Finding Undervalued Stocks with Machine Learning

Ramneet Singh Rekhi
rr1602@nyu.edu

Tucker L. Ward
tlward@stanford.edu

Huan Wei
huanw2@stanford.edu
Department of Statistics
Stanford University, CA

Acknowledgement
Michael Vincent
Downs

1. Introduction

1.1 Past Work

There has been a great amount of studies with Machine Learning on financial market. However, a majority of the studies focus on short-term market performances and high frequency trading strategies. Nevertheless, there are studies that focus on long term strategies. One study shows that nonlinear support vector machines can systematically identify stocks with high and low future returns[1]. Other studies also suggest that SVM would a preferable methodology. This will be also confirmed through our analysis.

1.2 Predicting

We are predicting if a stock, given certain features, is significantly undervalued. This will be a classification problem. We define a significantly undervalued stock to have a 50% increase annually. The annual S&P 500 return is 10% on average. [2]

1.2.1 Labeling

We decided to employ three different labeling techniques for our study, We label the stocks based on a (time, return_threshold) pairing. (1, 50%): Significantly undervalued stock, which is a stock increases 50% in a year. However, we are also experimenting on having various input for return_threshold. For now, we have 10%, 20%, 30%, 40%, 50%. (1,-1) or (0,-1) : Not a significantly undervalued stock. In logistic regression, this will be labeled as (1,0) and in SVM, this will become (1,-1). However, as Huerta and Corbacho point out, this approach does not control for risk [1]. Therefore, we borrowed two labeling schemes they used in their study. First we calculated sharpe ratios for each stock. A sharpe ratio is defined as:

$$S = \frac{E[R - R_b]}{\sqrt{\text{var}[R - R_b]}}.$$ [1]

This adjusts the return of a given stock by a benchmark risk-free rate, b, for which we used the US treasury constant maturity three month series available from the federal reserve, and divides the asset by its standard deviation. We then rank ordered stocks at time t, based on their sharpe ratios, and gave the top 10% of this list a 1 label, and everything else a (-1,0) label. The final labeling technique we employed, was from the Capital Asset Pricing Model[2]. This theory proposes the return of a stock can be decomposed, into market return, measured by its covariance with a market index and an idiosyncratic component represented by alpa, whose expected value is zero. Hence the return of a stock can be viewed as:

$$r_{a,t} = \alpha + \beta r_{b,t} + \varepsilon_t$$[3],

Therefore we regressed each stock onto the sp500 index, at a given time interval, and ranked ordered them by their alphas, and mapped the top 10% to a 1 ranking and all else to a (-1,0) ranking.

2. Data

Our primary data set contains 1,500 stocks with approximately 150 features reported quarterly from 1999 to 2012, for a total of nearly 13 million data points (the data for some stocks does not cover the entire time period). The 150 features encompass everything from earnings to capital expenditures to taxes. We primarily collected data from a Morningstar database and Wharton's WRDS databases. After downloading the raw data, we organized and processed it into a common format. While

1 http://en.wikipedia.org/wiki/Sharpe_ratio
2 see http://www.sciencedirect.com/science/article/pii/0304405X93900235 for more details
3 from http://en.wikipedia.org/wiki/Beta_%28finance%29

conceptually simple, completing just this preprocessing required nearly 2,000 new lines of Python code and a significant time investment.

Selecting which stocks to analyze was a difficult and deliberate choice. On one hand, well-known and widely-followed stocks (eg, Apple, Exxon) will tend to be more stable and price may more accurately follow commonly-accepted valuation metrics. On the other hand, such stocks present fewer opportunities for significant, market-outperforming growth or decline precisely because they are so well-followed and tend to be large, well-established firms. We thus compromised and started our analysis with the S&P 500 stocks, which encompass both big names like Apple and relatively unknown names like AbbVie. After our initial analysis, we decided to expand our research to mid-cap stocks in the S&P 1000 range to provide yet more examples of relatively unknown companies with higher growth potential.

We employed two techniques for feature selection at the data collection stage. First, we research common market indicators and ranked features according to how likely we thought they were to provide useful information. This ranking helped identify several important features such as momentum and short interest. Second, we used a "shotgun" approach to test all of the features available to us and see which were most useful, after eliminating features that were a function of stock price (eg, market cap).

