Stock Market Price Prediction Using Linear and Polynomial Regression Models

Lucas Nunno
University of New Mexico
Computer Science Department
Albuquerque, New Mexico, United States
lnunno@cs.unm.edu

***Abstract*—The following paper describes the work that was done on investigating applications of regression techniques on stock market price prediction. The report describes the linear and polynomial regression methods that were applied along with the accuracies obtained using these methods. It was found that support vector regression was the most effective out of the models used, although there are opportunities to expand this research further using additional techniques and parameter tuning.**

***Keywords*-stock market; regression; machine learning;**

I. Introduction

The stock market is known to be a complex adaptive system that is difficult to predict due to the large number of factors that determine the day to day price changes. We do this in machine learning through regression which tries to determine the relationship between a dependent variable and one or more independent variables. Here, the independent variables are the features and the dependent variable that we would like to predict is the price. It is apparent that the features that we are using are not truly independent, we know that the volume and outstanding shares are not independent as well as the closing price and the return on investment not being independent. However, this is an assumption that we are making to simplify the model in order to use the chosen regression models.

This study aims to use linear and polynomial regression models to predict price changes and evaluate different models' success by withholding data during training and evaluating the accuracy of these predictions using known data.

This research concerns closing prices of stocks, therefore day trading was not modeled. The model for the stock market was only concerned with the closing price for stocks at the end of a business day, high-frequency trading is an area of active research, but this study preferred a simplified model of the stock market.

II. Motivation

Stock market price prediction is a problem that has the potential to be worth billions of dollars and is actively researched by the largest financial corporations in the world. It is a significant problem because it has no clear solution, although attempts can be made at approximation using many different machine learning techniques. The project allows techniques for real-world machine learning applications including acquiring and analyzing a large data set and using a variety of techniques to train the program and predict potential outcomes.

III. Related Work

A variety of methods have been used to predict stock prices using machine learning. Some of the more interesting areas of research include using a type of reinforcement learning called Q-learning [5] and using US's export/import growth, earnings for consumers, and other industry data to build a decision tree to determine if a stock's price will rise or fall [3].

The Q-learning approach has been shown to be effective, but it is unclear how computationally intensive the algorithm is due to the large number of state alphas that must be generated. The decision tree approach may be particularly useful when analyzing a specific industry's growth. There has also been research done as to how top-performing stocks are defined and selected [7] and analysis on what can go wrong when modeling the stock market with machine learning [4].

IV. Methods

A. Data Representation

The dataset that was used was collected from the CRSP US Stock Database [2] as a collection of comma-separated values where each row consisted of a stock on a specific day along with data on the volume, shares out, closing price, and other features for that day in time.

The Python scientific computing library numpy was used along with the data analysis library pandas in order to convert these CSV files into pandas DataFrames that were indexed by date. Each specific stock is a view of the master DataFrame that is filtered based on that stock's ticker. This allowed efficient access to stocks of interest and convienient access to date ranges.

These stock DataFrame views are then used as the data to be fed into our regression black boxes.

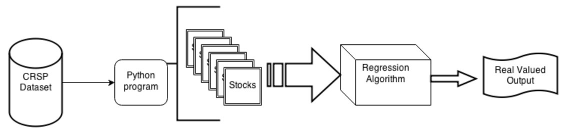

Figure 1: Data-flow of the program showing how stock data turns into prediction value vectors.

B. Prediction through Regression

The regression process is done through the scikit-learn [1] machine learning library. This is the core for the price prediction functionality. There are some additional steps that must be done so that the data can be fed into the regression algorithms and return plausible results. In particular, every training dataset must be normalized to a Gaussian normally distributed or normal-looking distribution between -1 and 1 before the input matrix is fit to the chosen regression model.

1) Data Normalization: There are a couple important details to note about the way the data must be preprocessed in order to be fit into regression models. Firstly, dates are normally represented as strings of the format "YYYY-MM-DD" when it comes to database storage. This format must be converted to a single integer in order to be used as a column in the feature matrix.

