Predicting Stock Prices and Analyst Recommendations

Saumitra Thakur
SUID: 05921351
Stanford University
sthakur2@stanford.edu

Theo Vadpey
SUID: 06005208
Stanford University
tvadpey@stanford.edu

Sandeep Ayyar
SUID: 05795996
Stanford University
ayyars@stanford.edu

I. Introduction:

Since the mid-2000s, virtually all financial trades are executed via computers. Much of that trading occurs algorithmically, with computers executing purchases and sales in response to a market made heavily of other algorithmic traders. [1] About 66 percent of all equity transactions in the United States now are done via high-frequency (algorithmic) trading. [2] As a consequence, market behavior has fundamentally changed. In several occurrences in the past decade, markets experienced "flash crashes" where one algorithm making a big sale triggered a waterfall of other algorithms to respond similarly and the market tumbled to a fraction of its value within seconds. [3]

The rise in computerized trading has been accompanied with increasing efforts to use machine learning to predict future market behavior. Previous research has focused on forecasting movements in price (either as a continuous price or in discrete intervals) or forecasting decisions to buy or sell. These studies have met with mixed results. For both practical and theoretical reasons, the price of traded equities remains difficult to predict, with the task described as "difficult if not impossible" by some. This is due both to the large number of possible features that could be used to train an algorithm as well as the lack of consensus on what, theoretically, ought to underpin the valuation of a security. [4]

Researchers have tried a range of techniques, including supervised learning techniques, artificial neural nets, back-propagation networks, hybrid Kohonen self organizing maps, and other methods. [5] Several prior projects in this course have examined financial trading as well.

Although computerized trading has left its mark on equities markets, much of equity transaction today still occurs based on value judgments by humans. These judgments, in turn, often reflect the sentiment of professional equity analysts employed by financial institutions like Goldman Sachs. Indeed, many computer scientists have shifted their focus away from forecasting prices or other stock technical properties and instead focusing on forecasting perceptions.[6]

A large number of traders conduct market research through Bloomberg terminals. Bloomberg is the market leader in distributing financial data.[7] Bloomberg's market power is so substantial that the UK government had to postpone a major sovereign debt buyback when the Bloomberg network went offline briefly in 2015.[8] For public equities, Bloomberg offers a consensus analyst recommendation from 1-5 (strong sell to strong buy) that reflect the aggregate opinion of equity analysts. Prior research on how these recommendations impact the market suggests that they may actually move markets in the opposite direction (fueled, perhaps, by a perception that others with the same information will move in the direction of the recommendation). Regardless of direction, researchers agree that such recommendations have a major impact on market perceptions and prices. [9]

We take a novel approach to applying supervised learning to financial modeling. In addition to forecasting the price of an equity in the future, which we treat as a regression problem, we also forecast Bloomberg consensus analyst recommendations, as a classification problem. (Analyst recommendation categories between 1-5). As discussed previously, these consensus numbers have tremendous power to shape market perceptions and are associated with detectable movements in the value of the stock. Prior studies have forecasted price, but little to nothing has been published on forecasting analyst recommendations.

II. DataSets

We collected over 130 data sets spanning about 8 years (quarterly) for each company in the Standard and Poor 500 (S&P 500) and the Russell 2000. The S&P 500 consists of the 500 largest companies by market capitalization, while the Russell 2000 consists of 2000 small companies. We were curious how our methods would perform between the two datasets. However we ended up abandoning the Russell 2000 due to the large amount of missing data relative to the S&P companies.

III. Methods:

A. Data Processing

The data took the form $\{X^i, y^i\}$, where X^i is a matrix of features in $\mathbf{R}^{32\times 133}$ for company i. The rows of X^i represented time, the columns represented the features. The label for company i was given by y^i in $\{1,2,3,4,5\}^{32}$. Rows corresponded to time in $\{2006Q2, ..., 2015Q4\}$.

We thinned the original dataset, including only features and companies with an adequate number of observations. We prioritized keeping companies in the dataset over features. From an initial 133 features, and 505 companies, we narrowed the dataset to 499 companies and 100 predictors, which included balance sheet items, like cash stock and goodwill, income statement items like revenue, and several macroeconomic indicators, e.g., GDP growth, 10-year treasury rate.

We added percentage change over the previous period as a feature for each existing feature. We anticipated that in some cases the percentage change from the previous quarter would be more significant than the quanitity in a single quarter.

