Stock Market Analysis and Prediction

Eric Alexander, Emily Kawaler, and Dan Szafir

1 Introduction

Stock market analysis is a widely studied problem as it offers practical applications for signal processing and predictive methods and a tangible financial reward. Creating a system that yields consistent returns is extremely challenging and is currently an open problem as stock market prices are extremely volatile and vary widely both within a given stock and comparatively amongst many stocks. Further, stock market data is influenced by a large number of factors including foreign and domestic economies, trade agreements, wars, seasons, and even day of the week [12, 3]. Any trading strategy must of necessity balance desire for immediate returns with the possibility of larger payoffs in the future, and many different approaches towards prediction have been attempted including neural networks and fuzzy reasoning [1], support vector machines [9], and even attempting prediction using data-mining techniques over textual data in financial news [14]. In this paper, we examine the abilities of linear regression, random forests, and support vector machines (SVM) with SMO to predict future prices and trends in a variety of stocks.

2 Related Work

Although many different methods of predicting stock prices have been considered, none have become widely accepted which offers a testament to the difficulty of the problem. Although a simple goal (buy when prices are low, sell when prices are high) seems obvious, figuring out when prices are low and when prices are high is by no means a trivial problem. It is often easy to spot ideal buying and selling points in post-hoc analysis, however the fluctuation of prices makes spotting and predicting trends in realtime a challenge. Below we describe the various predictive methods we attempted to solve this problem.

2.1 Linear Regression

One of the key difficulties of modeling and predicting stock data is the fact that it is represents data that is not independently and identically randomly distributed which violates a key assumption behind many machine learning methods. Instead, it represents a time-series collection of points where each value is somewhat dependent on some number of previous days' values, as well as other invisible factors such as company performance and economic robustness. Although it has been suggested that stock price changes are largely independent [2], more recent research has shown regression to be somewhat successful strategy, particularly when used to bolster other input such as neural networks [10]. Further analysis has offered the possibility of a nonlinear unconditional dependence of stock market data that could possibly be captured with nonlinear regressive techniques [11]. We wanted to examine the capabilities of purely regressive techniques to see if they would allow us to capture both local peaks and nadirs corresponding to ideal buying and selling points as well as predicting future prices solely from training data in the form of previous prices. We created a two-tiered system using regressive techniques that represents a pseudo-ensemble: (1) Use an exponential moving average to predict future prices and (2) Use a combination of least squares regression as well as derivative analysis to create a realtime trending threshold to identify times to buy and sell.

Exponential Moving Averages (EMAs) decrease the weights given to older events exponentially and are often used for smoothing noisy data [5]:

$$S(t) = \begin{cases} Y(t) & : t = 1 \\ a * Y(t-1) + (1-a) * S(t-1) & : t > 1 \end{cases}$$

where S corresponds to the value produced using an EMA, t is time, Y is the stock market price, and a is a regularization constant that is inversely proportional to the relative weighting of less recent events. A larger value of a provides a closer fit to the original data, and a value of .5 was used for this study after several tests on various stocks as we want to closely model the actual stock price (see Figure 1. One downside to using an EMA is that it only allows the prediction of stock price one day in advance, however standard linear regression could be used to extend this prediction if needed.

To establish a consistent means of trending (locating peaks and nadirs), two thresholds were created from the stock data in real time and polled for possible future losses. The first threshold level compares the average slope of the data within a given timeframe with the average slope seen so far:

$$DT_B(S) = \begin{cases} 1 & : \frac{d\bar{S(x)}}{dt} < \frac{d\bar{S(y)}}{dt}, \frac{d\bar{S(x)}}{dt} < 0 \\ 0 & : otherwise \end{cases}$$

