Examining Long-Term Trends in Company Fundamentals Data

Michael Dickens

2015-11-12

Introduction

The equities market is generally considered to be efficient, but there are a few indicators that are known to have some predictive power over future price changes. This suggests that the market has some room for identifying inefficiencies. Much work is done on applying machine learning to short-term trading, but there exists little research on using machine learning to identify long-term inefficiencies; almost all mutual funds and hedge funds rely on the judgment of humans to make long-term bets about the market. Therefore, we have reason to believe *a priori* that there exist long-term market inefficiencies which can be found with machine learning.

To test this, I collect a set of fundamentals data across several thousand companies over a fifty-year period, taken from the CRSP/Compustat Fundamentals Annual database. First I apply linear regression over these fundamentals to see if they predict future returns; I find that they do predict returns better than any common indicators, but do not predict returns well on a risk-adjusted basis. Then I use support vector machines to classify stocks as high or low predicted returns. I find that for well-tuned parameters, an SVM can produce a classification where positive examples have high risk-adjusted returns and where this result generalizes well to a large test sample.

Terminology

Fundamental: A unit of information about a company taken from its income statement, balance sheet, or cash flow; e.g. assets total, net income, net sales.

Indicator: A unit of information about a company that ostensibly indicates something about a company's growth or value, and is typically derived from a ratio of fundamentals; e.g. price to earnings, return on equity, net assets.

Stock: Typically refers to a single share of a publicly traded company. Here it is used slightly differently to refer to a company and the fundamentals data.

Strategy: A function applied over a stock that produces a single real number predicting how "good" that stock is.

Score: The real number returned by a strategy for a particular stock.

Related Work

Very little published research has been done on applying machine learning to long-term stock prediction. There exists plenty of literature on applying machine learning to other financial problems, but minimal work has been published on fundamental analysis. One paper by Huang et al. (2005) [5] uses support vector machines to predict stock market movement, but the authors do not use company fundamentals as features. The most comprehensive work I have found is a dissertation by Andersen (2012) [2] which does apply machine learning over company fundamentals, but still uses only a limited set of fundamentals and contains limited discussion of their significance.

There has been substantial previous work examining the predictive power of company fundamentals. Fama and French (1992) [3] found a strong result by examining market deciles by size and book to market ratio. Greenblatt (2010) [4] published a book on how individual investors can outperform the market by following simple fundamentals-based strategies, and Abbey and Larkin (2012) [1] successfully reproduced his results. However, none of these use any strategies more sophisticated than finding stocks that rank highly on one or two fundamentals; if these simple strategies can outperform the market, perhaps we can find even better strategies by applying regression or classification using these fundamentals as features.

Dataset and Features

Getting Data

First, I have to get useful data on company fundamentals and prices and import this data into a useful format. I am using the CRSP/Compustat Merged Fundamentals Annual database available through Wharton Research Data Services. This data has annual stock quotes for many companies. I wrote software to import this data and produce an internal representation that maintains annual time slices of the market where each slice has data on all companies available during that year.

Identifying Features

We can draw on companies' published income statements and balance sheets to get fundamentals data. These data give us basic features that we can find for all companies in the Compustat database. These features include items such as net tangible assets, debt in current liabilities, net income, etc. We can also combine basic features to produce derived features: for example, price to earnings is computed as share price divided by earnings, where earnings is net income minus dividends on preferred shares.

There exist a handful of derived features where conventional investing wisdom suggests that they have some predictive power. Common derived features include price to earnings, return on capital, return on assets, debt to equity, etc. I originally considered exclusively using these features, but I believed it was possible that there exist features with predictive power that are not commonly used.

To identify useful derived features, I took a set of basic features and computed all possible ratios of these features. Ideally I would like to compute ratios of sums of features as well as other more complex combinations of basic features, but this becomes prohibitively expensive as the number of possible derived features expands rapidly.

I considered several methods for testing the predictive power of derived features. I decided to use a scoring function that divides the stock market into deciles according to a given feature, then returns the absolute value difference in average risk-adjusted returns between the first and last deciles. I chose this scoring function because it most accurately illuminates exploitable market inefficiencies. Using correlation coefficient as the score might find features with greater predictive power, but we care more about predicting which stocks lie at the tail ends of the returns distribution than identifying stocks in the middle, so this measure is less useful.

I was not able to find any features that had stronger predictive power than well-known indicators. Earnings yield performed by far the best, followed by other metrics that are very similar to earnings yield. A few unpopular features did have reasonably strong predictive power, but none did as well as well-known features such as earnings yield and return on capital. Perhaps I would find useful results by using more complex combinations of features instead of just ratios, but this quickly becomes computationally infeasible: the number of possible ratios is quadratic in the number of basic features, and if we used something more complex like a ratio of two sums, the runtime would increase with the fourth power of the number of features.

