Using Google Trends to Trade Stocks

Version 1.0

Shaefer Drew

Abstract

Automated trading has become exponentially popular in recent years with the abundance of big data. The majority of today's trades are executed by computer. Quantitative Analysts, or "Quants", are often hired to program automated trading algorithms which influences the way a firm trades or invests its money. The problem I'm working on is using trend data to build a trading algorithm which predicts stock price movement, maximizes profit, and minimizes risk. I am using time series analysis, measuring popularity trends and their correlation to stock price in conjunction with a mean reversion strategy to predict whether to buy or sell a stock. Here I show that my algorithm outperforms baselines in profit, non-negative profit change, and precision for 3 different stocks over 4 time intervals. These results indicate that my approach can effectively outperform other strategies, at least for certain stocks and time intervals. Thus, this trading algorithm can serve as a useful tool for trend-based trading.

1 Introduction

Stock market analysis is at the forefront of data science research. Algorithmic, computer-driven trading has increased exponentially throughout the years. As of now, the quantitative hedge fund industry is on the brink of surpassing $1 trillion in assets under management as investor preferences switch from traditional strategies to more systematic strategies(Wigglesworth, 2018).

Every single day, people are exploring strategies to beat the market. According to efficient market theory, consensus is built into the market. Therefore, this requires people to think differently about strategies and explore methods that haven't been implemented. There is an entire industry based on beating return benchmarks such as the S&P 500 index. Since my algorithm successfully beats its benchmarks and provides some insight into the relationship between popularity sentiment and stock trends, there are plenty of individual traders and money management firms who can benefit from my approach.

2 Problem Definition and Data

The broad problem I am addressing is the same as every other quantitative analyst in the stock market: how to maximize returns. The more specific problem I am addressing is figuring out how to use trend data to maximize profit, precision, and minimize risk. My algorithm will search for buy and sell signals in my datasets based on defined criteria. I am not predicting the stock price and I am not predicting the direction of the stock price at every point in time. Rather, I will only predict the direction of the stock price during the times when my criteria is met.

I use a mean reversion strategy in buying and selling equities. If there is an overreaction to a mild positive trend, I short the stock. If there's an overreaction to a mild negative trend, I buy the stock. If there's an under-reaction to a very positive trend, I buy the stock. If there's an under-reaction to a very negative trend, I short the stock. I sell or cover the stock when its price crosses its moving average.

The logic behind this strategy assumes trend data and stock price data to be roughly correlated. When a stock is doing well, people are bullish and search for it more because they are considering purchasing shares. When a stock is doing poorly, people are bearish and fewer people are looking into buying it. Therefore, when the price and search trend don't move together as expected, my algorithm sends a signal to act on this deviation until it corrects itself.

	price	baseline_rec	num_positions	profit
2018-09-09	27.379999	cover and buy	10	174.20004
2018-09-16	32.720001	sell and short	10	227.60006
2018-09-23	31.020000	short	20	244.60007
2018-09-30	30.889999	short	30	247.20009
2018-10-07	27.350000	short	40	353.40006

Figure 1: Baseline Recommendation Example

I back-tested my algorithm, purchasing stocks based on its signals and accumulated a total profit, precision, and non-negative profit change over time.

My primary data set is from the Google Trends API, pytrends, with search terms being the stock's ticker. The dataset is 157 rows for each 3 year time interval, with an index of weekly dates dating back to October 2012 for the earliest interval. Its only attribute is "interest over time", as measured by total searches per week for the stock ticker in the "finance" category.

I also used stock market data from the Yahoo Finance API. I pulled data from the same time frame and ticker as the trends data. The data source was comprised of 756 rows, a daily index, an open, high, low, close, adjusted close price, and a volume attribute.

I re-sampled the Yahoo dataset by week and considered its last closing price. After doing so, I merged this with the Google Trends dataset. I created rolling averages for each trend. The resulting dataset had a date index and 4 columns: price, number of searches, rolling price, and rolling trend. It was 157 rows long.

