Predictive Modeling of Stock Market Technical Analysis

Steven Miller

**Predictive Modeling of Stock Market Technical Analysis**

**Abstract**

Technical indicators have been used for decades by traders to gain better understanding of the movement of stock prices. The idea is that by better understanding the direction of price movement, better investment decisions can be made. Many traders are split on the concept of technical analysis. Some are strongly in favor of it, swear by it, and use it exclusively to make their investment decisions. Others are in the middle, they may use it as part of their decision making, but it is not the only factor analyzed. A final group considers it to be entirely worthless, lumping the concept in with the likes of astrology and feels that investors would be just as well off tossing a coin in the air than they would be to make their decisions based on technical analysis.

**Introduction**

Technical indicators are essentially formulas created to reflect the price movement of a stock. Dozens of these indicators exist, including the Simple Moving Average Crossover, the Relative Strength Index, Bollinger Bands and many more. All these indicators can be powered by numerous different input values. The Simple Moving Average Crossover, for example, looks at a short-term and long-term moving average. There are more common values, such as a 90-day short-term average and a 200-day long-term average, but realistically any number could be plugged into this formula to generate buy and sell signals for a stock. The goal of this project will be to focus on the SMA Crossover, looking at a wide variety of possible input values and testing them across many stocks to see if any of these strategies generalize well across a wide breadth of stocks.

**Methods**

The starting point of this project was to define the overall scope. For this project, that means deciding how many stocks to look at and how many indicators to analyze. Initially, I wanted to look at a wider variety of technical analysis indicators, but time became a limiting factor and I chose to look deeper at just a single indicator, the SMA Crossover. For the number of stocks to look at, 100 seemed like a round number that would be large enough to prevent the results of tests on any one stock from carrying too much weight on the overall results. For the analysis of the SMA crossover, I chose to look at every possible combination of 5-day increments for short and long-term moving averages from 5 to 200, with the obvious constraint that the short-term value must be less than the long-term value. I used independent combinations of these periods for entry and exit signals. For example, I might create a strategy where I choose to buy a stock when the 30-day moving average exceeds the 90-day average, but I would sell it when the 45-day average drops below the 120-day average. The buy signals and sell signals remain independent of one another. The result of this was a total of 608,400 possible combinations per stock.

Next, I needed to determine how results would be measured. While the overall goal was to build a classification model that predicted whether or not to make a trade on a given day, the typical metrics we would typically associate with a classification model such as precision, recall and F1 scores would not necessarily be enough to declare a model successful for this purposes. What if we achieve a model that is technically more accurate than taking every trade signal given, but it does so by missing out on some trades that would have been very profitable? Can we call the model a success if it is better by traditional metrics, but is not as profitable as a baseline? I believe not.

Once the goals of the project and scope were defined, it was time to begin. I sought out various sources of historical market data but settled on AlphaVantage for its easy to use API. One challenge of AlphaVantage’s free API was the rate limit. Only 500 API calls can be made in a day, and no more than 5 per minute. AlphaVantage does offer the ability to query many technical analysis indicators, but if I’m looking at 100 stocks and trying to query 20 different simple moving averages for each stock, I would hit the rate limit quickly each day. Spending multiple days just gathering data wasn’t really an option. I chose to retrieve simply the market data from AlphaVantage and calculate moving averages locally. I used a Python API wrapper I had previously written for AlphaVantage to make the process smoother.

At the point of data acquisition, I had a decision to make. I could either store all these results as text files and read or write from them as necessary, or I could take some extra time to create a database to store the results in. I chose the latter, and that paid off moving forward.

To determine the 100 stocks moving forward, I decided I wanted to look at the liquidity of each stock, that is the frequency each stock is traded. Using the website Finviz, I was able to screen stocks based on their average volume and filter this down to the top 100. Next, I wanted to split this group of 100 stocks into three categories based on the behavior of the stocks. To accomplish this, I used the Augmented Dickey-Fuller test. This test is used to determine whether a time series is stationary or not. Based on the results of this test, I assigned each stock to a group, either stationary, non-stationary, or undetermined. 90 of the 100 stocks analyzed fell into the non-stationary category. This means that in the long-term the time series shows a trend, either moving up or down, rather than reverting to the mean. Because 90 of the 100 stocks were non-stationary, I chose to ignore the other 10 for the remainder of the project. Had the numbers been closer to even, I would have likely moved forward developing a model for each category.