We tracked two dependent variables: market cap and maximum stock price attained within a given quarter. Both dependent variables are necessary because of a fundamental basis of stock price: company valuation divided by outstanding shares. Tracking market cap gave us an indication of how a company's estimated fundamental worth changed according to the features (the "company valuation") while maximum stock price attained gave us a more direct measure of stock price. A savvy investor carefully tracks both valuation metrics, and anomalies in either metric can indicate strong buy or sell signals.
Other relevant sections will speak more specifically about stocks, features, and independent variables, but significant thought was given to careful data selection long before we ran any machine learning algorithm.

2.1 Data Quality
The nature of the stock market data, which consists of various market indicators and accounting measures, is less structured and needs to be regularized in order to work with various models. So far we have used chi-square discretization to process data. Fortunately, accurate stock and company data is readily available through Morningstar and Wharton, and in general we need not worry about data accuracy. Finding relevant data was also never a problem, although as stated it required significant preprocessing.

2.2 Features
We employed two approaches for feature selection in our algorithms. First, our background research indicated that two features specifically--momentum and short interest--were most likely to be useful features. Momentum measures how much a stock has been up or down in the recent past and has been shown to be a reasonable predictor of immediate future success, although we target longer-term returns. Short interest, or the percentage of shares that investors are betting will decrease in value, reflects a sense of "sentiment" about whether traders think the stock will increase or decrease in the coming one to two months. We paid special attention to those features and several others that showed promise.

Second, we also evaluated each of the available 150 features to see if we could find useful information among the less-obvious features. Those features encompass everything from earnings to capital expenditures to taxes. We also implemented several feature selection methods, namely K means, Principle Component Analysis, Singular Value Decomposition, Entropy, Divergence and N Factors, to help resolve the more promising features.

3. Models and Results
We tested on the following models: Decision Tree, KNN, Support Vector Machine, Random Forest, AdaBoost, Naive Bayes, Logistic Regression, Linear Discriminant Analysis. We found that SVM with backward search (whether the search is being performed by Naive Bayes or SVM) and AdaBoost with is a powerful combination. We implemented a scoring system for all the models and feature selections methods:

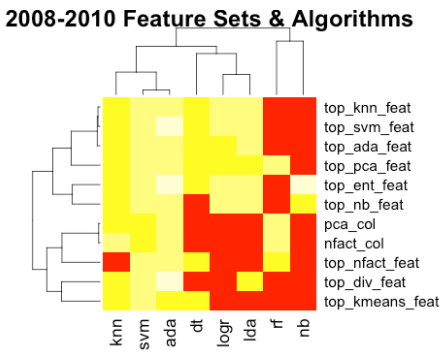
$$Score = \frac{1 \times precision + 100 \times postprod}{1000 \times misclass}$$

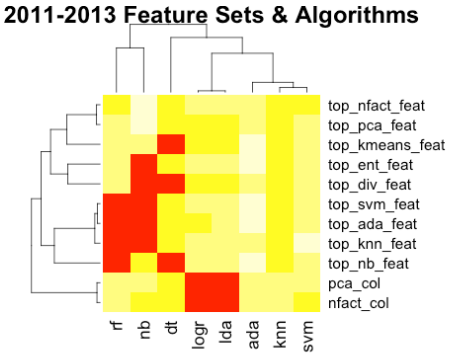
Interpretation of the formula: If you had 100% precision and 100% productivity and 0.0001% misclass (i.e if you predicted 100% of possible y's with 100% precision and 100% accuracy, you would score a 1). That means, numbers around 0.502 are good. Numbers around 0.54 are more likely to be resulted from over-fitting and will likely decrease with cross validation.

Here's an abstract of the scores across models and features:

	Features	DT	KNN	SVM	RF	ADA	NB	LogR	IDA
score	top_svm_feat	0.55	0.58	0.63	0.00	0.64	0.00	0.59	0.60
prod	top_svm_feat	0.03	0.05	0.04	0.00	0.01	0.79	0.07	0.07
precis	top_svm_feat	0.18	0.29	0.56	0.67	0.67	0.13	0.33	0.34
misclass	top_svm_feat	0.14	0.14	0.11	0.12	0.11	0.70	0.14	0.14

The following heatmaps generated from the scores shows how well each algorithm performs on different feature sets. The lighter the color, the better the scores. We can see that AdaBoost, KNN and SVM perform consistently well across various feature sets on two different timeframes.