This is done by using the date's ordinal value. In Python, this is quite simple. The columns in the DataFrame are stored as numpy datetime64 objects, which must be converted to vanilla Python datetime objects which are in turn converted to an integer using the toordinal() built-in function for datetime objects.

Each column in the feature matrix is then scaled using scikit learn's scale() function from the preprocessing module. Note that this is a very important step, as prior to this, the polynomial regression methods would return questionable results since it is documented that the scikit learn's non-linear regression models assume normally distributed data as the input for feature matrices.

2) Types of Regression Models: The price prediction function provides a few regression models that can be chosen to perform the prediction. This includes

1) Linear Regression
2) Stochastic Gradient Descent (SGD)
3) Support Vector Regression (SVR)

Please note that these were the regression models that were evaluated, not all had promising results. The results section goes into detail of the difficulties faced with each regression model and the attempts at the solutions.

The linear regression method initially seemed to be working well, but there were some difficulties using the polynomial regression methods, since the predictions that are being returned did not look like a non-linear fitting.

C. Stock and Date Selection

The stocks that are used in this study are a subset of the stocks that are publicly traded on the US market known as the S&P 500, which is an index of the 500 largest companies traded on the NYSE or the NASDAQ. The CRSP dataset that has been provided contains on the order of 5000 companies, so this data is filtered as it is loaded into the master DataFrame to avoid excessive memory usage.

D. Regression Model Evaluation

There are a number of scoring methods for regression that are implemented in scikit learn, such as explained variance score and mean squared error.

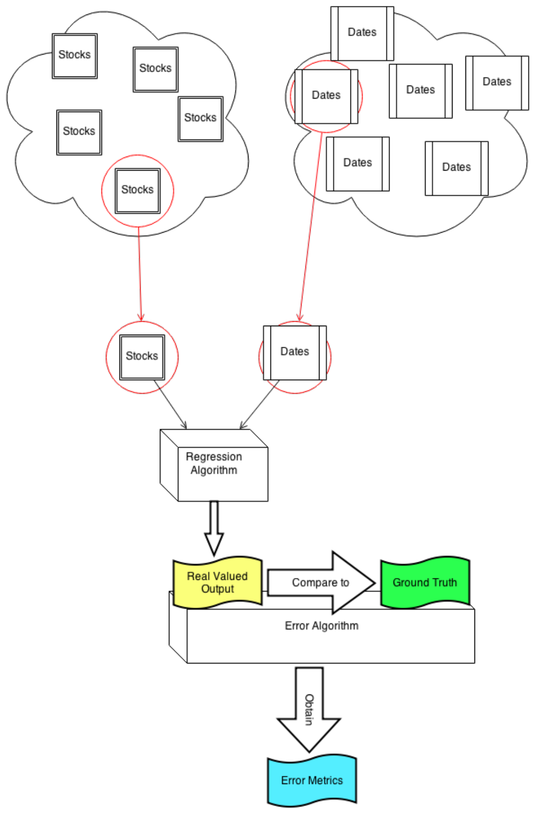

Figure 2: An illustration of the random sampling of both stocks and dates done by the software in order to obtain error metrics. These metrics are then compared against other regression models' results to evaluate their performance.

While these algorithms were investigated, it seemed to be more beneficial for this problem domain to implement mean absolute percentage error and use this in order to compare stocks of significantly different prices. Mean absolute error was used in the context of inspecting a single stock, where price difference was bound not to vary as much as comparing to disparate companies.

V. Results

Note that the following diagrams consistently use Apple's (AAPL) stock prices from 2006-11-16 to 2007-03-27. This is strictly for comparison reasons and to consistently compare regression methods with the same data. This date range was chosen specifically for its troughs and plateau features present in the date range since it would provide a sufficiently challenging topography for the regression models and for human experts as well. The evaluation described later uses a random sampling of stock tickers and dates.