Once I had selected the 100 stocks I wanted to analyze, and filtered this down further to just 90, it was time to begin the most time-consuming portion of this project, the backtesting. Backtesting is the process of using historical data to systematically test a trading strategy. Essentially, simulating the performance of a trading algorithm historically. When done on a single stock, this can be misleading as it’s easy to overfit a trading algorithm to the trends of a single stock. This doesn’t necessarily lead to quality results in the future. One reason for choosing so many stocks to analyze in the first place was to see if any indicators generalized well across a wider breadth of stocks.

Running each of these tests took about 12-hours to complete per stock. I was able to run approximately 12 tests simultaneously fortunately, but it still took me several weeks to finish as I was performing this task on my home computer and had to let it run at times I didn’t need the computer for anything else for the next 12 hours. If I move forward with testing any indicators in the future, I may investigate cloud-computing solutions to this challenge where I can run thousands of tests simultaneously.

Once all the backtesting was complete, I was able to analyze the results. This is the point where the decision made earlier to load everything into a PostgreSQL database rather than raw text files paid off. Creating 608,400 backtests for 90 stocks led to a total of 54,756,000 tests to aggregate and analyze. Had this all been saved as a CSV file, it would have been far too large to work with easily inside of Pandas. The table of trade results was around 24 gigabytes. Instead, aggregating the results was as simple as writing an SQL query with a GROUP BY clause.

I wanted to analyze the results of the trades based on three factors: Sharpe Ratio, Calmar Ratio, and Sortino Ratio. These ratios are all different methods of measuring the same concept, the measure of returns of a strategy compared to the risk taken on by the strategy. To aggregate the strategy results, I queried my table of strategy results and grouped them by the specific strategy used. The median Sharpe, Calmar, and Sortino ratios were averaged to determine the best performing strategies. From here, I selected the ten best strategies using a unique entry criterion. There were some strategies that did perform better than some of the ten used, but their entry signal was the same as another strategy already in use, and I felt it would be difficult in reality to use a strategy that had two potential exit signals, so these were omitted.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| avg medians | short\_entry | long\_entry | short\_exit | long\_exit |
| 0.2160 | 60 | 200 | 65 | 165 |
| 0.2113 | 70 | 200 | 70 | 190 |
| 0.1750 | 120 | 185 | 50 | 175 |
| 0.1733 | 145 | 195 | 120 | 165 |
| 0.1670 | 65 | 195 | 70 | 175 |
| 0.1667 | 65 | 200 | 70 | 175 |
| 0.1639 | 110 | 200 | 55 | 160 |
| 0.1635 | 60 | 195 | 65 | 165 |
| 0.1621 | 170 | 200 | 50 | 175 |
| 0.1613 | 85 | 200 | 55 | 175 |

Strategy criteria used in predictive model

After selecting the ten strategies to be used for my predictive model, I re-reran the backtests using these ten strategies, but instead of simply aggregating the results of the tests, I built a table of every trade generated by each strategy across all 90 stocks. This built the foundation of the input data for my predictive model. I then transformed this table by creating dummy variables for each strategy to have their own column. In some cases, multiple strategies generated a buy signal on the same day. Creating dummy categorical variables and merging these trades together allows for the model to understand when a buy signal is received from multiple strategies on the same day. This became the input data for the model. The output data for the model was simply a binary value indicating whether a trade was profitable it was taken after a signal was given.

