CS7646 Project 8: Strategy Evaluation

Thomas Kim
tkim338@gatech.edu

1 INDICATOR OVERVIEW

The indicators used in the Manual Strategy and Strategy Learner were simple moving average (using two different window sizes), Bollinger bands, and momentum.

SMA was used with 20 day and 50 day sliding windows. Buy signals are generated when the 20-day SMA is higher than the 50-day SMA. In the Manual Strategy, the 20-day SMA must be 0.5 (normalized) higher than the 50-day SMA for a buy signal to be generated and the 20-day SMA must be 0.5 lower than the 50-day SMA for a sell signal to be generated. The Strategy Learner does not have a hard-coded threshold, but instead uses the difference between the two SMA values.

Bollinger bands were used in the Manual Strategy to identify points where the current price crossed the upper or lower Bollinger band. A Bollinger band percentage was calculated using the formula: $BB\% = (price - BB_{lower}) / (BB_{upper} - BB_{lower})$. Optimization was performed to find the limits of BB% to generate effective buy/sell signals for the Manual Strategy. The limits of -0.01 and 1.01 were found to be effective and were used to generate buy and sell signals respectively. In the Strategy Learner, BB% was used as a parameter as part of the computed state used in the Q-Learning algorithm.

Momentum was used in the Manual Strategy learner to generate buy signals when momentum was positive and sell signals when momentum was negative. The exact values of momentum on which to generate signals were optimized and values of 2.5 and -2.5 were found to generate effective buy and sell signals respectively. In the Strategy Learner, the value of momentum itself was used as a part of the computed state.

2 MANUAL STRATEGY

The Manual Strategy was developed from three of the five indicators previously developed in a past project (simple moving average, Bollinger bands, momentum, volatility, on-balance volume). Volatility was eliminated as a possible contributor for the Manual Strategy, as plots of volatility against price seemed to show volatility tracking the rate of change of stock price but did not seem to track the direction of price change. On-balance volume was also eliminated as a possible contributor as it seemed to track price closely, but lagged behind, making it less useful as a future price indicator.

The remaining three indicators (SMA, Bollinger bands, and momentum) were used, as they all seemed to provide possible leading hints to future price changes.

SMA was used with two different sliding windows, 20-day and 50-day. Buy and sell signals were generated when these two averages were far apart. When the 20-day SMA is higher than the 50-day SMA, it's inferred that the stock price has increased recently (within the last 20 days) and may continue to increase, which means that a buy signal here may be profitable. In the opposite case, when the 20-day SMA is less than the 50-day SMA, a stock price may be on its way down and a sell signal may be prudent.

Bollinger bands were used to identify points when the stock price was near a low point or near a high point. Bollinger band percentage was computed as the relative ratio of stock price within the upper and lower Bollinger bands. When the stock price approached the upper Bollinger band, it was understood that the stock price was unusually high and could be expected to fall back down. The inverse is true for times when the stock price was near the lower Bollinger band and was expected to rise back up. Buy signals were thus generated when the Bollinger band percentage reached a certain lower threshold (0.01 in the Manual Strategy) and sell signals were generated when the Bollinger band percentage reached a certain upper threshold (1.01 in the Manual Strategy).

Momentum was used as a way to track the relative rate and direction of stock price changes. A large, positive momentum indicated that stock price was rising quickly and a large (in magnitude), negative momentum indicated that stock price was falling quickly. Buy and sell signals were thus generated when

momentum was high (2.5 was used in the Manual Strategy) and when momentum was low (-2.5), respectively.

These signals were combined to form the entire Manual Strategy. For every query, each signal returned a 0, +1, or -1. The three signals were summed and a net positive sum resulted in a buy signal and a net negative sum resulted in a sell signal, while a sum of 0 resulted in no signal (hold position). This algorithm was tested and optimized iteratively against the in-sample dataset until an acceptably high performance was reached. The only parameters optimized where the thresholds at which signals were generated using BB% and momentum. The signals generated (buy when momentum is positive or when BB% is low) align with currently accepted market theory and common strategies, so these results seem to be logical. The performance of the Manual Strategy using in-sample data is shown below against the benchmark of JPM.