A. *Linear Regression*

Linear regression was less sensitive to normalization techniques as opposed to the polynomial regression techniques. Some plausible results were appearing early on in the study even when a small number of features were used without normalization, while this caused the polynomial regression models to overflow. Linear regression also provided plausible results after normalization with no parameter tuning required due to its simplified model, although the accuracy was less than would be desired if relying on the results for portfolio building.

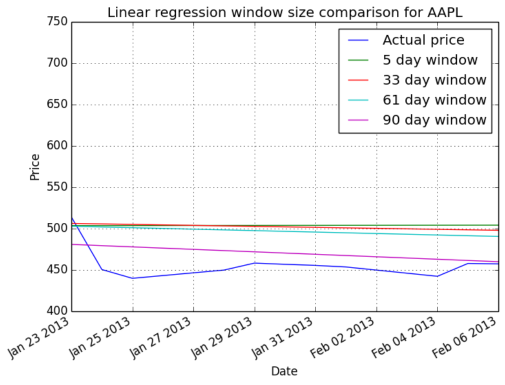

Figure 3: Price prediction for the Apple stock 10 days in the future using Linear Regression.

It is interesting how well linear regression can predict prices when it has an ideal training window, as would be the 90 day window as pictured above. Later we will compare the results of this with the other methods

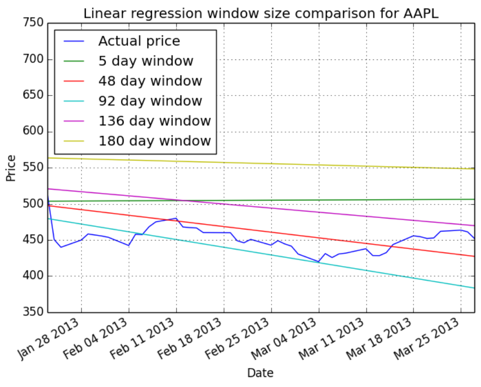

Figure 4: Price prediction for the Apple stock 45 days in the future using Linear Regression.

Large training windows appeared to overfit for larger prediction windows, as can be seen by Figure 4. However, it appeared to be more accurate in instances where the price deltas were consistent with the price trends over that same period for relatively short buying periods over a couple of weeks as seen in Figure 3.

B. *Stochastic Gradient Descent (SGD)*

At first, it appeared that Stochastic Gradient Descent would be an appropriate fit to a problem of this type for long term price prediction. However, since the dataset that was used only covered the time period of 2005-2013 the training data could only provide a maximum of $(365 * 8) = 2920$ training samples to be used. Obviously, the stock exchange is not open every day of the year, therefore this number would be significantly lower. This appears to be a problem according to the algorithm's documentation source, since it is only recommended to be used for problems with a training set size of greater than 10,000.

The scikit-learn documentation mentions this with a few suggestions for alternatives. [1]

> The class SGDRegressor implements a plain stochastic gradient descent learning routine which supports different loss functions and penalties to fit linear regression models. SGDRegressor is well suited for regression problems with a large number of training samples ($> 10,000$), for other problems we recommend Ridge, Lasso, or ElasticNet.

Further along in the paper, we will investigate some of the alternatives mentioned above, but this is also an opportunity for future research on linear methods applied to this domain.

C. *Support Vector Regression (SVR)*

The scikit-learn documentation has an illustrative figure of the differences of available kernels when using Support Vector Regression. Below is the figure that shows what kind of fitting is done using various kernels, note the difference between the radial basis function (RBF) kernel and the other two. [1]

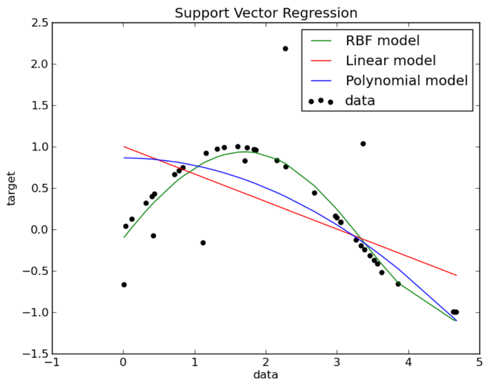

Figure 5: Support Vector Regression data-fitting with a rbf, linear, and polynomial kernel on a set of normally distributed data with random noise introduced into the data-set.

