Predicting the Stock Market One Quarter at a Time

*Note: Sub-titles are not captured in Xplore and should not be used

1st Cory Carter
Computer Science Master of Science Student
Oakland University
Auburn Hills, USA
email address or ORCID

2nd Tom Tisdall
Software Engineering and IT Master of Science Student
Oakland University
Auburn Hills, USA
ttisdall@oakland.edu

***Abstract*—Predicting the stock market is a very challenging task, with hundreds if not thousands of papers addressing this problem. This paper sets out to explore predicting stock quarterly trends by determining the best quarterly features to use and then leveraging them to determine the direction of the upcoming quarterly report. The thought was to eventually marry this with daily trends to improve accuracy of predicting stock market prices by factoring in the quarterly direction, which would be covered in a future paper. By determining the stocks that will go up or down for a quarter, we will predict whether buying or selling is the best course of action for a stock. Testing for accuracy, we will simulate buying and selling in the stock market based on our predictions to determine if this project was a success or failure.**

***Index Terms*—Index Terms - Stock Market, Prediction, Machine Learning**

I. Introduction

Determining the direction of a stock from day to day is both a challenging and frustrating problem. There are so many features to choose from that factor into a stock changing price from day to day. It can be as simple as positive or negative news, changes in the overall market or industry, technological breakthroughs, production hurdles and many more, which don't even cover the technical or quarterly numbers that typically are used to determine a stock's rating. This paper is going to attempt a different tactic on predicting stocks, at a more generic and higher level, using the company's quarterly returns. The thought is this information can be added to other predictors to better refine the accuracy of those models. In this project we have gathered 28 Technology, Software and IT Services large and mega cap stocks with earnings history, typically between ten and thirty years. We started with the features used in the paper Stock Market Trends Prediction after Earning Release [1]. We reviewed and refined those quarterly features by using several feature selection algorithms over the datasets that we obtained, which provided us with the most important columns to include in our machine learning algorithm. We decided to implement the tried and tested Support Vector Machine (SVM) to help us determine what stocks to buy or sell each quarter. Using our model, we then started with a set amount and bought/sold as recommended to see how we faired in the long run.

II. Review of existing techniques for predicting the stock market

For this project, we started by reviewing papers that were published in the realm of stock market analysis via machine learning. One of the first papers we found Stock Market Analysis: A Review and Taxonomy of Prediction Techniques [2] discusses how stock market analysis falls into four categories, statistical, pattern recognition, machine learning, and sentiment analysis. Since this paper focuses on machine learning, we focused on that part of their paper. They and other papers they referenced felt that the stock market can be predicted, the challenge is to determine what information provides the most value in prediction the stock price. In the paper they discussed the algorithm space continously evolving with newer techniques like Random Forrest replacing naïve Bayes and Artificial Neural Networks (ANN), Deep Neural Networks (DNN), and Sentiment Analysis becoming more popular. They note that in cases where transaction fees are not considered that the traditional models perform better, but where transaction costs are included, the DNN models perform better. Bin, Ahmed and Fadel in their paper Stock Market One-Day Ahead Movement Prediction Using Disparate Data Sources [3] found that using data from outside the stock market could help improve the prediction accuracy. In their paper they found using Wikipedia traffic along with stock data helped improve their SVM model. Using sentiment analysis to predict stock price gains and losses seems to be gaining popularity and some success. The article Automated Stock Price Prediction Using Machine Learning [4] used a SVM with traffic visits to Wiki to determine Apples stock price over time at an 85% success rate. Unfortunately their prediction testing only used Apple, it would have been interesting to see how the model performed against other stocks. Interestingly enough they did find that the ordering of the features made a difference in their modeling, reminding us that there are many aspects to defining and developing an accurate model. Automated Stock Price Prediction Using Machine Learning [4] also applied sentiment analysis by using news articles to determine positive or negative trends. In their case they found that using a SVM was the best algorithm. Seeing the benefits of sentiment analysis, this would be something we would like to include in future research.

