Momentum Stock Picker

Caleb Tonkinson
Kyle Hale
Bryce Perkins
Benjamin Gardner

Brigham Young University

April 13, 2017

Abstract

Our paper details our experiment to find the predictive power of momentum (defined as consistent patterns in stock movement) on future stock returns. We examine a variety of machine learning models and test which are best at learning the proper weights for the input features and predicting whether a stock will move up or down. We frame the investor as someone who wants to buy at market open, hold for the entirety of the trading day, and sell at market close. He is trying to maximize his profits using this approach. We find that the best algorithm for this problem is the logistic classifier, but that feature selection has more of an effect on predictive power. We conclude that it is possible to predict the up and down trends of the market with a fair amount of accuracy, but it is impossible to know future profitability.

1 Introduction

1.1 Motivation

The US equity market is a complex system that facilitates the exchange of a large volume of stocks each day. A large camp of academics support an efficient-market theory. The theory states that stock prices already include all past and present information as well as some degree of future information about the underlying firm. However, recent academic research and empirical results have shown that there are often inefficiencies in the market. An inefficient market implies an incorrect price. If an investor were able to identify this mispricing, he could exploit it to make a profit.

We wish to identify mispricing related to momentum in individual stocks. Finding this pattern in the data would exploit the psychological phenomenon of crowd behavior. Essentially, we hypothesize that some investors are buying and selling stocks for a reason unrelated to the fundamental value of the company. Instead, investors examine the recent performance of the stock. If it has been going up, they buy. If it has been going down, they sell. The increased volume of trades for stocks with strong momentum leads to even more gains, a sort of self-fulfilling prophecy. The converse is true of stocks with poor momentum.

Another possible explanation for the performance of momentum stocks ties directly to the success or failure of the underlying company. A well-run company with a track record of past success will have a greater probability to succeed in the future. A mismanaged company will run into more headwinds and struggle to succeed. Depending on the rate at which these reported successes or failures are reflected in the price of the stock, identifying this trend early could allow for some profitability.

1.2 Task

Our task is to use a machine learning model to predict the future performance of a stock based on its trading information (price, volume of shares traded, opening price, etc.) and past performance (momentum). Due to the sheer size of the stock market, we have decided to narrow our experiment to a smaller class of stocks. We decided to use data on stocks in the technology industry: Google, Microsoft, Apple, and Amazon. We chose these stocks because of personal interest and familiarity with the companies. Even with such a narrow range of stocks, we will still have a large data set because we are using daily information. For a one year period there are approximately 250 data points available for any specific stock. Running the model

on four stocks over a two year period means we end up with more than 2,000 data points to use in the training and test sets.

2 Method

2.1 Features

To gather the data set we used a service called Quandl. Quandl provides a number of different free and paid APIs for the financial industry. The API that we used provided day by day information for an individual stock from the time it started on the market to the present day. The base feature set that it provided was the following: Date, Open, High, Low, Close, Volume, Ex-Dividend, Split Ratio, Adj. Open, Adj. High, Adj. Low, Adj. Close, Adj. Volume.

- Date - The date on which the trading information is reported
- Open - The price of the stock when the market opened
- High - The highest price the stock reached during the trading day
- Low - The lowest price the stock reached during the trading day
- Close - The price of the stock when the market closed
- Volume - The number of stocks traded during the day
- Ex-Dividend - The date used to determine ownership for dividend payments
- Adjusted versions of the variables are changed to correct for events that occur outside the trading day (e.g. corporate actions after market close)

However, our end goal was to allow for classification of a new instance of information, before most of this information could be known. This would in theory allow someone to run our algorithm in the morning to make investment decisions for that day. To that end, we eliminated features like, Closing price of the day, Return for the day, Total volume, and a number of others. That left us with fairly basic information about each day. Namely the opening price of the stock on that day. The rest of the usable features had to be calculated based on the other data.

We calculated several momentum characteristics of differing time periods, one for one week, one for two weeks, and one for 1 month. In order to calculate this variable, we took the average of the output class over the specified period. For example, if a stock had increased in four of the last five days its momentum would be 0.8. A stock with a momentum of 1 would have increased over every day in the measured period. A momentum of 0 corresponds to a constant decline. Our hope is that these created features will capture the effect of the momentum characteristic we identified. We calculated momentum values for different time periods in an attempt to identify which time period was most relevant.

In conjunction with the momentum variable we also calculated a momentum * open price attribute. We observed that the average up or down attributes were only measuring how many days the stock had moved a certain direction and failed to capture the magnitude of the change in price. We hypothesized that the magnitude of price movement would also factor into predicting future stock movement. Our list of used features and their descriptions is below.