1) Using the Polynomial Kernel: The degree of the polynomial is by default set to 3, this setting was used for the radial basis function as well.

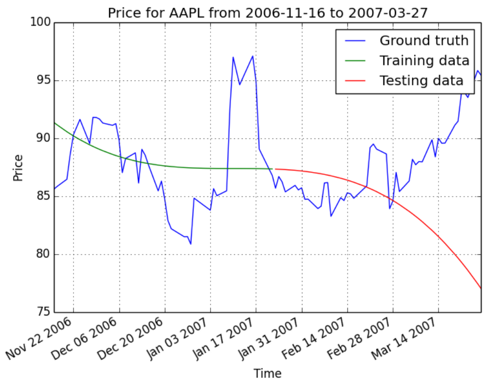

Figure 6: Sample result of using the polynomial kernel with the SVR. This data was trained on the previous 48 business day closing prices and predicted the next 45 business day closing prices.

From the multiple trials performed, the polynomial kernel tended to have better predictions for a subset of the testing data, but then would tend to diverge abruptly from the ground truth at varying periods. This behavior can be seen above, where it diverges around the February 28th 2007 data point.

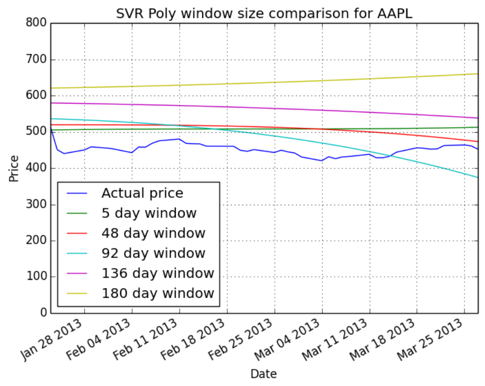

Figure 7: Window size comparison for SVR using the polynomial kernel.

2) Using the RBF Kernel:

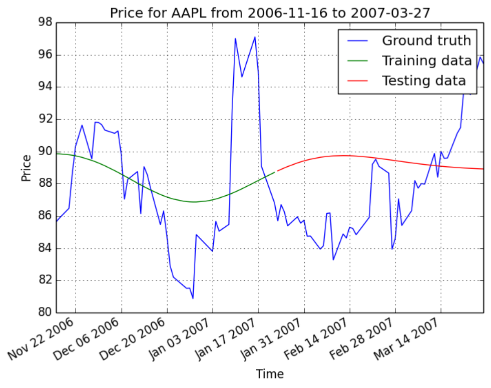

Figure 8: Sample result of using the RBF kernel with the SVR. This data was trained on the previous 48 business day closing prices and predicted the next 45 business day closing prices.

The RBF kernel tended to fix this divergent behavior that we were consistently seeing with the polynomial kernel. However, it seemed to come at the cost of not as accurate predictions at the beginning of the test data. Overall, the RBF kernel performed the best on average for each day that it was tested on. It is important to note that it doesn't mean that its results were always the most accurate. This depended quite heavily on the training window, as linear and polynomial regression were able to have more accurate predictions than SVR with the RBF kernel. However, SVR with the RBF kernel was the most consistent overall, therefore it's important to make this distinction.

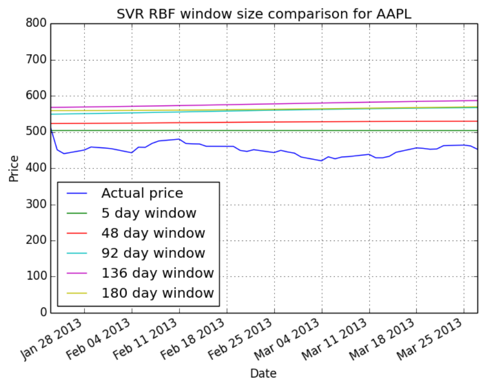

Figure 9: Window size comparison for SVR using the RBF kernel.