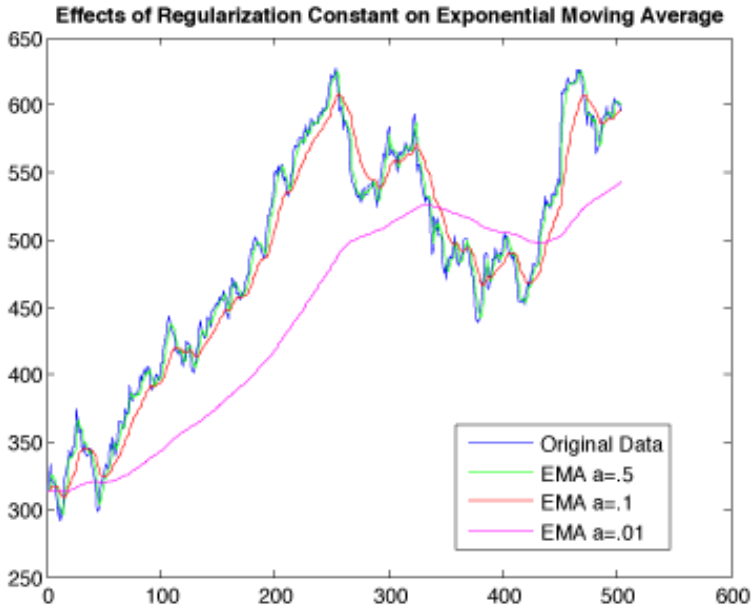


Figure 1: A given stock (blue) with various exponential moving averages computed using differing regularization constants.

$$DT_S(S) = \begin{cases} 1 & : \frac{d\bar{S(x)}}{dt} > \frac{d\bar{S(y)}}{dt}, \frac{d\bar{S(x)}}{dt} > 0 \\ 0 & : otherwise \end{cases}$$

where x corresponds to the timeframe (a given number of days), y represents the total number of days seen so far, S is the stock price, and DT_B represents the derivative threshold determining function for buying and DT_S represents the derivative threshold determining function for selling. The buying function looks at the most recent data and determines if the prices have been declining at a faster rate than on average, while the selling function looks at the most recent data and determines if prices have been inflating.

The second threshold level is based on Least-Squares Regression (LSR) which minimizes a function F where:

$$F = \sum_{i=1}^{n} (y_i - (ax_i + b))^2$$

such that

$$\frac{\partial F}{\partial a} = 0 \text{ and } \frac{\partial F}{\partial b} = 0$$

where F corresponds to the minimized function which generates slope a and intercept b around the data represented by x (time) and y (stock price) from the

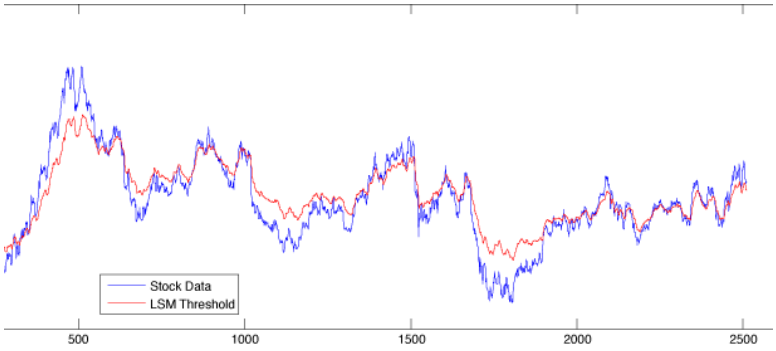


Figure 2: A given stock (blue) with a generated LSR threshold used in trending prediction. This prediction aided in guessing whether it was a good time to sell (stock price is above the threshold) or buy (stock price is below threshold)

start to time n. To generate the second threshold, multiple LSR functions are generated: one based on the past 3 days, one based on the past week, one based on the past month, and one based on all the data seen so far. These are then averaged and weighted (weights were determined following testing on various stocks):

$$T = .4 * F(t) + .3 * F(m) + .2 * F(w) + .1 * F(r)$$

where T is the generated threshold, $F(t)$ is the function generated by LSR around all data seen so far, $F(m)$ is the function generated using the past month's data, $F(w)$ used the past week's data, and $F(r)$ is the LSR function based on the data in the latest 3 day interval (see Figure 2). This creates a constantly updating "average" level for stock values.