I split the data set into training and test data, using a 70% 30% split.

The final product is the test data table consisting of 17 columns and 48 rows. It contains a stock purchase signal column and a profit column (in addition to others).

An alternative source I considered was the Google News API to perform sentiment analysis on headlines to determine positive or negative sentiment for a stock. Unfortunately, after connecting to that API, I was only able to obtain headlines from up to a month ago. A month's-worth of data doesn't give me a sufficient amount of observations to calculate a trend or make a confident prediction.

Another data source I considered pulling from was StockTwits (Twitter for stocks), which has sentiment data based on whether a "twit" about a stock is marked as "bearish" or "bullish". Using StockTwits data is another possible way I could refine my algorithm. However, for now, it only uses Google Trends data and stock price data.

My algorithm searches for significant deviations in stock price and popularity trends and compares the two by normalizing the data and using certain confidence intervals as signal criteria.

My approach works because it looks for instances where sentiment isn't yet reflected in the stock price, and buys or sells stock until this discrepancy vanishes.

3 Related Work

One approach used sentiment from news data to predict whether a stock goes up or down(Joshi et al., 2015). They did so using sentiment analysis. They counted positive and negative words related to finance and outputted a polarity score. They used this score as well as a tf-idf vector space as input for different classification models. These models predicted whether the stock went up or down. The models provided a directional accuracy measure, and they were able to plot sentiment polarity alongside stock price. However, they didn't have a method of measuring correlation of analyzing the graph any further. They also measured their success differently. Their goal wasn't to make return, but to predict the direction of a stock for each data point. After seeing the initial results of my algorithm, I may include this metric in my evaluation. Since I don't predict movement for each observation, the accuracy measure would only include those dates in which I did have a prediction. In a way, it could be considered a detection rate. (percent of the time I short the stock and it goes down / buy the stock and it goes up).

Another report attempted to predict the S&P 500 market trend using Google Trends. They searched for keywords such as "stocks to buy" and "how to short sell" in order to interpret bullish or bearish sentiment at a certain point in time(Capital, 2018). They then created a sentiment index, which took the difference between bullish and bearish searches and fit the data with a 6-month moving average. This study's goal aligned with mine: impliment a trading algorithm that maximizes return over time. The algorithm used invested in consumer staples (XLP) when the sentiment index was negative and consumer dis-

Dataset	Description	Website
Google Trends	Time series data of quantity searched over time	https://trends.google.com/trends/
Yahoo Finance	Stock Data	http://finance.yahoo.com

Table 1: Datasets

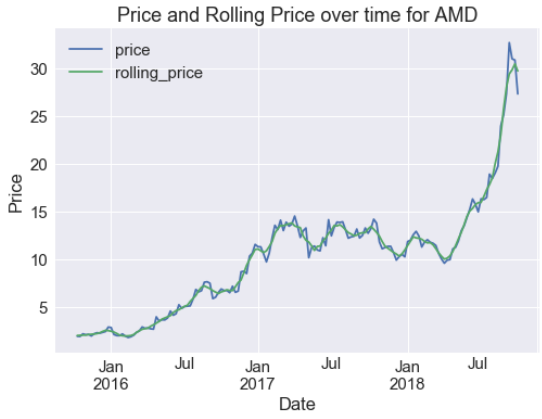

Figure 2: Price and Rolling Price over time for AMD

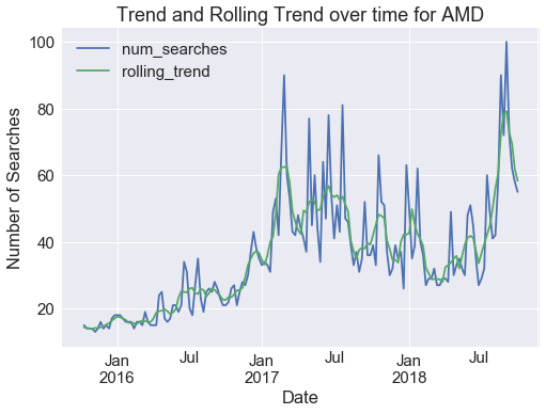

Figure 3: Trend and Rolling Trend over time for AMD

cretionary (XLY) when the sentiment index was positive. Since I am likely abandoning any news APIs, another route I could take is creating a sentiment index.

