Strategy Evaluation

Charles Wesley Warbington
cwarbington7@gatech.edu

Abstract—This paper analyzes various trading strategies for the stock market. Multiple implementations were built around technical indicators including Bollinger bands, relative strength index and exponential moving average. Parameters were adjusted manually and by a machine learning model to enter long or short positions in order to maximize portfolio value.

1 INTRODUCTION

The goal of this project was to find strategies that increased a portfolio value over time utilizing only technical indicators and investing only in US stocks. To do this, historical stock data was split into training and testing buckets. The training data was used to tune parameters and the test data was used to measure the accuracy of the model's predictions and iterate on this process. Two separate strategies will be analyzed in this paper, but both utilize the same technical indicators. The strategies differ in how the parameters are tuned. The first strategy which we will call "manual strategy" was tuned manually by the author based off observation of how certain changes increased or decreased performance. The second strategy, which we will call "classification strategy learner" uses a decision tree that decides when a long or short position should be entered based off the three technical indicators.

The constraints of the problem allow for a starting cash position of $100,000. For simplicity, only one stock may be traded per run of the models. The only legal positions are to be 1000 shares long, 1000 shares short, or all in cash.

2 INDICATOR OVERVIEW

As stated above, three technical indicators were used as the basis of how trading decisions are made. The three indicators are Bollinger bands percentage, relative strength index, and an exponential moving average cross. Both the manual strategy and the decision tree call the same code that returns a single vector data frame indexed on date for each of the respective indicators.

The various indicators have lookback periods which require data from a certain past window. Because of this, there will be a delay corresponding to the lookback period, in which the indicators first values occur. To make sure that we have indicator values on the first day of the training data, the code looks back farther into the past to the point where the start date has values for all indicators.

2.1 Bollinger Bands Percentage

A lookback period of 20 days was used when calculating the simple moving average and rolling standard deviation, which go into the calculation of the Bollinger bands percentage. The top and bottom bands are found by adding or subtracting two times the rolling standard deviation to the simple moving average. Lastly, we need to reduce this information down into an actionable single vector. The Bollinger bands percentage is thus found by the following formula: (price – bottom band) / (top band – bottom band).

2.2 Relative Strength Index

The relative strength index is the ratio of positive price changes to negative price changes. This is calculated as a moving average, so again a lookback period is defined. A lookback period of twenty days was used, so the average upward price movements for twenty days are calculated, then the average downward price movements for 20 days are calculated, and lastly the ratio of those two values is tabulated.

2.3 Exponential Moving Average Cross

The exponential moving average is very similar to the simple moving average. The lookback period is also used for EMA, and two different ones (13 and 21) are used. The EMA uses a multiplier that puts more emphasis on recent days. So, yesterday's price has more effect on the average price than ten days ago does. Two different lookback periods are used because short versus long term average prices crossing has a relationship with price trajectory. To combine these two moving averages into one vector, the ema cross is simply the ema with a lookback of 13 days divided by the ema with a lookback period of 21 days.

3 MANUAL STRATEGY

The manual strategy takes into account all three indicators described above each day to determine if it should enter a long position, a short position or do nothing. A signal count is first initialized. If a signal count of 2 is calculated at the end, then the model says to enter a long position. Similarly, if the signal count is negative 2, then the model says to enter a short position. Nothing is done if neither of these conditions is met.

For Bollinger bands, if the value is less than 0.13, then one is added to the signal count and if the value is greater than 0.75, then one is subtracted from the signal count. For rsi, if the value is less than 30, then one is added to the signal count and if the value is greater than 50, then one is subtracted from the signal count. Lastly, for ema cross, if the value is less than or equal to 0.967, then one is added to the signal count and if the value is greater than or equal to 0.985, then one is subtracted from the signal count.

To arrive at these values, I looked at multiple values for one single indicator and found the optimal values that increased the portfolio just using strictly one indicator at a time. Then I did some minor tweaking based on how the indicators interacted together. There must be at least two points (signal count) for a signal to trigger. This means that if Bollinger bands and rsi signal a buy, but the ema cross signals a sell, then nothing will happen. This is a distinct difference from just two indicators agreeing one way. There must be majority agreeance without the opposite signal from the other indicator.

