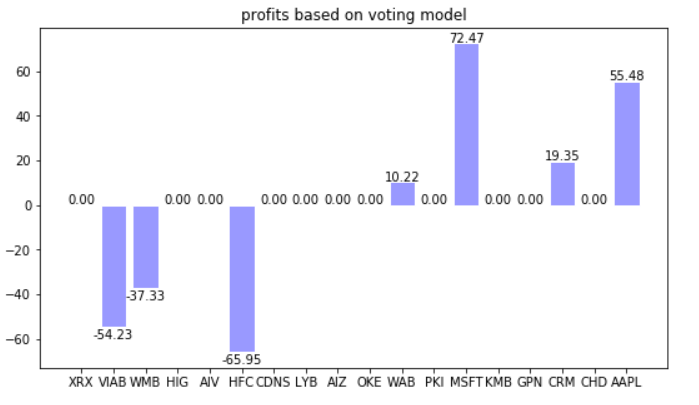
*Figure 6*. The profits of 1000 dollars based on the random forest model

The random forest model seemed to be more conservative than the SVM model. The good news was that it avoided the downward of VIAB and WMB, and capture the huge upward of CDNS, but it also failed to detect the upward of AAPL like the SVM did.



*Figure 7*. The profits of 1000 dollars based on the KNN model

The KNN model performed more progressively than the previous two models. It falsely predicted the movement of low correlated stocks as I expected and correctly predicted the upward of high correlated data. One exception was the PKI which had a relative high correlation score, but the movement was failed to be predicted.



*Figure 8*. The profits of 1000 dollars based on the Voting classifier model

Being the best model in testing accuracy, the voting classifier did not stand out too much more above the other algorithms. From the graph we could tell that the performance was also related to the correlation scores of stocks. The predictions were more accurate on high score stocks.

**RECOMMENDATIONS FOR FUTURE RESEARCH**

To improve the accuracy of predictions, what could be done was training the data on more complex models like deep neural networks which provided more reasonable and efficient ways of doing dimensions reduction and features transformations.

The second suggestion was gathering more informative data with a lot more features, since my current model used only the adjusted closing price as features. My second dataset would be a very good example for this. Features like cash dividends, splitting factors and trading volume were also very important factors needed to be considered while trading the stocks.

There were two obstacles for my suggestions above as well. Firstly, to train with more informative datasets, it required much more work and time to preprocess the whole dataset. Financial companies normally paid lots of money to hire more than one database engineer to clean and preprocess the dataset, and the task was almost impossible for one person alone to finish them.

Secondly, the designing of a perfect deep learning structures that fit the data was very challenging. With all the possibilities out there, I need to do a lot more research studies in order to finally find out a good deep neural network structure for this topic.

**CONCLUSION**

The data stated the fact that correlations did exist between different pairs of stocks even though two companies might seem not related intuitively. The results of four machine learning models proved that they would perform better on the stocks with high correlation scores. The accuracies from the models on the testing set were all better than random choosing, with the 40 percent of voting classifier being the best. The profits based on the four models on highly correlated stocks were good but not very stable.

The four models still needed improvement and were all far away from being used in practice. But if more informative datasets and more reasonable deep learning models were applied, it would be easier to detect the correlations between the stocks and more likely to guarantee the profits.

**BIOGRAPHY**

**Wenye Ouyang** is a graduate student in the Data Science Program at The George Washington University.

He used to work as a data analytics consultant with Deloitte Consulting LLP and Standard Chartered Bank.

His interests include researching financial market and voice recognition with artificial intelligence. He enjoys basketball, music and chess.

**Dr. Nima Zahadat** is a professor of data science, information systems security, and digital forensics. His research focus is on studying the Internet of Things, data mining, information visualization, mobile security, security policy management, and memory forensics. He has been teaching since 2001 and has developed and taught over 100 topics. Dr. Zahadat has also been consultant with the federal government agencies, the US Air Force, Navy, Marines, and the Coast Guard. He enjoys teaching, biking, reading, and writing.

**REFERENCES**

Xiao, Ningchuan. “Machine Learning.” *Wiley Online Library*, American Cancer Society, 6 Mar. 2017, onlinelibrary.wiley.com/doi/abs/10.1002/9781118786352.wbieg0673.

Li, Xiaodong et al. “Stock Prediction via Sentimental Transfer Learning.” *IEEE Access* 6 (2018): 73110–73118. Web.

Campbell, John Y, et al. “Trading Volume and Serial Correlation in Stock Returns.” Cambridge, Mass: National Bureau of Economic Research, 1992. Print.

Santoso Murtiyanto, Sutjiadi Raymond, and Lim Resmana. “Indonesian Stock Prediction Using Support Vector Machine (SVM).” *MATEC Web of Conferences* 164 (2018): n. pag. Web.

Lahmiri, Salim. “Features Selection, Data Mining and Finacial Risk Classification: a Comparative Study.” *Intelligent Systems in Accounting, Finance and Management* 23, no. 4 (October 2016): 265–275.