Another research paper expands on the famous Bollen paper which attempts to predict stock movement based on twitter sentiment. One strategy they used is granger causality analysis with different lags(Mittal and Goel, 2011).

My study shows the correlation between price and trend movement and compares them at different lags. I have not performed granger causality because I don't believe it would be of much benefit. I already understand that the highest correlation is at lag 0, which means prices and trends tend to influence each other in the same weekly time frame. My algorithm doesn't assume one causes the other; however, assumes they are correlated and exploits instances of weak correlation. While it would be interesting to study, finding causation would be out of scope.

4 Methodology

After merging the Google Trends and Yahoo Price data, I used the ASAP approach to create a 5 week rolling mean of trend and stock data. I experimented with cross-correlation for the two trend lines and found the highest correlation is .78 at a lag of 0 (for the 2015-2018 time interval).

I measured the differences between the trend and its moving average as well as the price and its moving average. After splitting the data into train and test, I normalized these differences in train data using a z-score method and defined its confidence intervals (for deviation) as those I would use on the test data. After normalizing the differences for test data, I wrote my recommendation algorithm to produce a buy, short, sell, cover, cover and buy, or sell and short signal. These signals are determined by trend and price differences and their position on the confidence intervals I determined from the train data. Using the signals, I start with a portfolio of $1000 and invest a certain number of shares for each signal (dependent on pre-determined risk)

I explored different anomaly detection techniques, from simply measuring the deviation from the smooth moving average to machine learning techniques, in order to identify overreactions. I tested these 2 techniques. First, I generated signals based on significant deviation from the moving average on one trend and less on the other. The second technique involved using unsupervised machine learning to detect anomalies, which ended up only detecting extreme anomalies and wasn't of much use. I used the first approach and tested it with different intervals. The intervals I ended up using were 75%, 60%, 40%, and 25%.

I backtested my algorithm over 4 different time periods: 2012-10-08 to 2015-10-08, 2013-10-08 to 2016-10-08, 2014-10-08 to 2017-10-08, and 2015-10-08 to 2018-10-08.

I customized risk for the trend following strategy and my algorithm to reflect the maximum number of shares to trade at once without ever having to use leverage. This method is still imperfect; however, is the most reliable method tested which accounts for leverage.

I tested my algorithm along with its benchmarks with 3 different stocks (AMD, SPY, and NVDA). The results are shown in different output tables.

5 Evaluation and Results

Success is measured as the profit, precision, and risk of my investment strategy compared to others.

Resulting metrics include profit, precision, and my risk measurement: non-negative profit change (posn_profit_change). Profit is measured as revenue - cost, precision calculates the proportion of correct predictions on a weekly basis. It only considers weeks where recommendations were made, and of those weeks, its accuracy in predicting the price movement from the current to the following week. This metric should be considered with scrutiny because not all of the positions are opened and closed in a week's time. Non-negative profit change is measured as the proportion of weeks where profit is increasing or staying neutral. This metric allows me to measure risk.

I measured my algorithm's performance against 2 benchmarks. The first benchmark is the buy and hold strategy. This is simply purchasing a certain number of shares and holding. The second benchmark is the trend following algorithm, which purchases the stock when the trend increases and sells the stock when it decreases (adjusting for risk and avoiding leverage).