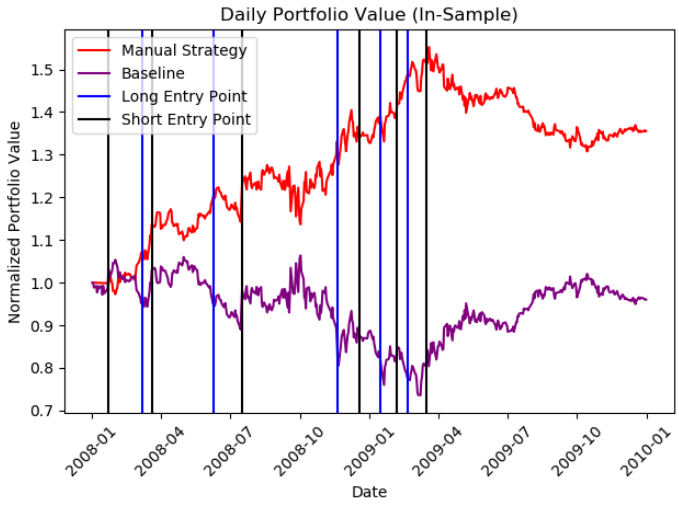


Figure 1 — Daily Portfolio Value (In-Sample)

Above in figure one, two lines are plotted representing two distinct portfolios. Each portfolio has a starting cash balance of $100,000 and is over the time period of January 1, 2008, to December 31, 2009. The stock that is traded in this instance is "JPM". The purple line is a benchmark and is the result of purchasing 1000 shares on day one and holding. The red line illustrates a portfolio when using the manual strategy. The black vertical lines represent entering a short position and the blue vertical lines represent entering a long position. The portfolio is normalized thus both start at 1. This is for the in-sample or training data so we should expect to perform well as the model was tuned on this very data. The manual strategy performs quite well over the benchmark resulting in close to a 35 percent return during the time period.

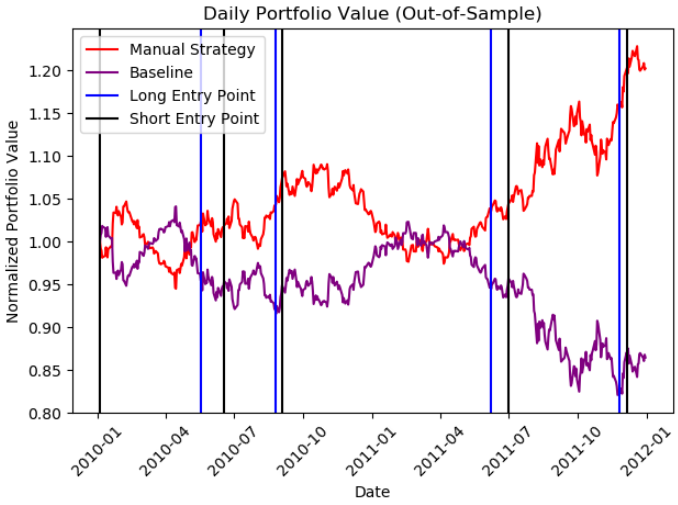


Figure 2 — Daily Portfolio Value (Out-of-Sample)

The next important thing to do would be to see how the model performs on data it has not seen yet. Figure 2 above illustrates the out of sample portfolio values. All the constraints from figure 1 stay the same except for the time period. In this case, the period is from January 1, 2010, to December 31, 2011. As the red line reflects, the manual strategy yet again outperforms the benchmark. The difference though is that the portfolio this time resulted only in a 20 percent increase opposed to the 35 percent increase from the in-sample portfolio. While this is a decrease, it is understandable as the model should perform better on data it has already seen. It is important to note that the model was not tweaked by looking at this test data. The manual strategy has generalized quite well (not overfitting) for this dataset and results in good performance on data it has not

seen. Below, table 1 summarizes statistics of the four portfolios. In both cases the manual strategy outperforms the benchmark. The in-sample manual strategy portfolio increased by 35.5% and the out-of-sample manual strategy portfolio increased by 20.2%.

	Cumulative Return	Std Daily Returns	Mean Daily Returns
Benchmark In-Sample	-0.039914	0.017468	0.000071
Benchmark Out-of-Sample	-0.135365	0.008782	-0.000251
Manual Strategy In-Sample	0.355038	0.012346	0.000679
Manual Strategy Out-of-Sample	0.202176	0.007606	0.000395

Table 1 — Manual Strategy & Benchmark Statistics

4 STRATEGY LEARNER

The second strategy is a classification learner. A random decision tree was used as the model. To make sure the decision tree was a classification tree and not regression learner, mode opposed to mean was used when building the tree. For example, if a leaf size of 5 is used, and three of the remaining y values are 1 and two are -1, then 1 will be chosen as the prediction. A bag learner was also used to converge to a more stable predictions that do not overfit. A leaf size of 10 was used and 30 bags were used.