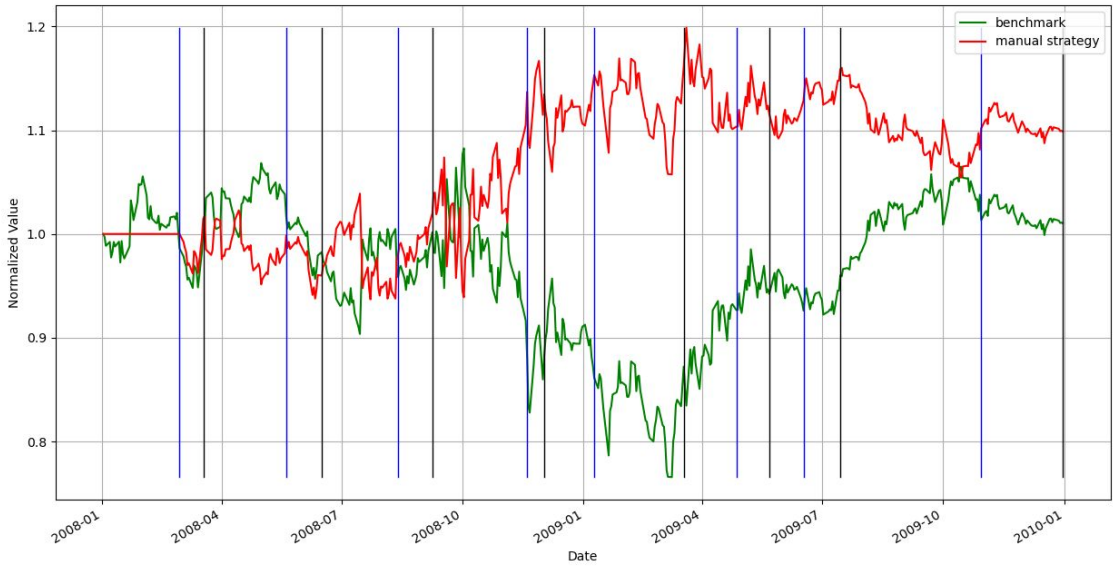

Figure 1—In-sample testing of Manual Strategy.

Below is a figure illustrating the performance of the same Manual Strategy using out-of-sample data against the same benchmark of JPM (for a different time period).

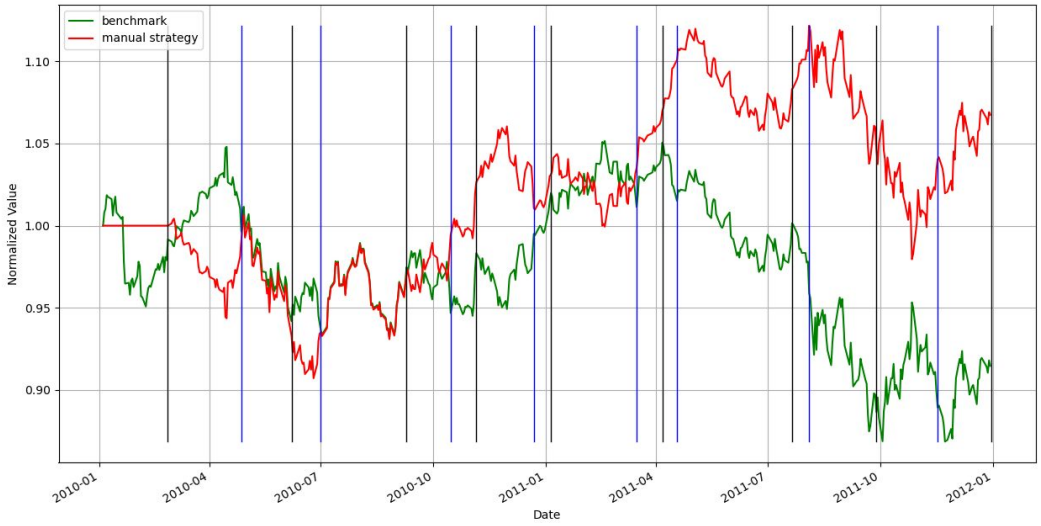

Figure 2—Out-of-sample testing of Manual Strategy.

The performance of the manual strategy is compared between the two datasets (in-sample and out-of-sample) in the figure below. The values shown on the y-axis have been normalized relative to the benchmark value for a more direct comparison.