Support vector regression with the rbf kernel was not very sensitive to window size changes, which is very different than linear and SVR with the polynomial kernel; which were both very sensitive to window size changes.

D. Summary and Comparison

Below is a superimposed version of all the regression methods discussed previously.

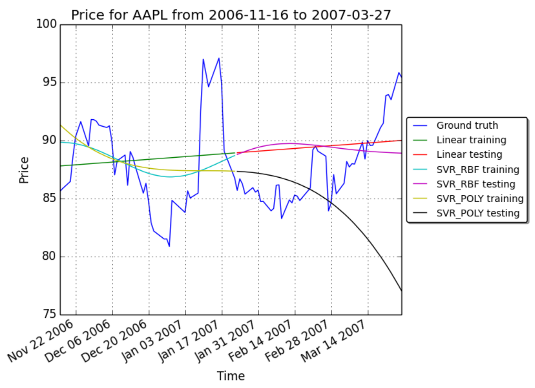

Figure 10: Comparison of several regression methods of a single stock on a fixed time-frame and their training and testing models visualized.

The following is the error of each of the regression methods from the superimposed figure above. Note that error here is measured in dollar amount, not percentage as it will be in the following figures.

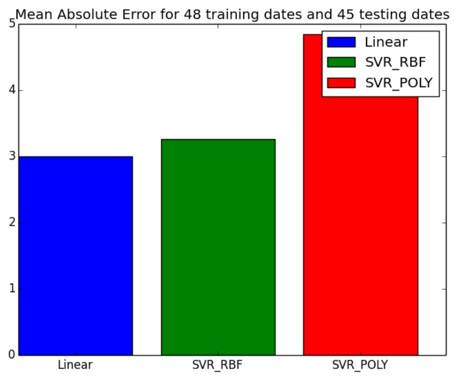

Figure 11: Mean absolute error for Figure 10.

When comparing the regression, mean absolute percentage error was used because of the high variation of stock prices. This ensures that the results are not biased against stocks with higher prices, since the error is calculated as the percentage of that stock price that the prediction was off by.

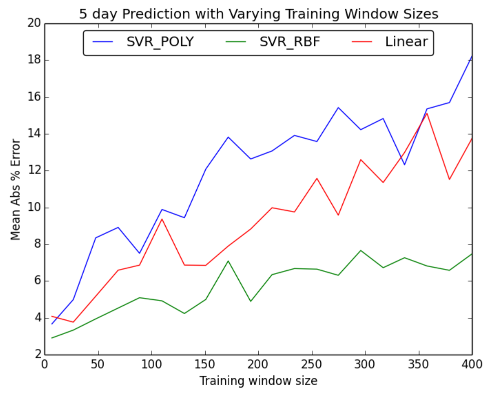

Figure 12: Mean absolute percentage error of a 5 day stock price prediction for the three regression methods.

Support Vector Regression with the RBF kernel performed the best overall in the trials that we have run, with the Linear and SVR with the polynomial kernel varying more significantly. SVR with the RBF kernel had consistent short term results of $\approx$ 5% mean absolute percentage error (MAPE) while SVR with the polynomial kernel and linear regression had $\approx$ 10% MAPE on average.

It is interesting to note that for a majority of the trials run, we have found that smaller training window sizes almost always had better results, with some of the best accuracies at around 50 prior dates.