My algorithm outperformed its benchmarks in all 3 metrics for each stock and in each time period. Out of the 3 metrics, it performed significantly better in terms of profit and non-negative profit change. Since my algorithm tends to use more risk, this may be a factor as to why it is more profitable. Non-negative profit change is not dependent on profit. I consider this the most reliable metric. It tells me risk (by accounting for upside and downside). Since it only measures risk, it should still be used in conjunction with profit in order to determine the algorithm's success. Precision measures the "correctness" of the algorithm in determining price movement; however, has its limitations due to positions with different holding windows.

timeframe	type	posn_profit_change	precision	profit	risk
2012-2015	base	0.574468	0.500000	8.470000	121
	buy_n_hold	0.531915	NaN	-94.380000	121
	my	0.808511	0.555556	313.200000	108
2013-2016	base	0.510638	0.523810	9.000000	50
	buy_n_hold	0.638298	NaN	246.000000	50
	my	0.893617	0.730769	1466.400000	130
2014-2017	base	0.625000	0.600000	144.160000	16
	buy_n_hold	0.583333	NaN	104.640000	16
	my	0.937500	0.828571	696.000000	25
2015-2018	base	0.541667	0.488372	324.640144	16
	buy_n_hold	0.625000	NaN	257.440000	16
	my	0.812500	0.533333	423.840080	16

Figure 4: AMD Performance Results

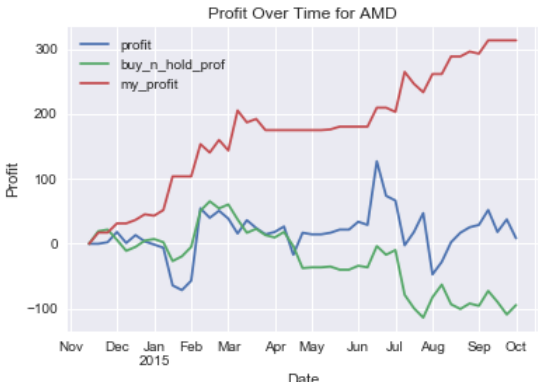

Figure 5: Profit Over Time for AMD (2012-15 Interval)

The results indicate that my method is more profitable, more precise, and less risky than the benchmark methods for the chosen time periods and stock bundle. Other algorithmic traders should, therefore, consider this method over others when using popularity sentiment to trade stocks.

6 Discussion

At first, my baseline algorithm did not do very well from multiple standpoints. It underperformed one of the other baselines and, at a glance, didn't seem to be very accurate.

Mostly, it did not perform well because I failed to account for leverage. My algorithm accounts for risk in the amount of shares it buys at a time; however, the baseline was constantly using leverage. Therefore, I had to use a very low risk metric in order to compare it to my algorithm which limited returns significantly.

I took 2 different approaches to this issue. First, I built a function that adjusts recommendations to as to account for using zero leverage. It reduces

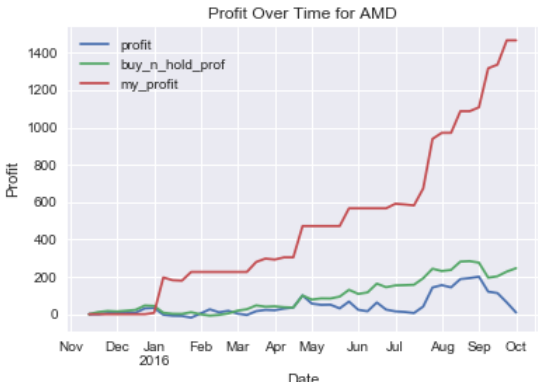

Figure 6: Profit Over Time for AMD (2013-16 Interval)

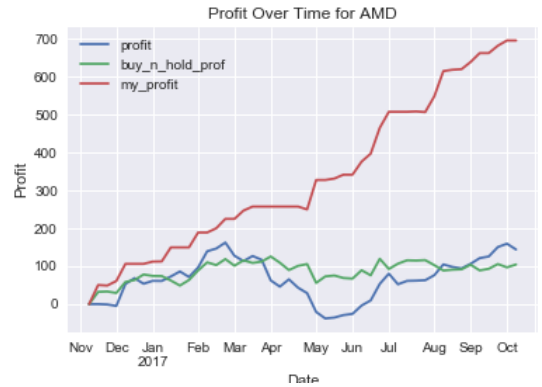

Figure 7: Profit Over Time for AMD (2014-17 Interval)