1. Date - We hypothesized that the model could learn which dates (more recent vs more distant) to weight more heavily.
2. Open - The opening price of the stock
3. Day Before - The output class of the day before (up or down)
4. Momentum5 - The average output class of the last 5 days
5. Momentum10 - The average output class of the last 10 days
6. Momentum30 - The average output class of the last 30 days
7. Open * Momentum5 - The opening price * the average output class of the last 5 days
8. Direction - Output Class

It is feasible to have this feature set at the beginning of a trading day and therefore could be used for determining the direction of a stock in a real world situation. There are a variety of additional features that might be beneficial but due to the complexity in gathering them, we did not include them. They are: general market conditions and trends, a consumer sentiment score about the company, general industry conditions and trends, and so forth. Most of these features would have needed to be deduced or aggregated from separate services.

2.2 Data Organization

In order to access the daily stock data, we simply downloaded it using the QUANDL API. We then stored it as a python dataframe and in .csv files.

In our initial investigations of the stock trade, we found that while we had easy access to large amounts of historical data, training on the joined data of many different companies did not provide decent results (sub 50% prediction accuracy). We decided that a company's stock may react differently, positively, isolated from others; certain company characteristics might show in the stock trend and, being isolated, be easier to predict.

Using only the historical data of the stock in question meant we saw variance in the accuracy across different models. Certain models had better predictive power for different stocks. There was not one model that performed the best across all the stocks we considered. We believe that this demonstrates our hypothesis well. Focusing on predicting based off of the history of an individual stock can take into account variances that are consistent with the history of only that stock.

3 Results

3.1 Initial Results

TensorFlow Results
One of the methods we used for classifying data was a Tensorflow neural network built using Keras, a high level neural net API. We tried a number of different configurations, where we changed batch sizes, number of nodes, number of layers, bias weights per node, activation functions, randomization and more.

Company	Standard	2 Layers	Momentum
MSFT	52.1	52.2	52.3
GOOG	51.4	51.4	52.9
AAPL	53.0	53.6	52.3
AMZN	52.2	51.6	54.5

Displayed in the table are results from a standard neural net. The standard net had 1 hidden layer with 2x the number of input nodes. The table also contains the results from a net with two additional hidden layers and one with additional momentum features.

Because we are dealing with such a complex system, a financial market, we were not expecting wildly high percentages for accuracy. We supposed that a stock picker that could predict stock movement a little more than half the time had the potential to produce a profitable strategy. We hoped for an accuracy in the high 50's to low 60's, but we were not able to accomplish this with the neural network on the Tensorflow platform.

If we were to try and make money with the model at this level of accuracy, we run into major problems. First, we do not know that the correct picks would have returns that are higher than losses incurred from the incorrect picks. For example, picking 51% of the stock movement for Microsoft could net a profit of $10, while incorrectly picking 49% of the movement could lead to a $20 loss. The net result is negative because the model is silent on magnitude of movement. Second, buying and selling stocks on a daily basis would incur a significant cost from brokerage fees. These could easily wipe away any gain from correctly predicting a stock 51% of the time.

We attempted to improve our results by changing the classification model. We used Naive Bayes and Logistic classifiers in addition to the MLP. After running the stocks through these models we did not see a significant difference in accuracy from simply changing the model. Because we used the same features and training approach for our experiment, we did not see an improvement.

Initial Weka Results (outstanding results)

- Microsoft
  - Logistic - 52.42% correct
  - Naive Bayes - 53.1% correct
- Google
  - Logistic - 52.6% correct
  - Multilayer Perceptron - 52.9% correct
- Apple
  - Multilayer Perceptron - 50.94% correct
  - Logistic - 52.90% correct
- Amazon
  - Naive Bayes - 51.99% correct
  - Multilayer Perceptron - 51.35% correct

3.2 Training

For the training stage of our experiment we used a variety of time periods in order to find the optimal window to measure momentum. These windows ranged from the entire history of the stock (reported at one day intervals) to a month

of data. For Apple, using the entire trading history meant we were training on more than 6,000 data points.

Although it is generally a good idea to add more data to reduce overfit, our testing approach lent itself to shorter windows, particularly ones closer to the present day (more on that in the next section). Depending on the model, these windows of time would be sent through a different training algorithm, but all of this was taken care of under the hood by Weka.

Overall, for our runs we used the Weka default, cross validation. Aside from understanding that our training set is being divided, we have little insight on the usefulness of this approach. When we programmed our own models, we had control over the stopping criteria, which helps us understand when we have achieved good generalization and should stop training. Although we are not guaranteed better test results, we should be more confident about their precision.

3.3 Testing

We chose the test set to be a smaller subset of the trading history, generally 15 days to two months from the present day. This approach meant that training sets that were confined to more recent windows led to better test accuracy. This follows logically from the idea that price movements in the present are better predicted by prices in the near past. In order to test accuracy, we simply compared the price movement and measured whether it had gone up or down.

Alternatively, we could have created chunks of contiguous time with test sets preceded by blocks of training data. This approach maintains the locality of the training and test data and allows for longer periods of time.

3.4 Changes to the Model

We began to adjust which momentums were used and scaled down the datasets so that we only used recent data; we scaled the training set to two years of daily data and only a month for the test set. We also combined methods using the Vote option in Weka, that allows for multiple algorithms to vote for the final prediction. This Ensemble model used the ZeroR, Naive Bayes, and Logistic classifiers to produce one output.