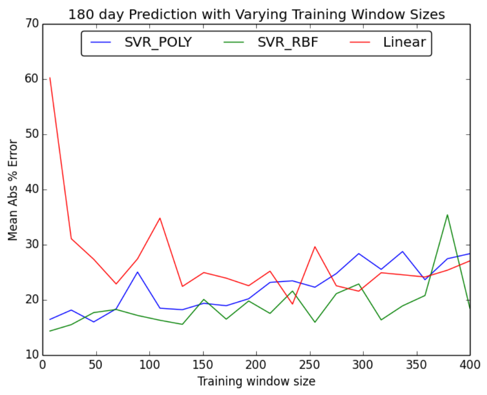

Figure 13: Mean absolute percentage error of a 180 day stock price prediction for the three regression methods.

The results for 180 day price predictions were chosen to provide insight onto the performance of these algorithms for a longer period of time. Overall, SVR with the RBF kernel performed the best, but it is interesting to note that SVR with the polynomial performed better in comparison with the rest of the algorithms on these longer time frames.

Linear regression performed very poorly when its window size was small for long-term price prediction, but actually ended up outperforming the other algorithms when their window sizes tended to be on the order of years (365-400 days).

VI. Conclusion and Future Work

Several issues have been addressed throughout the paper, including the process of feature selection, normalization, and training set window size. These issues warrant expansive research on their own regard, but this research has done its best to mitigate these factors by using features collected by a top finance research institute [2] and algorithms that are provided in an actively developed and maintained machine learning library [1].

The subset of CRSP data that was selected was not substantial enough on a global GDP level to create a decision tree of buy/sell decisions based off the industry and sector data as in [3], but future work could include data from other sources corroborated with the CRSP data to perform regression and/or binary buy/sell classification for whole industries or sectors at once, if so desired.

It is interesting how linear regression can perform better than polynomial methods at certain intervals due to the reduced chance of linear regression overfitting the training data. In some cases, we found that for long term projected market fluctuations linear regression performed well. This case was especially true when a polynomial method would overfit the training data and have increased performance at the beginning of the testing data, but at the cost of very inaccurate results in the later prediction dates. Conversely, linear regression was less accurate at the beginning of the prediction, but wouldn't perform as badly as a polynomial regression method that diverged.

An opportunity for future research also emerges from applying additional linear and polynomial regression methods to this problem. This software suite was architected in such a way that the regression is only at one critical point such that different regression algorithms can modularly be swapped in and out as needed. This is also true of the parameters that can be used for these algorithms, more research could be done in this area since many of the parameters are embarrassingly domain specific for regression models, which may result in drastic performance increases.

Higher order polynomial regression methods were more likely to overfit the training data than linear regression, and it is quite often the case that it is situational of when the right order of polynomial best fits the training data without overfitting. Often, it is only apparent after we know the ground truth of the prices, therefore it is difficult to recommend to use most of these models for any high stakes financial planning, be it personal finance or otherwise.

Acknowledgment

The CRSP dataset was generously provided by the Department Chair of Finance at the University of New Mexico, Dr. Leslie Boni.

I would also like to thank Dr. Trilce Estrada for providing guidance on the project and helping to motivate the various regression techniques used above.

References

[1] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[2] Center for Research in Security Prices The University of Chicago. Us stock databases. web, 2014.

[3] C. Tsai, and S. Wang 2009. Stock Price Forecasting by Hybrid Machine Learning Techniques. *Proceedings of the International MultiConference of Engineers and Computer Scientists*, 20-26, 2009.

[4] Hurwitz, E, and T Marwala. 2009. "Common Mistakes when Applying Computational Intelligence and Machine Learning to Stock Market modelling." University of Johannesburg Press.

[5] Lee, Jae Won, Jonghun Park, Jangmin O, and Jongwoo Lee. 2007. "A Multiagent Approach to Q-Learning for Daily Stock Trading." IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS 864-877.

[6] Wang, Yanshan, and In-Chan Choi. 2013. "Market Index and Stock Price Direction Prediction using Machine Learning Techniques: An empirical study on the KOSPI and HSI." *ScienceDirect* (ScienceDirect) 1-13.

[7] Yan, Robert, and Charles Ling. 2007. "Machine Learning for Stock Selection." *Industrial and Government Track Short Paper Collection* 1038-1042.