We made an important modeling decision to treat each feature as a random variable, and assumed that this was iid across companies. Using this assumption, we restructured the data such that instead of a sequence of matrices, $[X^1, y^1], [X^2, y^2], ..., [X^{499}, y^{499}]$, we operated on a large matrix:

$$\begin{bmatrix} X^1 & y^1 \\ X^2 & y^2 \\ \vdots & \vdots \\ X^{499} & y^{499} \end{bmatrix}$$

In order to attempt to preserve the time information, before restructuring in this way we applied a simple autoregressive model to each data series (on each company) to find the number of significant lags. Knowing that linearly dependent features would be removed by our feature selection procedure later on, we chose the maximum number of lags that a feature demonstrated for any company, and included those lags as features.

Missing values were a significant problem in our dataset. We chose to approach the problem with 0-1 dummy variables. We replaced missing data with an arbitrary value, 0, and included a dummy feature where each element corresponding to a missing value took a value of 1, and was 0 otherwise. We found that this method conformed with our lack of an opinion with respect to missing values.

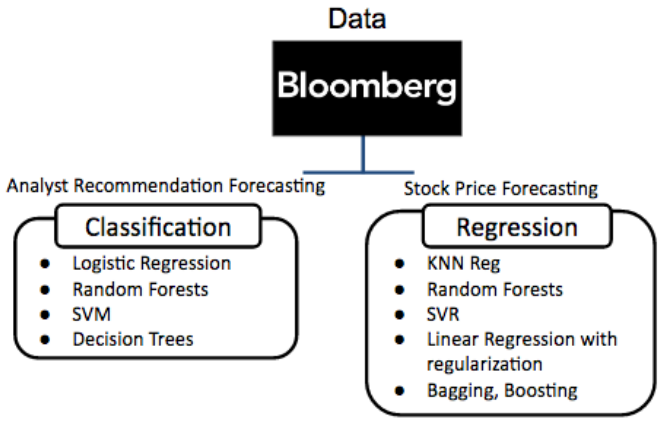


Fig. 1: Overall Pipeline for Predicting Components of Portfolios

B. Model training and Evaluation

We randomly divided our data samples into training (80%) and test sets (20%). We trained several supervised learning models on the training data using 10 fold cross-validation to evaluate model performance. We also performed parameter tuning using 10 fold-CV to select the optimum performing metrics for each model algorithm. We then fit our model on the entire training set and predicted responses on the test set. We divide our learning tasks into two parts as seen in Fig 1: Classification for predicting the analysts stock ratings and Regression for predicting Stock prices. Accordingly, we use different metrics for evaluating the performance of our models. For the classification problem, we use metrics such as "Accuracy", "Specificity", "Sensitivity", "Positive Predictive Value (PPV)" and "Negative Predictive value (NPV)".

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

$$Sensitivity = \frac{TP}{TP + FN} \tag{2}$$

$$Specificity = \frac{TN}{FP + TN} \tag{3}$$

$$PPV = \frac{TP}{TP + FP} \tag{4}$$

$$NPV = \frac{TN}{TN + FN} \tag{5}$$

where,
TP = true positive rate, TN = true negative rate
FP = false positive rate, FN = false negative rate
For regression, we use root mean squared error (RMSE) to evaluate models

$$RMSE = \sqrt{\frac{1}{n} \times \sum_{i=1}^{n} \times \left(\frac{y_i - \hat{y}_i}{y_i}\right)}$$

We briefly describe some of the methods used:

1) Support Vector Machines: We used support vector machines for classification and support vector regression, an extension of SVM, where the response is continuous instead of binary, for regression. SVM's are quite effective in high-dimensional spaces and fairly versatile in terms of choosing different kernels e.g linear, polynomial, radial basis functions. We select the optimum models by tuning parameters such as cost, gamma and kernels.

For classification, since we have $K > 2$ classes, we used the one versus all classification approach. Here we fit K SVMs, and each time we compare one of the K-classes to the remaining $K-1$ classes. For example if β_{0k}, β_{1k}, ..., β_{pk} are the parameters that result from fitting an SVM comparing the kth class (coded as $+1$)to the others (coded as -1). If x^* is any test observation, then SVM assigns this observation to the class for which β_{0k} + $\beta_{1k}x_1^*$ + $\beta_{2k}x_2^*$ + ... + $\beta_{pk}x_p^*$ is

the highest, as this corresponds to a higher likelihood that the test observation belongs to the kth class rather than any of the other classes.