Changing direction, we found other articles that compared recent techniques such as Particle Swarm Optimization, Least Square Support Vector Machine (LS-SVM) and Artificial Neural Network (ANN) to determine which one was a better fit. A Machine Learning Model For Stock Market Prediction [5] found that using Particle Swarm to optimize Least Square SVM was better than just SVM by itself and that LS-SVM was better than an Artificial Neural Network (ANN) which tended to run into an overfitting problem. Along the same lines An Empirical Study of Machine Learning Algorithms for Daily Trading Strategy [6] compared stocks from the US and Chinese markets across six Machine Learning Algorithms (Logistic Regression (LR), Support Vector Machine (SVM), Classification and Regression Tree (CART) , Random Forest (RF), naïve Bayes (BN) , eXtreme Gradient Boosting (XGB)) and six DNN algorithms (MultiLayer Perception (MLP), Deep Belief Network (DBN), Stacked Autoencoders (SAE), Recurrent Neural Networks (RNN), Long Short-Term Memory (LSTM), Gated Recurrent Unit(GRU)).

They found that ”some traditional ML algorithms have a better performance than DNN algorithms in most of the directional evaluation indicators” and that the best DNN algorithms were not significantly better than the traditional ML algorithms without considering transaction costs. As transaction costs increase, they found that the ML algorithms performance decreased. The DNN models degraded when including transaction costs but at a much slower rate than the ML Algorithms.

Interestingly enough, along the same lines of differences in outside variables like transaction costs, Deep Architectures for Long-Term Stock Price Prediction with a Heuristic Based Strategy for Trading Simulations [7] found that when they created an application to simulate trading based on the predictions of their two models Long Short Term Memory (LSTM) and Convolutional Neural Network (CNN) that LSTM outperformed by gaining a higher total dollar amount, but that the CNN model had a higher number of days with gains. This once again shows that there are many influencers on the stock market that make it challenging to predict. Predicting the daily return direction of the stock market using hybrid machine learning algorithms [8] leveraged 60 financial and economic features showed that using PCA to select features had improved accuracy over those of the entire feature set. They also found that Deep Neural Networks performed better than Artificial Neural Networks when predicting the stock market. It is worth noting that most of the papers we researched ended up with prediction models in the range of 55% to 60% accuracy which is better than a coin flip but typically not better than the buy and hold strategy. In summary we found that modeling the stock market is a hard problem with small improvements being the goal and that it is not purely a mathematical puzzle to be solved.

III. Formulation of our solution

In our work we chose to focus on predicting quarterly results, whether the stock would go up or down and by what percent in the next quarter. For our first step, we gathered quarterly data from a site called StockPup.com, which is now defunct. Fortunately, we gathered our data before they shut down so we were able to perform our testing with the twenty eight stocks. We chose S&P 500 large and mega stocks from the technology sector and software & IT services industry. The data included fourty one columns to use for predicting. Before we could use a formula to select the best columns, we had to normalize and clean the data.

The data cleanup that was done over the stock data consisted of two functions. First, we updated all dates to a year/month/date format from what was originally provided in the files. Next, we cleaned up all of the fields that came with a value of ”none” to support our processing. Once we had that complete, created some new columns to help with our model predictions. We added a quarter start date, forcing the data to fit in windows of quarters. Since the companies all have different quarter end dates, we added quarter year and quarter period columns to fix what quarter the date applied to. We added a few other columns, Last Quarter Price, StdRisk, StdReturn and Class. We used Class to define what degree of percent change for the stock was from the prior quarter. The data did not come with a start or end price, it only had high, low and average, we elected to use the average price to keep the project simple. If we had more time, we would incorporate daily data to correctly include the start and end price of a quarter. In order to mix our stock data when training and testing, we had to normalize the data across stocks. We started that by using a MinMax scale to adjust the columns and then generated change percent columns to represent the change of a value for past quarters.

Once the data was properly cleaned and normalized rather than using all the features to predict we chose to implement several feature selection algorithms to test with. This idea came from the paper Predicting the daily return direction of the stock market using hybrid machine learning algorithms [8] which found that minimizing the feature set improved the prediction accuracy. We decided to start by using the Python Library Sklearn with function SelectKBest. This function provides the ability to choose various algorithms such as f Classification, Mutual Info Classification, Select Percentile and others. Each of these algorithims return the top selected features to be used in our model. We also tested with the algorithm Extra Trees Classifier which we found to produce the best results.

The next step was to classify the data for our model. We implemented a classification category on the average price gain of a stock over several quarters. The thought was to classify the stocks in groups such as those that gained in 1 to 2 percent in a quarter, those that were 2 to 4 percent gains etc. This helped us in a couple of different areas. First, it allowed us to build regression into our calculations by looking at the prior quarters to improve our modeling with SVM. Second, we used it in the real-world scenario of stock purchasing to decide how risky the stock is. This way we could in the final real world test, purchase more of the stocks expected to go up with a bigger gain than the stocks with a minimal gain. To keep things balanced we opted to always buy some of the positive stocks, trying to eliminate some of the volatility that can come with the higher risk stocks.