2.2 Random Forests

Random forests are a variation on bootstrap aggregation algorithms developed by Leo Breiman [4]. In this algorithm, one creates multiple decision trees in the following manner:

- Create a bootstrap training set by drawing N examples from the training data, with replacement (where N is the size of the training set).
- Choose a random i features (where i is some number much less than the number of features), and split upon the one of these that gives the best information gain (change in entropy from one level of the tree to the next).
- Continue recursively building this decision tree until each node is completely pure, without performing any pruning.

The "forest" aspect of the algorithm happens after one creates a large number of these decision trees, each using a different bootstrap set. In order to classify a given feature vector, all of these trees "vote" on what its classification should be.

2.3 SVM with SMO

Recently, support vector machines (SVM) [6] has become one of the most widely-used algorithms in machine learning. SVM utilizes a hyperplane to separate two classes of data points, while attempting to maximize the margin between that hyperplane and the nearest data points (called support vectors). This cannot always be done in the given set of dimensions, so a kernel may be used to transform the points into higher-dimensional space.

Normally, to train an SVM requires solving a complex optimization problem, which can be expensive both computationally and financially (it can be very costly to obtain a license for a quadratic programming solver). Platt's sequential minimal optimization algorithm (SMO) is able to reduce the large QP problem to a series of smaller subproblems that can be solved analytically. See [13] for technical details.

The Weka machine learning dataset [8] implements a version of SVM called support vector regression (SVR). SVR allows for the use of SVM in regression problems. For more information on SVR, please see [7].

3 Methodology

3.1 The Agent

To test our predictive abilities, we set up a pseudo-stock exchange and created various "agents" that acted as stock brokers. The exchange used real world stock values downloaded from Yahoo Finance which included opening prices, daily highs and lows, and volume traded for each date. Agents were first given a year's worth of this data to train over and $10,000.00 to start with, and for each new day in the testing data, the agent was asked whether they would like to buy or sell, how many shares to trade, and at what price. The exchange would not allow agents to sell higher than the daily high or sell less than the daily low (the agent had no prior knowledge of what these prices would be), and charged a $10.00 commission fee to mimic real-world exchanges. Once the day was over, the exchange updated the agent with that date's data, allowing the agent to combine test and training data to further update its model after each day for more accurate predictions. At the end of testing, the agent was forced to sell any remaining stocks at the average market price on the last day. We tested agent performance both in the short term

(train on one year, test on one year) as well as the long term (train on one year, test over the next 10 years). We used IBM, Intel, Apple, and Google stocks as well as NASDAQ data for our datasets.

3.2 Random Forests

In practice, random forests have been shown to be very accurate and reliable. In our application of them, the trickiest obstacle is actually creating the feature vectors themselves. Historical prices of stock market data do not easily translate into a set of features and a classification. Nonetheless, we were able to fit it into this framework. We ultimately created a dataset in which each datapoint represented a particular stock on a particular day. The class of each datapoint was a boolean value representing the direction the stock would take on the following day. Into the features, we built historical data about the directions the stock had taken in the past. These were grouped such that more recent data was given extra weight so as not to have minor fluctuations in the past affecting our current-day predictions. Specifically, we had features representing the percent increase (or decrease) in price and volume over the previous day, the previous three days, and the previous week. We also incorporated the same data from NASDAQ (the stock index of which all of our stocks are a part), since it stands to reason that the performance of the market as a whole should affect the performance of individual stocks. In addition, when measuring performance of the Apple stocks, we looked at the performance of the Microsoft stocks as well, the theory being that the stock performance of a similar company might reflect real-world events that could have a bearing on the performance of our target stock.

Once we had converted our data into feature vectors of this type, we created a random forest of 100 trees.