Methods

Measuring Returns

To judge a basket of stocks that some ML algorithm selects, we need to calculate that basket's future returns. To do this, we take the basket of stocks, compute the return on each of them, and take a weighted average of the results based on some predetermined weighting function. (The two most

obvious weighting functions are equal-weighted and cap-weighted (i.e. each company is weighted in proportion to its market capitalization).)

To compute the return for a company over n periods, we look at the future price over those n periods and compute the percentage price increase. This process is somewhat complicated by dividends and taxes. To include dividends, we take returns one year at a time and add in dividends at the end of each year. (Dividends typically compound quarterly, not annually, so this requires a little extra arithmetic.) To include taxes, we subtract income tax from the dividends each time they pay out and subtract capital gains tax each time we sell shares of a company.

How do we decide what basket of stocks to examine? I consider two different methods.

Deciles. We can measure a strategy by dividing stocks into deciles by their score for that strategy. More predictive strategies should show greater differences between deciles, and especially a greater range between the first and tenth deciles.

Top N. Measure returns on the n top-ranked stocks according to a strategy. This method more realistically reflects how a strategy would behave in practice for any investors who do not have enough capital to replicate an index fund.

The top-n method provides results that are more useful in practice, while the deciles method gives a better sense of a strategy's predictive power. Here I primarily use the top-n method.

Note that we want to measure both returns and standard deviation. If a strategy gives X returns at standard deviation Y, we can use the strategy to get $2X$ returns at standard deviation $2Y$ by using $2:1$ leverage. Higher returns are not meaningful unless the standard deviation is appropriately low.

Results

Baseline Result

There are a few simple well-known fundamentals indicators that predict future stock returns. For a novel strategy to be valuable, it must have better predictive power than any simple fundamental.

One of the most well-known fundamentals is price to earnings. Here I use a slight variation, earnings yield, which is known to have somewhat better predictive power. A strategy must have better risk-adjusted return than earnings yield to be worth using.

Linear Regression

To apply linear regression, I took a set of 16 indicators that are already known to be useful according to conventional investing wisdom. Linear regression can be applied over these indicators by taking the list of indicators for a training example as the x value and using this to predict annual return (the y-value). I applied a linear regression using a subset of fundamentals and returns data from 2010 as the training set. Then I tested a strategy where I buy the 30 stocks that are predicted to get the best returns according to linear regression. I tested this strategy over the period 1963 to 2013. I compared the strategy against earnings yield as a baseline. I also compared it against dividend yield because the regression weighted dividend yield most heavily of all the fundamentals it used.

Linear regression weakly outperforms both earnings yield and dividend yield, but does not have better risk-adjusted return (see Table 1). I attempted some modifications to the regression but was not able to produce substantially better results.

The regression plot of all stocks tested show that almost all stocks lie within a fairly thin band and there are a few outliers that dramatically change the shape of the line. It would be unreasonable to remove these outliers; any real-world stock purchasing strategy cannot identify and remove outliers in advance.

Unlike with SVM, I found that adding more indicators as features to the regression did not reduce accuracy. The regression algorithm assigned weight 0 or near-0 to most indicators.

Linear regression finds stocks with high return but not high risk-adjusted return. Because of the way risk is calculated, it is not feasible to measure risk-adjusted return over a training set, so we cannot use risk-adjusted return as the y axis for the regression. For future work on this subject, it would be useful to find a way to measure risk over a training set. However, it is not obvious that this would work. Even over a linear regression that optimizes for return, it still barely has higher un-adjusted return than a simple dividend yield strategy.

Support Vector Machines

The problem of predicting long-term stock returns can be conceptualized as a regression: returns fall on a continuum, and we can predict a stock's position on the continuum. It is perhaps somewhat unnatural to conceive of this problem as a classification problem. On the other hand, in practice we want to classify stocks as "buy" or "don't buy". I used support vector machines to classify stocks in this way. A support vector machine produces a dividing hyperplane that categorizes stocks into "buy" and "don't buy" categories. SVM also allows the use of the kernel trick to efficiently classify over many features.

Before beginning this project, I hypothesized that, instead of using well-known indicators like price to earnings, I could produce more novel results by using combinations of basic company fundamentals. As discussed in "Identifying Features" above, it appears unlikely that there exist any simple ratios that have good predictive power. Nonetheless, thanks to the kernel trick, it's easy to implement an SVM that uses ratios of fundamentals as features.

Similarly to with linear regression, I trained on a subset of stocks from a single year and tested on the remaining stocks over every year. To convert returns into binary classification, I treated a stock as a positive example if it got a one-year return greater than some cutoff. I tried a range of values for this cutoff to see what worked best. I initially used 15%–a company that gets 15% returns solidly beats the market, but it's still sufficiently low that lots of stocks will perform this well. However, I found that this cutoff was close enough to average market return that classification produced very noisy results with limited predictive power. I found more success using 25% as the cutoff–lower values are too noisy, and higher values produce too few positive examples.