Please see attachment for a detailed list of results.

4. Discussions and Analysis

4.1 Model and Feature Selection Analysis

After evaluating the models and feature selection methods with our unique scoring system, we found the following conclusions:

- Backward search appears to work better than other supervised and unsupervised feature selection approaches.
- Un-optimized logistic regression can still perform if focused on the right features.
- Results suggest there is enough variability of errors that mixing data sets and algorithms in an ensemble of votes will likely outperform any individual algorithm.

4.2 Discussion on Model Productivity

While training on different models and feature selection process, we want the model to be able to encounter a variety of economic circumstances and still predict. Therefore, we developed a number of data sets and are finding that the algorithms behave differently to each one. For instance, SVM and AdaBoost each has its favorite datasets which it will be able to predict consistently. That leads us to believe that one way to ensure generalization to a variety of economic environments will be to use a range of datasets and algorithm combinations to develop ensemble predictions that will be more accurate than any individual one.

We've also automated both feature selection and algorithm parameterization. Regarding the latter, for our top performing algorithms, we run iteratively over each data set changing parameters using a "tournament" model. So, for example, in SVM algorithm, it has a "fast" mode which uses the best all around parameter set, but it also has an "accurate" mode where it starts by fitting based on model type, then moves to kernel type, then cost, then gamma, then nu. This results in algorithms that could be specifically dialed into each data set to maximize the individual result.

4.3 Discussion on Memory on Prior Performance

Noting that our data set was in a panel format, because we have observations for multiple individuals, over multiple time periods. We decided to exploit this property by incorporating time specific and individual specific effects into our model. To accomplish this we created an augmented feature vector as outlined by Dundar, Krishnapuram, et al., this entailed creating dummy variables for each quarter and each stock. This helped control the bias in our model parameters and also had the added effect of giving the model "memory" for each stock it looked at, over a given time period. When conducting cross validation, on our test set we zeroed out all of the time dummy variables, before they were fed into the model, because we were afraid this

could introduce out of sample information, however, we kept the stock dummies, exploiting any conclusions the model might have come to apriori about a given stock.

4.4 Real World Stress Test on SVM with Linear Kernel

We found that we are outperforming the market at 50% of the time on a quarterly basis, with our culmultive portfolio returning over 150% in the time period from 6/30/2009-9/30/2013, while the benchmark S&P 500 is up about 110% . The best results actually came from optimizing our model to the alpha labelings described above on a Support Vector Machine algorithm with a linear kernel, which was one of our highest scoring algorithms based on our earlier analysis:

date	market_shape	portfolio_sharpe	**outperformance**	alpha	tstat	beta
6/30/2009	22.80%	29.91%	TRUE	0.00	1.55	0.77
9/30/2009	11.40%	-0.62%	FALSE	0.00	-0.75	0.67
12/30/2009	8.84%	28.25%	TRUE	0.00	2.07	0.30
3/30/2010	-8.00%	-3.51%	TRUE	0.00	0.13	0.32
6/30/2010	16.11%	7.84%	FALSE	0.00	-0.09	0.35
9/30/2010	23.06%	9.09%	FALSE	0.00	0.41	0.20
12/30/2010	11.96%	4.60%	FALSE	0.00	0.31	0.06
3/30/2011	-3.59%	8.09%	TRUE	0.00	0.64	0.04
6/30/2011	-9.14%	23.84%	TRUE	0.00	1.72	-0.18
9/30/2011	11.87%	-11.90%	FALSE	0.00	-0.70	-0.19
12/30/2011	31.63%	27.79%	FALSE	0.00	2.23	-0.10
3/30/2012	-8.26%	-5.22%	TRUE	0.00	-0.31	0.18
6/30/2012	13.34%	2.14%	FALSE	0.00	0.06	0.12
9/30/2012	-4.67%	-11.94%	FALSE	0.00	-0.89	0.08
12/30/2012	25.67%	1.52%	FALSE	0.00	-0.04	0.10
3/30/2013	6.33%	26.23%	TRUE	0.00	2.08	-0.03
6/30/2013	14.69%	21.78%	TRUE	0.00	1.74	-0.05
9/30/2013	23.62%	39.37%	TRUE	0.00	2.55	0.30

To construct this stress test we used a moving 10 quarter window, where we estimated the model. Then tested this model on the following quarter. To map to actual stock predictions we employed a similar technique as Huerta and Corbacho. We rank ordered the stocks by the output of the decision function, and constructed an equal weight index of long the top 10% highest ranked stocks, and short the lowest 10% stocks.