After selecting our features, we implemented a Support Vector Machine to model and predict our results. We built our test and train sets of data by selecting a year and quarter to divide the data. All data prior to the year and quarter chosen was used to train the model (in random sequence). The data from the chosen year and quarter was used to test the accuracy of our model. We did this rather than splitting the data 80/20 as other processes recommend because we wanted to simulate a real world process. In the real-world we would be predicting the next quarter's price based on the prior quarters.

After building our model and testing the accuracy, we wanted to prove that our model would be profitable in the real world. So, we started by randomly selecting five stocks to remove from the testing phase and spending $10,000 to purchase $2,000 of each stock. We applied the left over to the cheapest stock to use as much of the $10,000 as possible. Then we ran our predictions for each quarter to determine if we should buy or sell stock. To simplify the calculations we rebalanced each quarter, that is sold all the stock and bought what was recommended by our prediction model. To validate our project, we then compared our process with a buy and hold strategy, which we implemented by purchasing $2,000 of each stock.

IV. Experimental Results

Our first step, once we normalized the data was to create a Class that we could use as our target feature. The class was built as a percent of price change calculated over the prior 3 quarters (we selected 3 but could have used any range). The Class was numbered -5 to 5 based on the percent change and price direction from the last few quarters to determine its trend. In our experiment we wanted to purchase stocks that we predicted had a higher gain likelihood over those of a lower expected gain. To do this we broke down the class by percent, 1-2 percent gain would be labeled a 1, 2-3 would be a 2 and so on. This became our target for the feature selection. In our experiments we felt the most important step would be to properly select the features to include in our model. We tested with several SelectKBest algorithms as you can see in the following images. After several rounds of testing we found that selecting the top 10 features seemed to provide the most consistent and highest accuracy. We didn't see as much of a change between selecting the proper features that we expected which we attribute to a small data set and many of the columns being directly related to each other.

Testing with Chi-squared stats of non-negative features (Chi2) 1 which we found that it produced an 86% accuracy. There were several others that also produced an accuracy of 86% includeding Mutual Info Regression and F Classification. We saw 87% with F Classifier (ANOVA F-value), 88% with Extra Trees Classifier. Our highest accuracy based on feature selection was found when we used Mutual Info Classification 2.