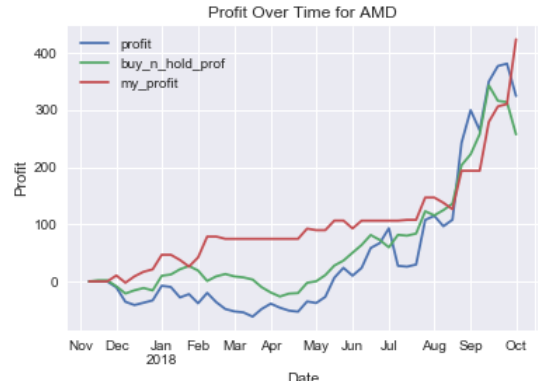

Figure 8: Profit Over Time for AMD (2015-18 Interval)

the amount of transactions made at once in cases where leverage would be used otherwise. The result is the risk value that returns the highest profit. At first I used this method to measure success, and it was successful in all 4 time intervals for AMD against its benchmarks. The problem was that this method will often not adhere to the signal at all if its risk number is too large to use without leverage. Since it isn't considering all of the recommendations and since I don't know the optimal risk until after the test data is revealed, I instead used a method where I found the maximum risk that allows for all recommendations without using leverage. I did this for both the baseline and and my algorithm and calculated the preferred risk for each. My algorithm also outperformed its benchmarks using this method.

7 Things Left Undone

Primarily, I do believe there are better ways of accounting for leverage and hope to impliment a new method for that.

Additionally, I would like to automate my program more so it is easier to run on multiple stocks. I want to easily be able to select a bundle of stocks and backtest them over different time intervals.

I also want to impliment different performance metrics and/or see if I can use my methodology in conjunction with certain backtesting libraries or software (since a majority of my bugs came from backtesting). In addition, knowing how the profits compare before and after brokerage fees could be helpful to individual investors.

Lastly, I hope to explore this same type of analysis with news sentiment, which was my initial goal but I hit a road block with the limited free availability of the Google News API.

8 Work Plan

My plan is to first figure out how I measure sentiment. Ultimately deciding which data source best correlates to stock sentiment. This will likely involve experimentation with different data sources and different sentiment algorithms.

The next step is to plot the data and make observations. Afterwards, I hope to create smooth moving averages of the charts. Then I will identify the deviations in each trend and see which overlap, measuring correlation/causation strength in each case.

Afterwards, I will backtest my algorithm based on the criteria of differing deviation and analyze the results against my baselines.

I have accomplished mostly everything in my plan with a few minor changes. I didn't end up measuring causation because it was out of scope.

In regards to backtesting, I am still optimizing my strategy and experimenting with different ways of measuring return and success(accuracy).

I plan on first accounting for leverage and creating a useful return measurement for my baseline. Next, I will optimize my algorithm and see which criteria cause it to produce greater return.

In producing greater return; however, there are often certain trade-offs such as market volatility. Therefore, my next step will be to incorporate different risk metrics.

Finally, I will analyze other stocks and other bundles of stocks and come up with a mean score for each scenario.

If time permits, I will go back and reinvent my trend line to incorporate sentiment.

Final Update: I accomplished most of my goals I stated in my work plan. I used Google Trends to measure sentiment, accounted for leverage by optimizing risk, used precision instead of accuracy, and explored different performance metrics.

9 Opt-out of online publication

I wish to opt-out of online publication so I can take advantage of my algorithm.

10 Acknowledgements

Thank you David Jurgens for your valuable data mining lectures and assignments.

Thank you Daniel Romero for your machine learning introduction, I still use many concepts learned in your class to this day.

References

Nairu Capital. 2018. Using google trends to predict stocks. *Seeking Alpha* page 4.

Kalyani Joshi, Bharathi H.N., and Jyothi Rao. 2015. A large annotated corpus for learning natural language inference. *arXiv preprint arXiv:1508.05326* .