3.3 SVM with SMO

We used SVM in three different ways, as follows:

- First, on the Apple 2009 dataset. We used ten-fold cross-validation so that the data could be both training and test set.
- Second, we used the Apple 2008 dataset as a training set and the Apple 2009 dataset as the test set. This was to see if we could use a stock's data from the previous year to accurately predict its behavior.
- Finally, we used Weka's built-in time series predictor, again training on the 2008 dataset and testing on the 2009 dataset, to see if that worked any better. This predictor uses SVR as a base classifier.

For the first two, we used the same feature vectors as in the Random Forest section, with the standard SVM/SMO algorithm. Our aim here was simply to test whether we could predict whether a stock's price would go up or down for the day. For the third method, we used a simpler dataset consisting only of the information about the Apple stock (its opening price, its trading volume, its high, its low and the net change for each day). The time series predictor automatically transforms these features so that for each day, twelve days' worth of lag are represented. In this way, it essentially builds an ersatz time series out of simple feature vectors (similar to what we did with our own feature vectors), and runs SVR on that. We hoped to be able to accurately predict the magnitude and direction of the stocks' price swings with this method.

In all cases, we used a radial basis function (RBF) kernel with a C of 1 and a gamma of 0.01.

4 Results

Our results showcase that, although prediction of price and trending is extremely challenging, it is possible to make an intelligent agent that can make money in volatile markets.

4.1 Linear Regression

Utilizing linear regression to consistently make a profit across a wide range of stocks proved challenging. Various values for the thresholds were tested and often resulted in extremely high performance on one stock, but dismal performance on others. This tradeoff was due to competing strategies that emerged from tuned threshold values: one strategy favors short term gains with lots of trading and small profits (Figure 3), the other is interested in long term gains and prefers to wait to buy until it reaches an expected global minima and sell at a global maximum (Figure 4). Tuning came about in the form of adjusting the EMA regularization constant, the timeperiod for the derivative threshold, and the weights given for each LSR value. Once tuned (which could come about in real scenarios in the form of domain specific expert knowledge), the regression agent was able to perform reasonably well across multiple stocks. Regression results are summarized in Table 4.1 with an example run shown in Figure 5.

4.2 Random Forests

The random forest algorithm performed rather poorly when merely tested for the accuracy of its predictions of the direction of a given stock on the following day.

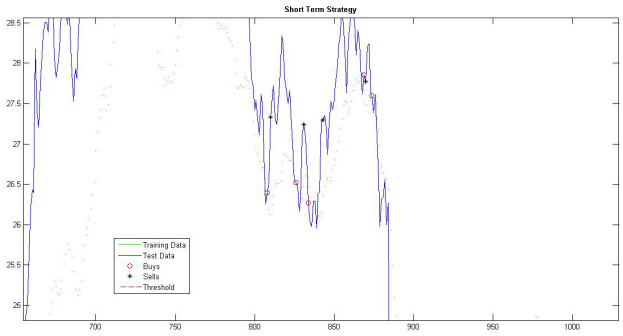


Figure 3: One strategy favored buying and selling at local minima and maxima. This short term strategy made little profit each time, but made many more transactions.)