The random decision tree learner or RTLearner needs x data and y data to train on. The three technical indicators defined above are the x data, but the y data had to be calculated. The y data needs to be buy or sell signals. Since this is past data, we know the future price values, so we can create buy or sell signals based off returns over a certain period. The period used is five days. So, we are looking at a comparison of a price today to the price five days in advance. Once this value is calculated, we need to decide on a threshold that needs to be set to determine if we should buy or sell. If the value is less than or equal to -0.03 minus "impact" then the y value would be a sell or -1. If the value is greater than or equal to 0.03 "impact", then the y value would be a buy signal or 1. One thing to note is that we cannot peek into the test data, so the last five days cannot be used because we are calculating the return based off five days in the future so the last five days

will not have values. As a result, we just throw away those last five days as they have no associated y values to train on.

Each time, the classification strategy learner is instantiated and trained with the add_evidence method. This method is a bag learner of RTLearner's built with x and y data as described above and then stored in the StrategyLearner class, to be queried on with new data as much as needed. When queried, the learner receives a set of indicators and based off those, predicts whether it should enter a long position, enter a short position, or do nothing. Sometimes, one would want to normalize or discretize the x data or feature values as large numbers could potentially be given more weight than smaller numbers. This was not an issue in this implementation because the decision tree is choosing a feature to split on randomly and does not care about the values of the features.

5 EXPERIMENT 1

Experiment 1 compares the portfolio values of the benchmark case (buy and hold 1000 shares), the manual strategy and the classification strategy learner.

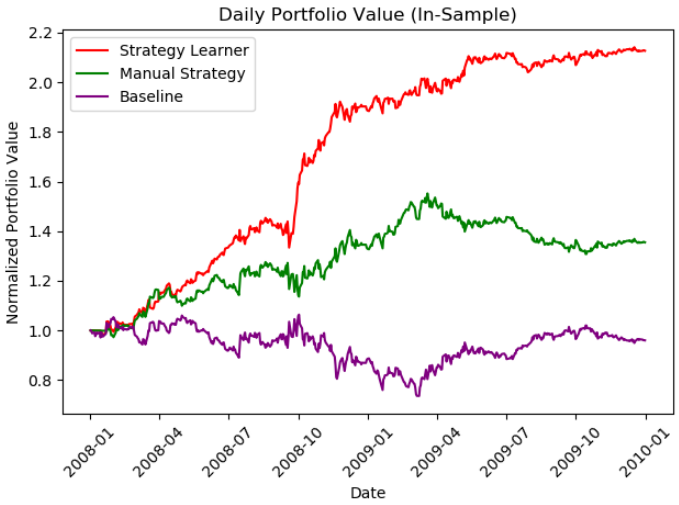


Figure 3—Daily Portfolio Value (In-Sample)

First, the in-sample performance was assessed. The stock "JPM" was again used with $100,000 starting cash and period from January 1, 2008, to December 31, 2009. As illustrated above in figure 3, the classification strategy learner (red) performs the best resulting in close to 120% increase in portfolio value. The manual strategy (green) performs the next best around a 40% increase. The benchmark (purple) hovers around the starting value. This is incredible

performance from the classification learner, and it clearly found successful insights on how to trade based off the three indicators. This makes sense that the classification strategy learner would perform the best, (if it was correctly implemented) because it should be able to more finely tune the parameters opposed to me as a human guessing and checking trying to figure out what parameters work best together. If the experiment were repeated multiple times, I would guess that this would be the general trend. The manual strategy and benchmark will stay the same while there will be some variability in the performance of the classification strategy learner because of the randomness built in to building the tree. The classification strategy learner will still perform better though despite the randomness because of the fine-grained tuning that it can do – finding the exact thresholds to make decisions on.

The next thing to look at is how the different strategies perform on data they have not seen (test data). Below, figure 4 shows the portfolio values when using

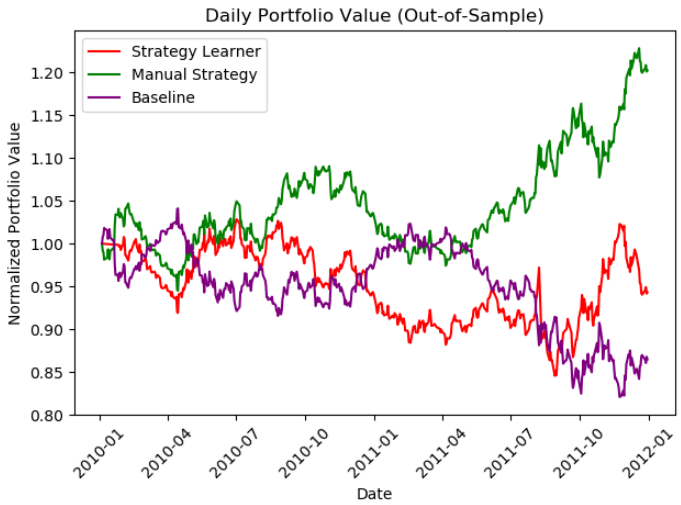


Figure 4 — Daily Portfolio Value (Out-of-Sample)

the separate strategies on the out-of-sample data. One downfall of finely tuning and high performance on the training data is that the model becomes biased or overfits. It seems that in this case the classification learner has overfit to the in-sample data to some extent. We can see in figure 4 that now the manual strategy performs the best while the classification strategy learner comes in second still beating the benchmark. It looks like the patterns of the stock price during this date range were quite different than the in-sample time frame, so the classification strategy learner was not robust enough to know how to optimally

trade on this new data. Nevertheless, it still does not perform poorly and seems to have some insights as it beats the benchmark of buy and hold.