In case of support vector regression, the method seeks coefficients that minimize an epsilon loss function, where only residuals (y_i - β_0 - $\beta_1 x_{i1}$ - .. - $\beta_p x_{ip}$) larger in absolute value than some positive constant contribute to the loss function.

2) K-nearest Neighbors: K-nearest neighbor is a non-parametric method where given a positive integer K and a test observation x_0, KNN first identifies K points in the training data that are closest to x_0, represented as N_0. KNN then estimates the conditional probability of class j as the fraction of the number of points in N_0 whose response values equal j. Then KNN classifies the test observation x_0 to the class with the highest probability by applying Bayes rule. KNN is useful as the cost of the learning process is not large and no assumptions need to be made about the characteristics of the concepts. However, in our case KNN is computationally expensive since number of features is large. We used 10-fold CV to select number of neighbors. In KNN regression, instead of combining discrete predictions of K-neighbors, we combine continuous predictions. These predictions are combined by averaging.

3) Lasso, Ridge based methods: In Ridge method, a linear model is fit by penalizing the coefficients using L2 norm by virtue of a tuning parameter λ. As λ approaches a large value, the 2nd term in the Eqn 6 called the shrinkage penalty, grows and the coefficient estimates approach 0. This forces some of the coefficients towards zero. The ridge method will therefore include all the p predictors in the model which is a disadvantage if we have a large number of predictors. In Lasso method, the coefficients are penalized using L1 norm by λ (Eqn 7. As λ becomes large enough, some of the coefficients will actually shrink to zero. Hence, Lasso performs variable selection and the models are easier to interpret as it will give rise to a sparse model. We select the value of λ using 10 fold CV.

$$\sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij})^2 + \lambda \sum_{j=1}^{p}\beta_j^2 \tag{6}$$

$$\sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij})^2 + \lambda \sum_{j=1}^{p}|\beta_j| \tag{7}$$

4) Tree-Methods: Tree-based methods involve separating the features into a number of regions. We then estimate the mean of training samples in the region for which a data point which we want to predict belongs to. Basic trees are simplistic and do not generalize very well. Hence we evaluate approaches such as boosting, bagging and random forests. Bagging takes a subset of samples of the training observations and fits several trees. The predicted values on are then averaged which helps to reduce the high variance in a basic tree. Random forests gives a slight improvement over bagging by decorrelating the trees by randomly sampling m predictors from the full set of p predictors as split candidates every time a split is considered. Boosting involves sequentially growing trees where each tree is grown by using information from previous trees. We optimize parameters of these tree based methods by estimating the out-of-bag error estimates.

TABLE I: Parameter Tuning for Selected Models

Classification Model	Tuning Parameters
Lasso, Ridge Based Methods	regularizing/penalizing term
Support Vector Machine	kernel function, cost, gamma
Decision Trees	number of trees
Random Forests	number of estimators, number of features
K-Nearest Neighbors	number of neighbors

IV. Results:

A. Classification

As we are dealing with $K > 2$ classes (multinomial), we used grouped- lasso penalty on all K-coefficients of particular variables. This shrinks them to zero or non-zero together. Fig 2 shows the cross-validation curve along different values of λ. The λ which gives the minimum CV deviance is chosen and corresponding coefficients at that λ are selected by the model.

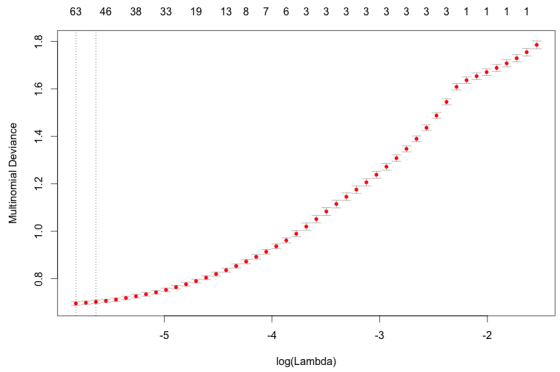


Fig. 2: 10-fold CV showing Multinomial Deviance of Lasso Fit

The nature of the label we tried to predict was such that in most cases, the previous label (included as a feature) was the same as the current label. Consequently, for all classification results, we felt an appropriate **naive benchmark would be the percent of labels perfectly predicted by the previous period's label**. **This was 82.4% in our dataset**. As seen in Table II and Fig 3, most of our results were close to this in terms of accuracy, outperforming the naive estimate slightly (by around 4%, with the exception of SVM). Overall, random forests was the best performing classifier. While most classifiers had high accuracy, specificity, PPV and NPV, they suffered from low sensitivity.