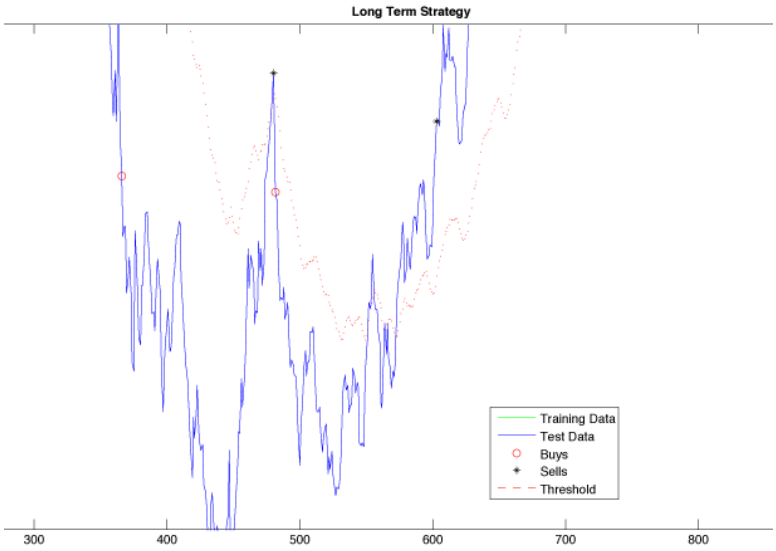


Figure 4: A competing strategy favored making fewer trades and waiting to maximize profits buy buying at more global minima and selling at more global maxima.)

Training Data	Test Data	Gain (dollars)	Predicted Price Accuracy
IBM 2001	IBM 2002-2011	905.22	.6219
Intel 2001	Intel 2002-2011	895.26	.5968
Apple 2001	Apple 2002-2011	4085.41	.5841
Google 2004	Google 2005-2011	6949.56	.5559

Table 1: Results of agent using regression strategies over several datasets

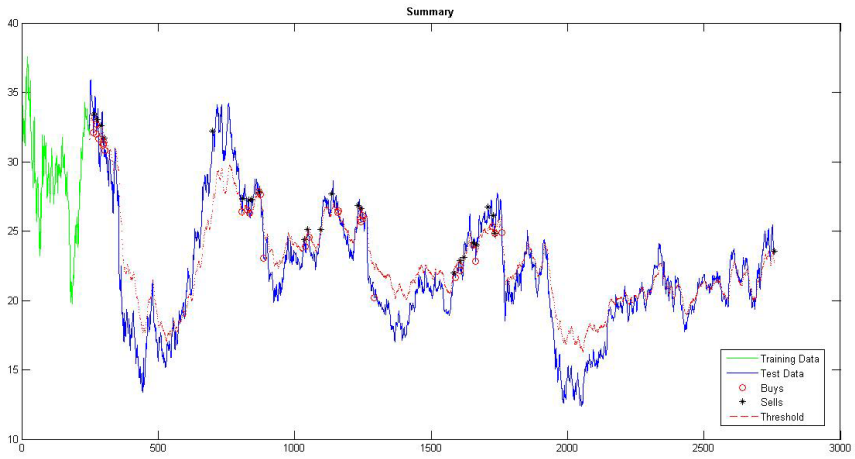


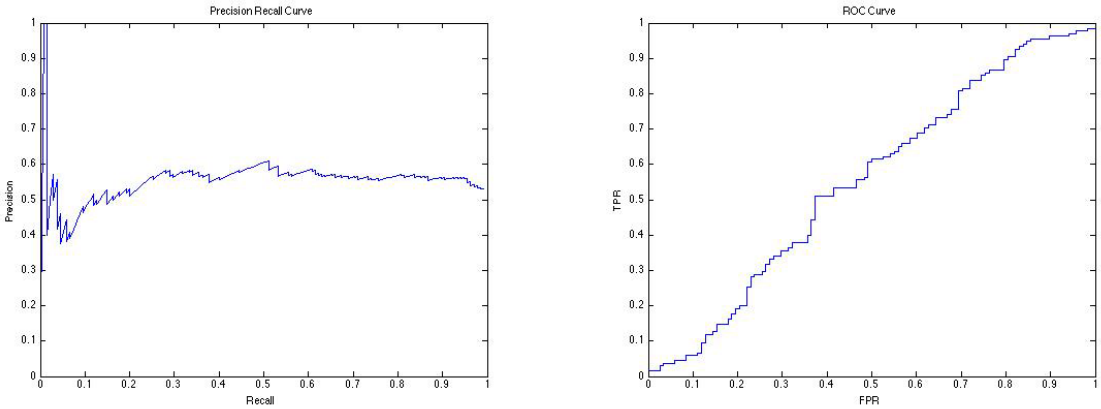
Figure 5: Results of the regression agent on a sample data set. Here the agent trained over Intel 2001 data and was tested over Intel 2002-2011 data