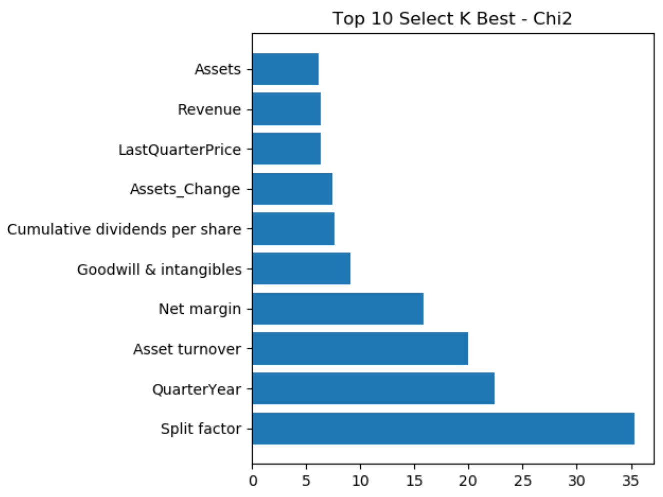

Fig. 1. Top 10 Chi2 - 86% Accuracy

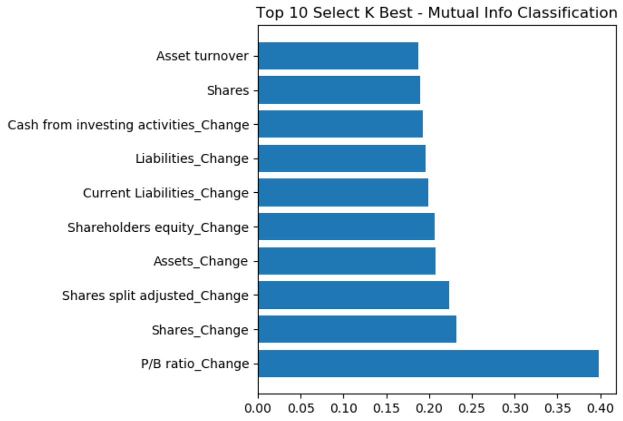

Fig. 2. Top 10 Mutual Info Classification 89% Accuracy

Once we completed the target classification and feature selection, we selected to run the experiment with training data from the beginning of each stock through the last quarter of 2016. The test data was selected as 2017 and forward. In our experiment we found the best accuracy with the Mutual Info Classification selecting its top 10 features, completing at an 89% accuracy rating. We feel this is much higher than it should be, our thoughts are the limited data set along with an industry that has been growing well over the last several years, is skewing the results to a higher accuracy Interestingly enough, when we selected 5 stocks (EBAY, FB, IBM, NFLX and MA) to run our cash analysis with the results were less than spectacular. In fact, we found the buy and hold strategy would produce $17,421.40 where our best strategy using Extra Trees Classifier at 89% accuracy would only achieve around $15,400, around $2,000 less!

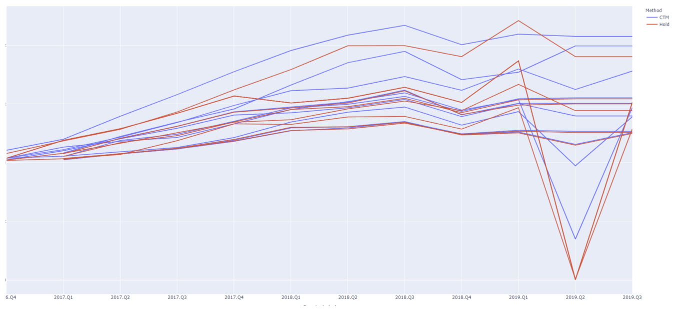

Fig. 3. 10 Runs CTM vs Hold Strategy

V. Summary and Conclusion

This project was a great learning experience covering a challenging if not impossible problem that many have attempted to solve with minimal success. We found that financial data provides many features to select which on one had is good, providing us the opportunity to select the best features. At the same time, we may not have enough features, since the financial data does not include other outside influences such as the market direction, industry news, stock news, company organization changes etc. With our testing we found that using the Extra Tree Classifier selecting ten features provided us the most consistent positive results. Generally though we found that our solution did not significantly outperform the buy and hold strategy. There were a few runs were our mode took three of the top four slots for making money, as you can see in 3

There are several things that we coudl do to improve our project. First having a larger set of data would help stabilize the model and allow us to better predict. We could do this by adding many more stocks quarterly returns to the training and testing data. Second we could tweak the feature selection models and average out many runs to determine the highest accuracy provided. We also wanted to run different prediction algorithms rather than just the SVM one we used. Given more time we would also add features from outside the financial numbers such as news and sentiment analysis to help create a broader picture of characteristics triggering stock movements.

References

[1] C. Qian, W. Zheng, and R. An, "Stock market trends prediction after earning release," *WebSite(Stanford)*, 2016.

[2] I. Shah and Zulkernine, "Stock market analysis: A review and taxonomy of prediction techniques," *International Journal of Financial Studies 7.2*, 2019.

[3] B. Weng, M. Ahmed, and F. Megahed, "Stock market one-day ahead movement prediction using disparate data sources," *ResearchGate*, 2017.

[4] M. Moukalled, W. El-Hajj, and M. Jaber, "Automated stock price prediction using machine learning," *WebSite*, 2019.

[5] O. Hegazy, S. Omar S., and M. Abdul Salam, "A machine learning model for stock market prediction," *International Journal of Computer Science and Telecommunications 4.17-23*, 2013.

[6] L. DongDong, S. Yuan, M. Li, and Y. Xiang, "An empirical study of machine learning algorithms for stock daily trading strategy," *Mathematical Problems in Engineering 2019(7):1-30*, 2019.

[7] C. Stoean, W. Paja, R. Stoean, and A. Sandita, "Deep architectures for long-term stock price prediction with a heuristic-based strategy for trading simulations," *PlosOne*, 2019.

[8] X. Zhong and D. Enke, "Predicting the daily return direction of the stock market using hybrid machine learning algorithms." *Financ Innov 5, 24*, 2019.

[9] J. Xu, "How to use machine learning to become a millionaire predicting the stock market," *Website*, 2019.

[10] A. Edell, "I spent 20 minutes trying to predict the stock market with ai, these are my results," *Website*, 2018.