B. Regression

For regression, we started out by evaluating ordinary least squares model for predicting stock prices which had a test RMSE of 41.1. To improve performance, we applied shrinkage

TABLE II: Performance Comparison of Different Classifiers for Analyst Recommendations (naive benchmark accuracy 82.4%)

Model	Accuracy	Specificity	Sensitivity	PPV	NPV
Lasso + Logistic Reg.	**0.867**	**0.920**	0.611	0.760	0.935
SVM	0.817	0.887	0.587	0.760	0.916
Random Forests	**0.867**	**0.920**	**0.625**	**0.891**	**0.939**
Decision Trees	0.864	**0.920**	0.617	0.621	0.929

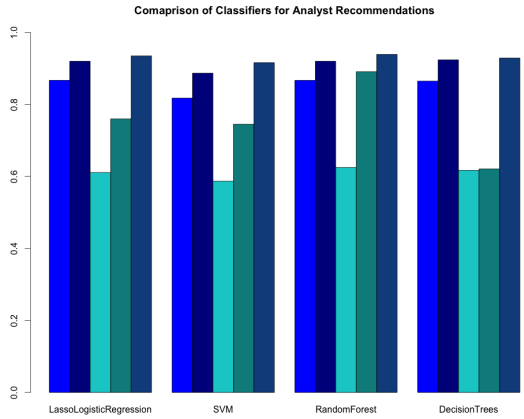
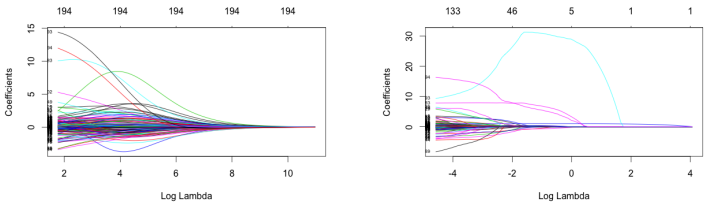


Fig. 3: Performance Comparison of Different Classifiers for Analyst Recommendations



(a) (b)

Fig. 4: Plots of (a) Ridge Coefficients as a function of λ (d) Lasso coefficients as function of λ

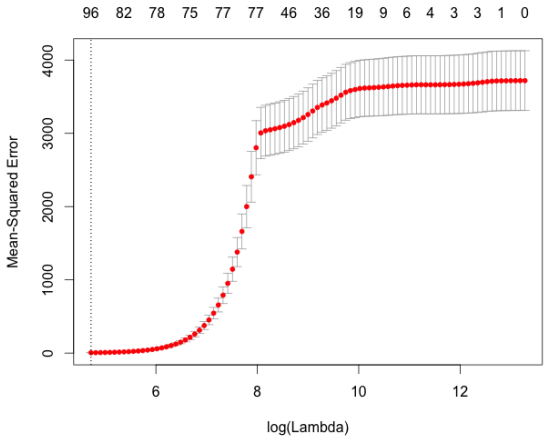


Fig. 5: 10 fold CV showing Mean Squared Error of Lasso Fit

methods such as Lasso and Ridge regression. Fig 4 shows the result of coefficients being shrunk towards zero 4a in case of ridge regression and completely to zero 4b in case of lasso. Fig 5 shows the reduction in training mean squared error(MSE) as a function of log λ values in Lasso where MSE increases with increase in the penalty term a dictated by the regularization parameter λ.

As we had a large number of predictors, Lasso works better than ridge regression as it performs feature selection and selects only non-zero coefficients in the model for training. Accordingly, as we see in table III, applying Lasso results in a test RMSE of 33.1 while ridge regression results in test RMSE of 38.1. Other methods such as SVM, and tree based methods perform slightly better than Lasso regression, with Boosting giving us the lowest test RMSE of 27.0. Overall, while we managed to significantly reduce the test error from simple linear regression, there is still room for improvement.