For these tests, the algorithm had a fluctuating accuracy that was usually above fifty percent, though it did drop below. Unfortunately, unlike the low accuracies of the predictions given by the linear regression agent, this did not translate into significant financial gain. When an agent trained on data from 2002 was set loose in the market data from 2003 to the present, it held steady, never gaining more than five percent and sometimes losing just as much of its initial capital. If loosed in the real market, the random forest agent would be a less lucrative investment option than a typical savings account.

4.3 SVM with SMO

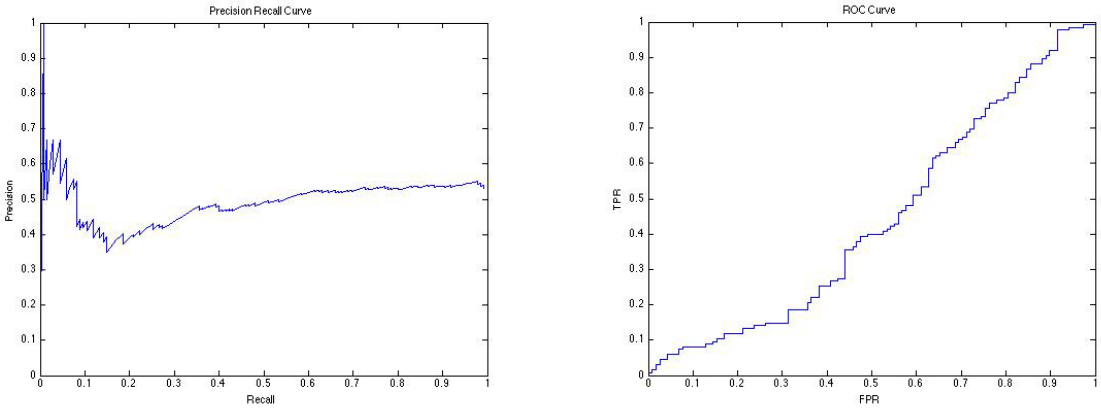
Much like the random forest algorithm, SVM proved to be not very useful in predicting stock values. For methods 1 and 2, the algorithm outputs a confidence for each day's prediction, going from 0 (stock will certainly decrease in value) to 1 (stock will definitely increase in value), so pulling out precision/recall/FPR numbers is a simple matter of organizing the predictions in order of confidence and setting different thresholds. As shown in Figure 6 and Figure 7, the algorithm performs very poorly, to the point where the accuracy is very close to random guessing. (In the 2009 dataset, the stock value goes up on 135/253 days, or roughly 53%.) As might be expected, the performance was slightly better when the 2009 dataset was trained on itself. This could be explained away by saying that 2008 was an abnormal year for the stock market as a whole, but it is very likely that this result would be the same no matter which set of training and testing years we might use.

The third method was slightly more interesting. Recall that this method consisted of using SVR on a transformed set of feature vectors using 12 days of lag for each day to simulate a time series. Though training was done on the 2008 set and testing on 2009, results were better, providing 64% accuracy on direction alone. Attempts to predict magnitude, however, were less successful. In Figure 8, one possible reason for this can be seen. The green curve represents the actual prices of Apple stock over the year. It fluctuates wildly from day to day. The SVR predictions, represented by the blue curve, are significantly more conservative. This most likely has to do with the fact that it is attempting to make predictions using several days worth of data, which often represent both ends of the spectrum from high to low. However, since the other two SVM methods were in effect using smaller lag and performed worse, lessening the lag here would not improve accuracy much if at all.