Weka Changes

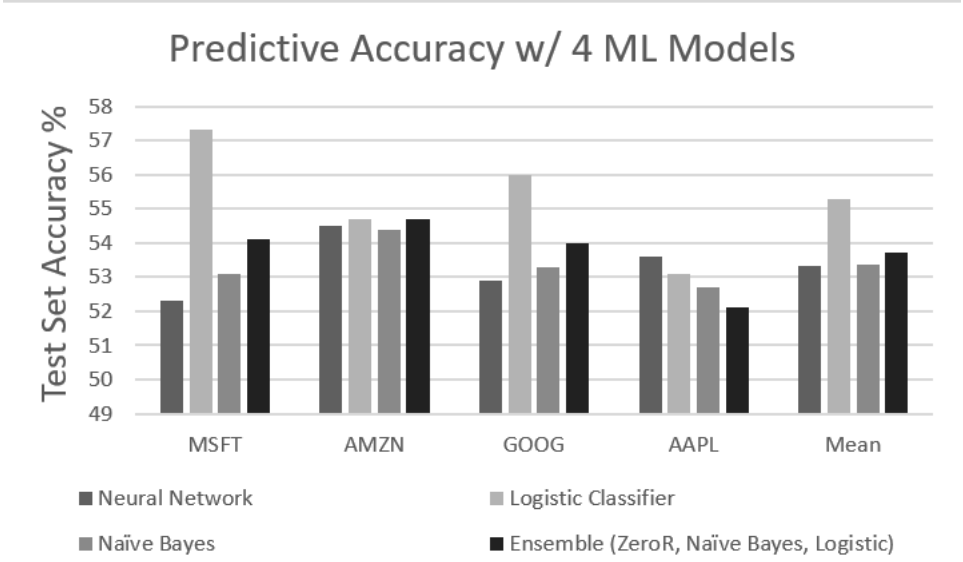
- Apple(2 years of data):
  1. Logistic - 53.15% correct
  2. Naive Bayes - 52.74% correct
  3. Vote (ZeroR, Naive Bayes, Logistic) - 52.17% correct
- Amazon(2 years of data):
  1. ZeroR - 54.43% correct
  2. Vote (ZeroR, NaiveBayes, Logistic) - 54.71% correct

3.5 Final Results

The following are the best results we achieved using combinations of different models and different momentum attributes.

Weka Results

- Google
  - 56.0% Accuracy
  - Using 5, 10, and 30-day momentum terms
  - Logistic
- Microsoft
  - 57.34% Accuracy
  - Using 5, 10, and 30-day momentum terms
  - Using recent data (2 years)
- Apple
  - 53.1% Accuracy
  - Using 5 and 10-day momentum terms
  - Logistic
- Amazon
  - 54.7% Accuracy
  - Using 5 and 10-day momentum terms
  - Using recent data (2 years)
  - Vote (ZeroR, NaiveBayes, Logistic)



3.6 Discussion

Our results show that the best model varied from stock to stock. Overall, the Logistic classifier performed the best on average with an accuracy of 55.3%. This makes sense in context because a logistic model does well for binary output classes (up/down). However, from our experiments, we observed that attribute selection and feature paring were more impactful than simply changing the model and throwing the same data at it.

The attributes that contributed the most to higher accuracy were the five and 10 day momentum values, as well as the momentum * open price, which encapsulated magnitude into the momentum. If we used an even shorter window for the training set, a month of past day trades, and the 15 subsequent days for the test set, we hit 60% training. However, because these are such short periods with fewer data points, they are much more susceptible to noise. As such, we chose not to report them as our final results.

4 Conclusion

One design flaw that might throw our results into question (there is no way we trust them enough to actually invest money in this strategy) is the narrowness of the stock selection. The results we discovered may be true only for technology stocks, and large ones at that. For future exploration, we want to test how this momentum pattern manifests itself in other industries and for a variety of stocks.

As previously mentioned, it was difficult to acquire accurate results when the stocks were lumped into the same data set. It would be advantageous to find a model that could learn this problem so stock predictions could generalize to entire industries and maybe even markets. This might lead to increased accuracy in predicting new instances of stocks within that industry.

The results from our stages of testing show that while stock prediction is not an easily performed, increases in prediction accuracy can be achieved through removal of noise (isolation of companies), feature optimization, and by finding the algorithm that can best match the trends of a company's stock. Further, we found that feature optimization weighed more heavily on predictive accuracy than algorithm selection.

For future testing, we hope to incorporate features that reflect the current circumstances of the company or the market (i.e. an acquisition was announced or a public event caused distress to passengers aboard a plane). These types of features may be difficult to predict, but would be valuable to an investor as they would be able to view the effects of certain types of situations. We also hope to include a prediction for the amount of change occurring for a stock. This would prove valuable to investors as they would determine if the cost of transaction would be covered by the change in amount.