Anshul Mittal and Arpit Goel. 2011. Stock prediction using twitter sentiment analysis page 4.

Robin Wigglesworth. 2018. Quant hedge funds set to surpass $1tn management mark. *Financial Times* .

timeframe	type	posn_profit_change	precision	profit	risk
2012-2015	base	0.574468	0.543478	69.479982	9
	buy_n_hold	0.617021	NaN	45.089982	9
	my	0.765957	0.606061	563.640044	22
2013-2016	base	0.595745	0.534884	166.899990	5
	buy_n_hold	0.638298	NaN	193.599990	5
	my	0.829787	0.548387	306.180000	9
2014-2017	base	0.479167	0.434783	46.009953	1
	buy_n_hold	0.666667	NaN	93.330002	1
	my	0.812500	0.444444	152.379974	2
2015-2018	base	0.416667	0.400000	-45.179973	1
	buy_n_hold	0.604167	NaN	53.719986	1
	my	0.666667	0.540541	560.760080	2

Figure 9: NVDA Performance Results

timeframe	type	posn_profit_change	precision	profit	risk
2012-2015	base	0.553191	0.523810	-9.230050	1
	buy_n_hold	0.553191	NaN	-9.240005	1
	my	0.851064	0.750000	130.839983	1
2013-2016	base	0.382979	0.340909	-64.759960	1
	buy_n_hold	0.617021	NaN	13.760010	1
	my	0.787234	0.517241	93.160015	1
2014-2017	base	0.437500	0.391304	-12.229994	1
	buy_n_hold	0.625000	NaN	37.949997	1
	my	0.750000	0.531250	71.520106	2
2015-2018	base	0.479167	0.454545	-94.009943	1
	buy_n_hold	0.625000	NaN	29.730011	1
	my	0.875000	0.680000	297.329772	3

Figure 10: SPY Performance Results

11 Supplemental Material

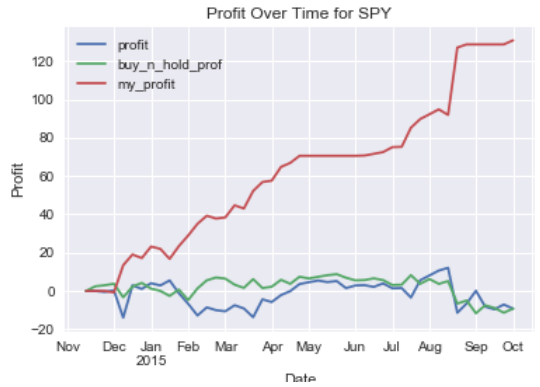

Figure 11: Profit Over Time for SPY (2012-15 Interval)

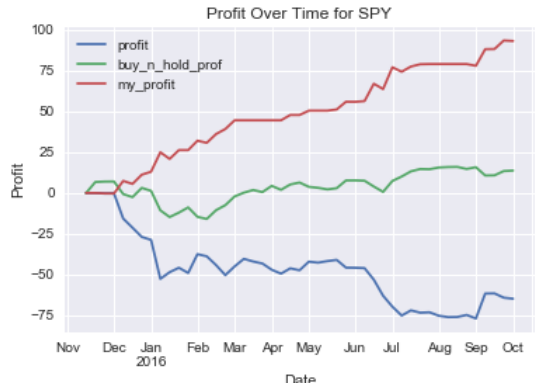

Figure 12: Profit Over Time for SPY (2013-16 Interval)

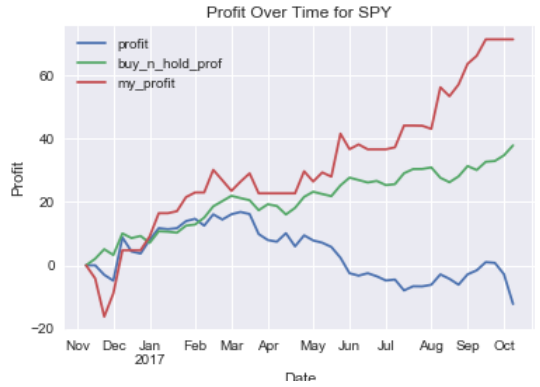

Figure 13: Profit Over Time for SPY (2014-17 Interval)