(a) Precision Recall Curve (b) ROC Curve

Figure 6: Precision Recall and ROC curves for the first SVM method (using cross-validation)



(a) Precision Recall Curve (b) ROC Curve

Figure 7: Precision Recall and ROC curves for the second SVM method (training on 2008 dataset, testing on 2009)

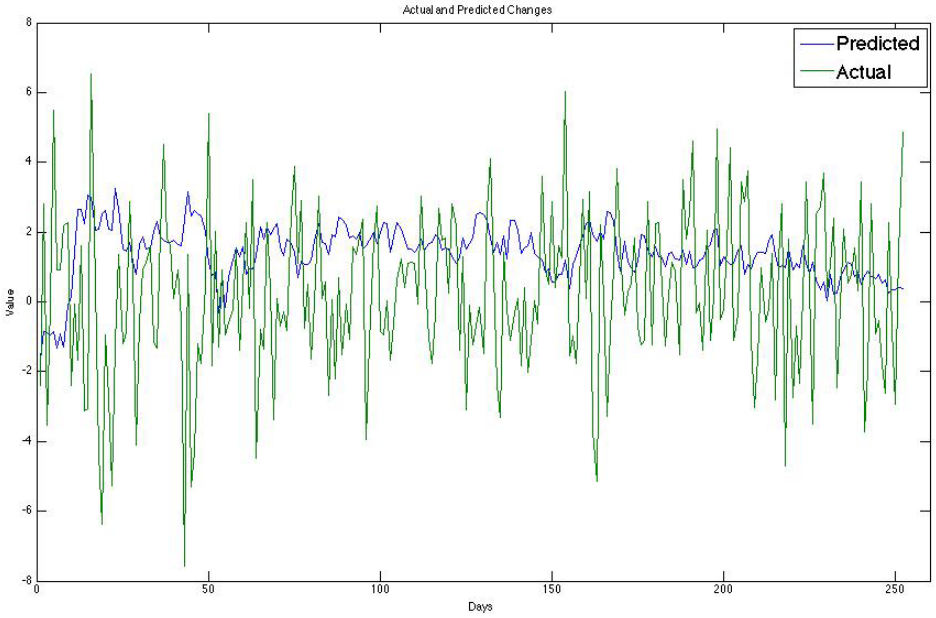


Figure 8: Shows predicted (by SMR) and actual values for change in stock value over the year 2009.

5 Discussion and Future Work

Although stock market prediction is extremely difficult, it provides an interesting and potential fiscally rewarding area of research. Our methods provide a starting point for future market analysis on which there are several possible areas for improvement.

We found the success of our relatively simple regressive techniques to be surprising given the relatively low accuracy in predicting the next day's price, the extremely limited features that they examine, and the possibility that stock prices are independent from one another. It is possible that these methods were over-fit to our training examples, and would not be generalizable to other stocks. One possible means of addressing this issue would be combining linear regressive methods with other machine learning techniques such as neural networks or support vector machines which could learn in real time the various tuning weights as opposed to having static weights found by hand. A further avenue of research would be polynomial regressive techniques such as Lagrange/Barycentric interpolation. Although we initially examined this as a possibility, we found very poor predictive measures when using more than three stock market data values which is likely due to Runge's phenomenon. It is possible that this problem could be mitigated by using Chebyshev nodes, however, we have not had time to explore this avenue of research yet.

The largest problem we see with the random forest application is the same one we saw going into the experiment: the creation of a viable feature vector. We attempted to include historical data about the stocks as well as historical data about similar stocks in our features, and to aggregate this data over increasing lengths of time. Ultimately, this failed. The assumptions we made in creating these feature vectors represent the best of our background knowledge, which is admittedly low—perhaps a true expert would be able to come up with more relevant features. However, there still seems to be an inherent incongruency in treating the data as independent datapoints at the same time that we encode historical data (clearly dependent) into the features. In addition, though accurate predictions of market direction are certainly valuable, they are obviously less so than the real-value price predictions of our linear regression models. Given this comparison, we will likely abandon this algorithm in future work in favor of the other models we have described.