Before training the model, I wanted to establish some performance baselines. I split the data off into a training and test set of data. One-third of all trades were held behind in the test set. I ran some tests on this data to see what the baseline performance was if I had made every single trade recommended by the signals in the base set. The results were that 49.762% of all trades generated by signals were winning trades, a 4.6694% average return on all signal-generated trades, and a 0.0% median return on all signal-generated trades. The large difference between mean return and median return is likely best explained by several trades that were significantly longer in duration than most but were also very profitable. As the duration of trade is not accounted for, this simply shows up as a very profitable trade with no consideration given to the long-term allocation of capital.

**Results**

To build the final model, I chose to use a random forest classifier. The model used 10,000 estimators, which was likely excessive in hindsight. The model performed with an accuracy higher than the baseline win rate of the test set, but the first thing I notice is that it is still taking a vast majority of the trades offered, despite only 49.762% of them being profitable.

|  |  |  |
| --- | --- | --- |
|  | **True** | **False** |
| True | 7226 | 1762 |
| False | 6967 | 1862 |

* + Accuracy: 51.00%
  + Precision: 51.38%
  + Recall: 21.09%
  + F1: 29.90%

To get an idea of how this model would work in practice, it was necessary to look at the financial results of the model on the test set. What I found was that 51.38% of all trades taken by the model were profitable, which was an improvement over the 49.762% from the baseline. 3.2986% was the average return, which was a reduction but could have easily been the result of one or two profitable trades not being taken. Finally, the median trade went from 0% to 0.08%.

**Conclusion**

There’s some evidence here to suggest that implementing a predictive model using technical analysis could be a profitable trading strategy, but I’d need to do further work and analysis before I would be willing to put any of my own money behind that. Moving forward, I would like to investigate implementing additional indicators to the predictive model. I was deliberate when designing my code to make sure that I didn’t make any decisions that would leave me too focused on just the SMA Crossover. The only limit of adding additional indicators is the very long time it took to complete backtests. I would also like to investigate adjusting the threshold of the predictive model. By default, the classifier is simply generating a buy signal if the probability of a profitable trade is predicted to be above 0.5. With the large number of trades still being taken by the model, I would like to see if raising this threshold results in fewer trades being taken, but also fewer unprofitable trades being taken in the process.

Finally, I would like to adjust the analysis of returns by accounting for the duration of the trade. Many mean values related to returns were skewed by trades that were in place for years being treated the same as a trade that took place over a few days. Efficiently allocating resources is important, and this isn’t necessarily captured well by the current model. Finally, I would like to investigate the derivatives market using similar approaches. With stock options, I believe there is even more data to be uncovered and analyzed. Additionally, it would be an easier area to diversify in with a lower initial cost of capital, making real-world testing a feasible option.

To conclude, this was an interesting project to take on. Like most real-world projects, the actual model building was a very small component of the overall task, which was primarily cleaning, transforming, and analyzing data. While the results weren’t spectacular, I believe promise is shown and I’m looking forward to continuing to develop upon this idea moving forward.

**Acknowledgements**

I’d like to thank my girlfriend for listening to make talk about this project even though she probably has zero interest in the idea. I’d also like to credit Van K. Tharp, whose books on trading gave me the initial interest in pursuing a project like this. Finally, I’d like to thank the team at TastyTrade, whose daily presentations on the findings of different option-trading strategies inspired me to pursue my own tests.

References

AlphaVantage. (2017). API Documentation | Alpha Vantage. Retrieved September 6, 2019, from Alphavantage.co website: https://www.alphavantage.co/documentation/

‌Finviz. (2019). FINVIZ.com - Stock Screener. Retrieved November 14, 2019, from Finviz.com website: http://finviz.com

Investopedia. (2019, June 25). Using Technical Indicators to Develop Trading Strategies. Retrieved September 5, 2019, from Investopedia website: https://www.investopedia.com/articles/trading/11/indicators-and-strategies-explained.asp

Managed Futures Investing. (2016, May 24). 8 Risk (Adjusted) Statistics Summarized | aiSource. Retrieved November 14, 2019, from aiSource website: https://www.managedfuturesinvesting.com/8-risk-adjusted-statistics-summarized/

‌