6 EXPERIMENT 2

Impact is the effect that entering into a trade has on the price of the stock. For example, if one buys a stock, theoretically this can cause the price of the stock to increase, thus making the subsequent purchasing of shares slightly more expensive. The situation is reversed when selling. For simplicities sake, impact was calculated by multiplying the impact value by the price and number of shares. This is then subtracted from cash holdings when a trade is executed. The higher impact is, the more money one loses out on when executing a trade. The ideal value would be zero as this means that your personal trading has no effect on the price. Nevertheless, impact does exist at a non-zero value and a trading strategy should take this into account. In this experiment, we look at two separate metrics to assess the effect of impact on the classification strategy learner.

First, we will simply look at how varying the impact effects the overall portfolio value. Below in figure 5, three different portfolios are plotted over the period

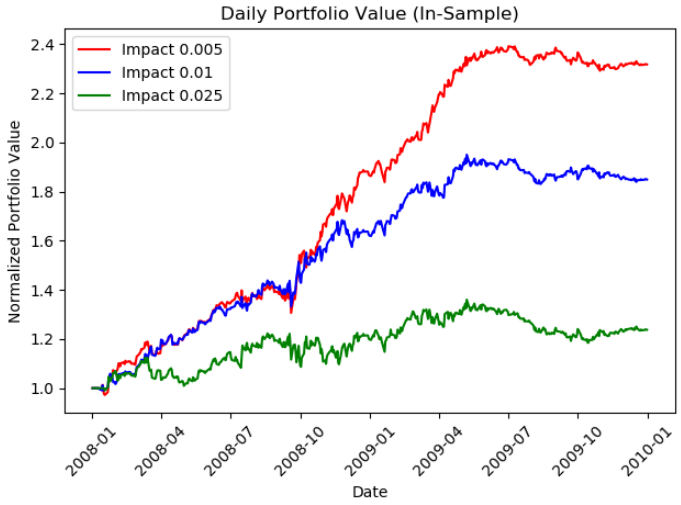
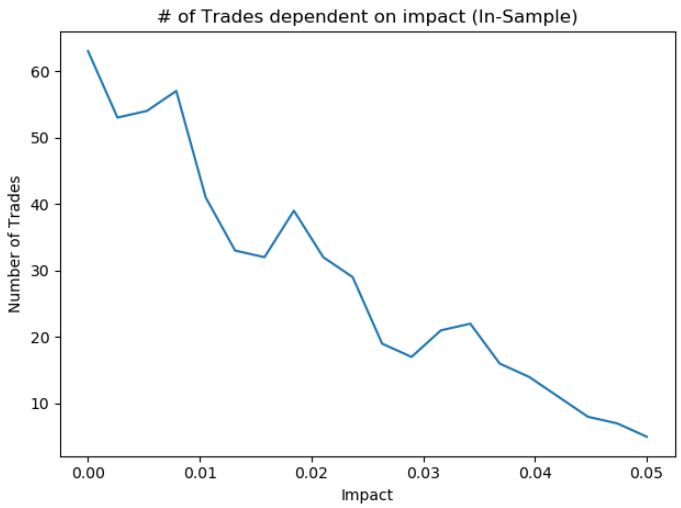


Figure 5—Daily Portfolio Value (In-Sample)

from January 1, 2008, to December 31, 2009. Again, the starting cash value is $100,000 and the stock traded is "JPM". The values of 0.0005, 0.01, and 0.25 were chosen as values for impact. In that same order (increasing) the values of the

portfolio decreases. The relationship that appears is that as impact increases, performance of the model (reflected in portfolio value) decreases. Figure 6 below



provides some insight on why this could be the case.

Figure 6— Number of trades dependent on impact

Figure 6 shows a downward trend in number of trades made as impact increases. This makes sense because as the cost of executing a trade increases, you would only want to execute the most profitable trades. This seems to be an answer to why the portfolio value decreases as impact increases. The model is not simply executing all the same trades and just losing out on the increasing impact fee. It is choosing to execute less trades as impact increases in order to maximize portfolio value. When the value of impact is very small, one can profit on many trades that may not result in a large increase in portfolio singularly, but all together add up to large portfolio growth. When the impact is high though, it is much more important to choose wisely when one gets in and out of positions.