I tested SVM using basic fundamentals as features and using a kernel to produce their ratios; I also tested SVM with the same set of indicators used for the linear regression. Both of these produced worse risk-adjusted returns than linear regression. They worked relatively well on the training set, but did not generalize to the test set. Using such a large number of features resulted in severe over-training.

I was able to correct this problem by substantially reducing the feature set. I took three features known to perform well individually (earnings yield, return on capital, dividend yield) and found that SVM dramatically outperformed the baseline earnings yield strategy with an extraordinary annualized return of 27.4%. Even so, SVM with these features did not perform substantially better than a simple strategy that ranks stocks by earnings yield and return on capital (see Greenblatt (2010) [4] or Abbey and Larkin (2012) [1]).

The SVM's predictive power is fairly sensitive to the training set. In the interest of caution, I wanted to use a fairly small training set and as large a test set as possible. I used a subset of stocks from a single year as the training set, but I found that which year I used had a nontrivial effect on the resulting support vectors. When I trained on a subset of stocks from the year 2005 instead of 2010, the algorithm produced an SVM with very weak predictive power. I am wary of using many different training sets and picking the best one; the more training sets I use, the more likely I am to get a strong positive result just due to chance. I found an annualized return of 27.4% using just the year 2010 and without testing many years, so this positive result is not simply the result of trying lots of training sets. Perhaps this concern is overstated, but it is easy to become overly optimistic about an investment strategy's ability to beat the market, so I believe caution is warranted.

Conclusion and Future Work

There do exist indicators such as earnings yield that have substantial predictive power, and it is possible to classify stocks using these indicators to produce good risk-adjusted returns. We can produce better risk-adjusted returns by using SVM to classify stocks according to several indicators with strong predictive power.

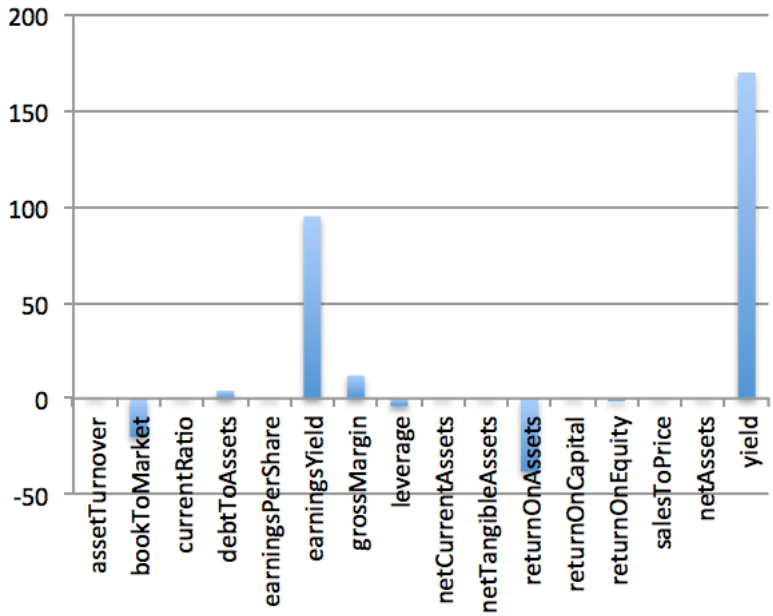
More work should be done to test the robustness of this conclusion. Does it apply independently in multiple sectors of the market? Does it hold in equities markets outside the United States? Additionally, more analysis is needed to determine if SVM produces better risk-adjusted return than simply ranking stocks by a small set of useful indicators.

Tables

Table 1: Returns

Name	Risk-Adjusted	Return	Stdev
Earnings yield	0.65	16.3	25.0
Dividend yield	0.59	17.9	30.5
Linear regression	0.56	18.1	32.2
SVM	0.91	27.4	30.1

Table 2: Linear Regression Feature Weightings



References

[1] Abbey, Boris S., and Patrick J. Larkin. *Can Simple One and Two-Factor Investing Strategies Capture the Value Premium?* Applied Finance: 149, 2012.

[2] Andersen, A. C. *A Novel Algorithmic Trading Framework Applying Evolution and Machine Learning for Portfolio Optimization*. Doctoral dissertation, Masters Thesis, Faculty of Social Science and Technology Management, Department of Industrial Economics and Technology Management, 2012.

[3] Fama, Eugene F., and Kenneth R. French. *The crosssection* of expected stock returns*. The Journal of Finance 47.2, 1992.

[4] Greenblatt, Joel. *The little book that still beats the market*. John Wiley & Sons, 2010.

[5] Huang, Wei, Yoshiteru Nakamori, and Shou-Yang Wang. *Forecasting stock market movement direction with support vector machine*. Computers & Operations Research 32.10, 2005.