5. Future Discussion

The model needs to be further improved. It optimizes against a single set of y's based on 10 prior quarters of x's. It's using a set of features that worked during a particular window based on value ranges that may have only occurred in that window. The ultimate objective is to finalize to a model that works across time periods. As such, we suggested the following for future implementations.

5.1 Technical Functionality
We discovered that "for" in controller loop can't accept character variable names and it could be fixed with sapply or lapply. We also like to modify the codes to ensure our models and results are replicable across different datasets, different time frames or when implemented by different end users.

5.2 Data
Firstly, if we can have new dataset that take in new x and y-variables from Tyler scaling models. We can then build new x variables from the value managers (original spreadsheet), for example earnings acceleration.
On feature selection, we could implement reverse search on Naive Bayes, KNN to validate their settings, feature sets. We would also like to refine clustering to improve results on entropy and divergence, as well using other clustering algorithms including latent class analysis to evaluate features. Investigate missed predictions (false pos then false negative) to uncover data and algorithm inadequacies. Smear y influence over key variables.

5.3 Algorithms
We would like to modify our current models. Each algorithm can be optimized further. We need to do k-fold cross validation to find best algorithm parameters. In addition we should write an algorithm to run the optimization of each algorithm varying parameters. We will also move to multivariate predictors consistent w/ new multi-y dataset.
For learning algorithm, we need to build feature memory and stock memory (c/b beta) into algorithm. Different algorithms have their own strength. Some algorithms, like logistic, let you set priors. Others, like AdaBoost, lets you weight features and/or observations. It takes more time to decide between using current algorithms or modifying existing algorithms.
We will also implement ensemble voting weighing votes for each record, for instance, having the following formula within an algorithm based on settings and data set or assess across algorithms based on strengths and weaknesses.

(SVM vote * SVM weight) + (Naive Bayes vote * Naive Bayes weight) + ...

We are also interested in incorporating new algorithms, for instance Localized logistic regression, Neural networks, LDA and additional clustering, including latent class or latent drichlet.
We will also like to diversify our portfolio strategies. We will consider both long and short portfolios strategies. We would also to incorporate tools to separate models in order to find worst performers and emulate profits of pair trades.

6. Attachments

Model Evaluation Score Results
https://drive.google.com/a/stanford.edu/file/d/0B4AVk02HpEPtZWw0NDBfa1hRb1E/view?usp=sharing

7. Reference
[1] Huerta, R., Corbacho, F., & Elkan, C. (2013, February 1). Nonlinear support vector machines can systematically identify stocks with high and low future returns. Retrieved December 7, 2014, from http://biocircuits.ucsd.edu/huerta/AF_Huerta.pdf

[2]Damodaran, A. (2014, January 5). Retrieved December 5, 2014, from http://pages.stern.nyu.edu/~adamodar/New_Home_Page/datafile/histretSP.html

[3] A Stock Selection Model Based on Fundamental and Technical Analysis Variables by Using Artificial Neural Networks and Support Vector Machines. (2012). *Review of Economics & Finance*. Retrieved December 10, 2014, from http://www.bapress.ca/ref/ref-2012-3/A Stock Selection Model Based on Fundamental and Technical Analysis Variables by Using Artificial Neural Networks and Support Vector Machines.pdf
[4] Varian, H. (2014, January 29). Machine Learning and Econometrics. Retrieved December 5, 2014, from http://web.stanford.edu/class/ee380/Abstracts/140129-slides-Machine-Learning-and-Econometrics.pdf

[5] Useful Lists. (n.d.). Retrieved December 10, 2014, from https://www.quandl.com/resources/useful-lists

[6] Dundar, M., Krishnapuram, B., Bi, J., & Rao, R. (n.d.). Learning Classifiers When The Training Data Is Not IID. Retrieved December 9, 2014. from http://people.ee.duke.edu/~lcarin/IJCAI07-121.pdf