TABLE III: Results for Regression: Stock Price Prediction

Regression Model	Test Root Mean Squared Error
Linear Regression	41.1
Linear Regression with Ridge	38.1
Linear Regression with Lasso	33.1
Support Vector Regression	29.7
K-Nearest Neighbors (5)	39.3
Bagging	28.5
Random Forests	27.1
Boosting	**27.0**

V. Discussion:

Overall, with respect to classification, most of our models mildly improved over the naive benchmark, achieving accuracy of around 86% compared with 82% (the naive benchmark). It would be interesting to see if the information captured by this improvement would be valuable for security selection. A simulation and related security selection procedure could give answers to this.

With respect to regression, we achieved great improvements over simple linear regression by applying different approaches. We managed to get RMSE from 41.1 to 27.0 in the best model (via boosting).

Some ways in which we might improve our error or extend analysis:
-We might include polynomial terms, nonlinear transformations, or interaction terms in the models.
-It might be more interesting to predict analyst labels further in the future because they are less related to the previous label.
-It might be interesting to model the analyst recommendations as Markov processes, and calculate metrics such as average holding time.

NOTES

[1]McGowan, M. J. (2010). Rise of Computerized High Frequency Trading: Use and Controversy, The. Duke L. & Tech. Rev., i.

[2]Patterson, Scott. (2010). "Fast traders face off with big investors over 'gaming.'" 29 June 2010. The Wall Street Journal.

[3]Kirilenko, A. A., Kyle, A. S., Samadi, M., & Tuzun, T. (2015). The flash crash: The impact of high frequency trading on an electronic market. Available at SSRN 1686004.

[4]Brabazon, Tony. "A Connectivist Approach to Index Modelling in Financial Markets." Aug.-Sep (2000).; Afolabi, M. O., & Olude, O. (2007, January). Predicting stock prices using a hybrid Kohonen self organizing map (SOM). In System Sciences, 2007. HICSS 2007. 40th Annual Hawaii International Conference on (pp. 48-48). IEEE. Chicago

[5]Afolabi, M. O., & Olude, O. (2007, January). Predicting stock prices using a hybrid Kohonen self organizing map (SOM). In System Sciences, 2007. HICSS 2007. 40th Annual Hawaii International Conference on (pp. 48-48). IEEE.; Zekic, M. (1998, September). Neural network applications in stock market predictions-a methodology analysis. In proceedings of the 9th International Conference on Information and Intelligent Systems (Vol. 98, pp. 255-263).

[6]Kaplan, C. A. (2001). Collective intelligence: A new approach to stock price forecasting. In Systems, Man, and Cybernetics, 2001 IEEE International Conference on (Vol. 5, pp. 2893-2898). IEEE.

[7](2015). "Bloomberg increases market share lead over Thomson Reuters." 24 March 2015. The Baron.

[8]Cox, Josie. (2015). "Bloomberg Terminals Go Down Globally." 17 April 2015. The Wall Street Journal.

[9]http://wealthmanagement.com/equities/do-analyst-investment-recommendations-really-drive-stock-performance

TABLE IV: Sample data dictionary

	Code	Definition
1	EE201	Analyst recommendation (1-5)
2	PX_LAST	Last Price
3	BS035	(BS) Total assets
4	BS081	(BS) Shares outstanding
5	BS010	(BS) Cash and cash equivalents
6	BS047	(BS) Short-term debt
7	BS051	(BS) Long-term debt
8	BS050	(BS) Total current liabilities
9	BS061	(BS) Preferred equity
10	BS032	(BS) Net PP&E
11	BS015	(BS) Total current assets
12	BS013	(BS) Inventories
13	BS012	(BS) Accounts and notes receivable
14	BS036	(BS) Accounts payable
15	BS011	(BS) Short-term investments
16	BS029	(BS) Long-term investments and receivables
17	BS161	(BS) Goodwill
18	BS138	(BS) Total intangible assets
19	BS030	(BS) Gross PP&E
20	BS065	(BS) Retained earnings and other equities
21	BS052	(BS) Other long-term
22	BS014	(BS) Other current assets
23	BS033	(BS) Other long-term assets liabilities
24	BS048	(BS) Other short-term liabilities
25	BS064	(BS) Share capital & APAC
26	BS091	(BS) Number of treasury shares
27	BS080	(BS) Pension and post-retirement obligations
28	BS031	(BS) Accumulated depreciation
29	BS180	(BS) Rental expense
30	BS187	(BS) Future minimum operating lease obligations
31	BS089	(BS) Intangible and other assets
32	BS210	(BS) Total capital leases