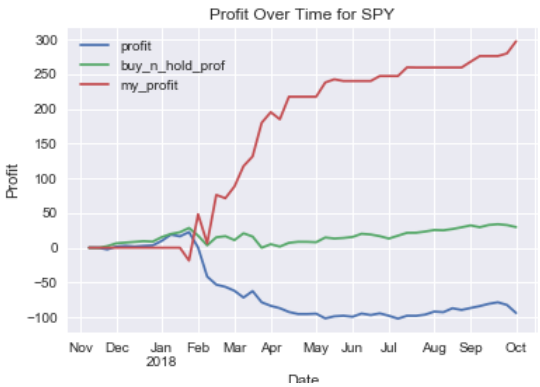

Figure 14: Profit Over Time for SPY (2015-18 Interval)

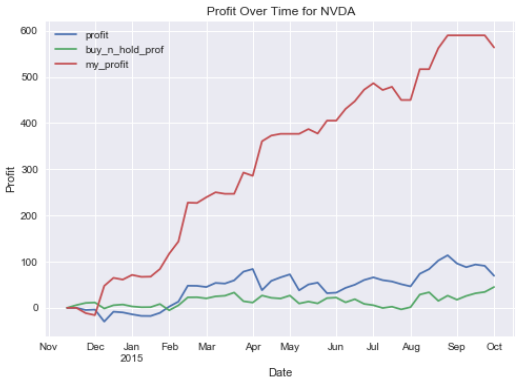

Figure 15: Profit Over Time for NVDA (2012-15 Interval)

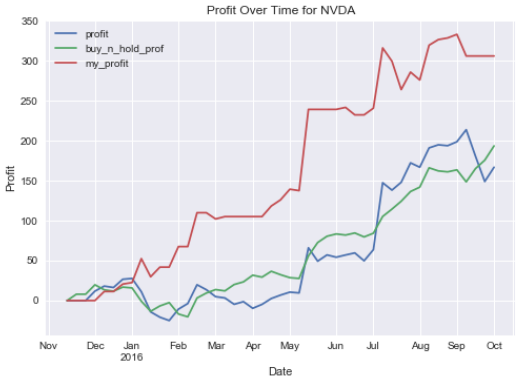

Figure 16: Profit Over Time for NVDA (2013-16 Interval)

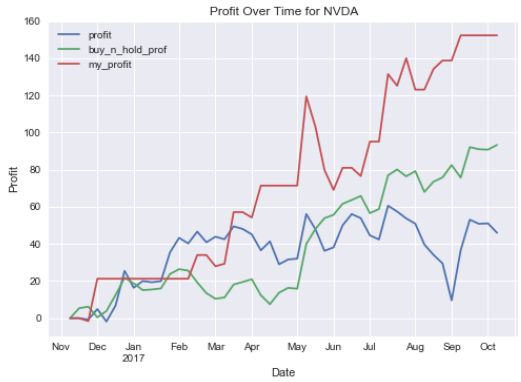

Figure 17: Profit Over Time for NVDA (2014-17 Interval)

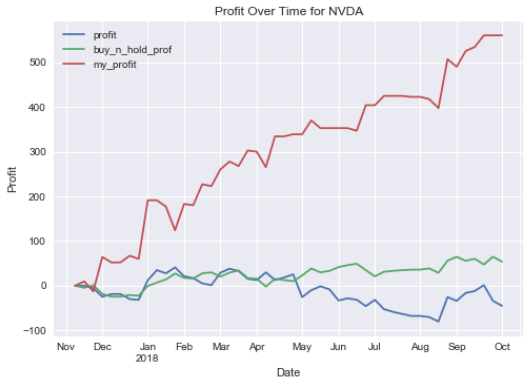

Figure 18: Profit Over Time for NVDA (2015-18 Interval)