Surprisingly, support vector machines did not work well on this problem either. Again, like random forest, perhaps this failure is a function of insufficient feature vectors, but it may also be a result of the inherent randomness in the stock market. Just looking at the variation between successive days in Figure 8, it becomes clear that finding a kernel to accurately represent this data without overfitting might be a very difficult task, and perhaps it is the kernel that causes the

problem here. Either way, this algorithm certainly performed very poorly when compared to the regressive model.

An interesting direction in which to take this in the future, and one that might help the latter two algorithms, is to focus on creating a better feature vector. To start with, we could take a closer look at how including different time lag affects predictions. In our case, the twelve days of lag outperformed our shorter lag, but there's a lot of middle ground there. We would also like to add features that capture information which was not in our dataset at all. For instance, real-world events often greatly affect the stock market. An expert might be able to look at the news and predict how that would affect a stock (for instance, during the recent Netflix/Qwikster debacle, the stock plummeted, which most industry experts saw coming but which is not something that could have been predicted by looking at the past data). It might be helpful to somehow add a variable that represents this outside influence, but this is something we were unable to do.

As for the algorithms we have discussed in this paper, random forests and SVM/SVR both fell flat, with accuracy roughly matching that of pure guessing. Regression is clearly the strongest method, showing higher predictive accuracy and an ability to actually make money in a real-world situation.

References

[1] Ajith Abraham, Baikunth Nath, and P. Mahanti. Hybrid intelligent systems for stock market analysis. In Vassil Alexandrov, Jack Dongarra, Benjoe Juliano, Renao Renner, and C. Tan, editors, *Computational Science - ICCS 2001*, volume 2074 of *Lecture Notes in Computer Science*, pages 337–345. Springer Berlin / Heidelberg, 2001.

[2] Eugene F et al, Fama. The adjustment of stock prices to new information. *International Economic Review*, 10(1):1–21, 1969.

[3] Hakan Berument and Halil Kiymaz. The day of the week effect on stock market volatility. *Journal of Economics and Finance*, 25:181–193, 2001.

[4] Leo Breiman. Random forests. *Machine Learning*, 45:5–32, 2001. 10.1023/A:1010933404324.

[5] Robert Goodell Brown. *Smoothing, forecasting and prediction of discrete time series*. Prentice-Hall Internat. Ser. in Manag. Prentice-Hall, Englewood Cliffs, NJ, 1962.

[6] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20:273–297, 1995.

[7] H. Drucker, C.J.C. Burges, L. Kaufman, A. Smola, and V. Vapnik. Support vector regression machines. In M.C. Mozer, M.I. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems 9*, pages 155–161. [MIT P]ress, 1997.

[8] M. Hall, F. Eibe, G. Holmes, B. Pfahringer, P Reutemann, and I.H. Witten. The [weka] data mining software: An update. *SIGKDD Explorations*, 11, 2009.

[9] Wei Huang, Yoshiteru Nakamori, and Shou-Yang Wang. Forecasting stock market movement direction with support vector machine. *Computers & Operations Research*, 32(10):2513 – 2522, 2005.

[10] Blake LeBaron, W.Brian Arthur, and Richard Palmer. Time series properties of an artificial stock market. *Journal of Economic Dynamics and Control*, 23(9-10):1487 – 1516, 1999.

[11] Esfandiar Maasoumi and Jeff Racine. Entropy and predictability of stock market returns. *Journal of Econometrics*, 107(1-2):291 – 312, 2002.

[12] Richard Roll Nai-Fu Chen and Stephen A. Ross. Economic forces and the stock market. *The Journal of Business*, 59(3).

[13] John C. Platt. Sequential minimal optimization: A fast algorithm for training support vector machines. *Technical Report MST-TR-98-14*, 1998.

[14] Robert P. Schumaker and Hsinchun Chen. Textual analysis of stock market prediction using breaking financial news: The azfin text system. *ACM Trans. Inf. Syst.*, 27:12:1–12:19, March 2009.