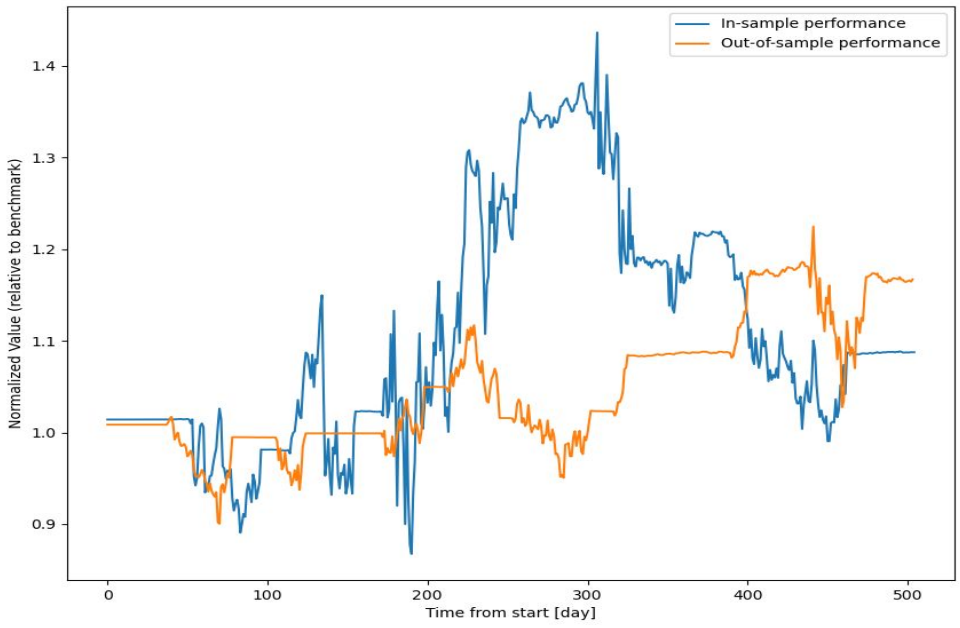

Figure 3—Comparison of Manual Strategy performance between in-sample and out-of-sample data.

Numerical performance metrics of the Manual Strategy are summarized in the table below for in-sample and out-of-sample data.

Table 1—Performance summary of benchmark stock and Manual Strategy, in- and out-of-sample.

	In-sample		**Out-of-sample**	
	Benchmark	**Manual Strategy**	**Benchmark**	**Manual Strategy**
Cum. return	1.02%	9.87%	-0.0853%	6.746%
Mean daily return	-0.0124%	0.00843%	-0.0213%	0.0109%
Std. dev. of daily return	0.0169	0.0154	0.00849	0.00783

As shown in the above figure and table, performance of the Manual Strategy is higher for the in-sample time period than the out-of-sample time period. This is reflective of the method by which this rule-based strategy was developed. It was developed with knowledge and access to the in-sample data and optional parameters in indicators were coarsely optimized to maximize cumulative return for the in-sample period over multiple iterations of testing and refinement. Of course, this leads to overfitting of the model to the in-sample data, which improves in-sample performance at the cost of reduced out-of-sample performance.

It should be noted that the out-of-sample performance of the Manual Strategy is still greater than the benchmark. This could be an indication that the Manual Strategy is not yet fully optimized (overfitted) for the in-sample data or that this particular set of rules happens to perform well for this particular time period. It should also be noted that the out-of-sample performance of the Manual Strategy normalized against the benchmark return is also higher than the normalized in-sample performance of the same strategy. This indicates that it's likely that the Manual Strategy can be further optimized for the in-sample data. However, the benchmark return for the out-of-sample data is also less than the benchmark

return for the in-sample data, so this accounts for at least some of the difference in performance between in-sample and out-of-sample tests.

3 STRATEGY LEARNER

The Strategy Learner was built on top of a base made up of the Manual Learner and Q-Learning Robot. Excluding hyperparameters and other set-up variables, Q-learning models require only a few inputs: state, reward, and number of actions.

The number of actions is set to be 3, allowing the learner to buy, sell, or hold a stock position (don't make any trades).

Reward was straightforward to compute. It was computed as the change in portfolio value since the last action was taken, which is equal to the portfolio position (-1000 or +1000) multiplied by the change in stock price. If a trade was made, reward was then reduced by the values of commission and impact.

Computing the state required more work. As the Q-Learner Robot was developed to take state as a single integer, the current state, or snapshot, of the stock price had to be reduced to a single integer as well. This was accomplished by binning, or discretizing, the indicator values returned from a given stock price history. 5 bins were used for each indicator, resulting in a total of 125 (5 bins ^ 3 indicators) states.

The indicator for computing SMA was discretized simply, using 5 bins with equally spaced intervals between them, using pandas.cut().

The indicator developed using Bollinger bands was discretized by first computing Bollinger band percentage to combine upper Bollinger band, lower Bollinger band, and stock price into a single series. This series was then fed through pandas.cut() that separated the data into equally-spaced bins. It was suggested to use pandas.qcut(), as this function separates data into bins of equal sizes, however, this masks outliers in the data, which seem to provide the best indicators of stock price changes.

Momentum is already computed as a single series, so only binning was required for this indicator. Again, pandas.cut() was used as outliers in momentum seemed to be the most effective markers for future stock price changes.

Hyperparameters for the Q-learning model were adjusted throughout the development process, by increasing learning rate and decreasing exploration to try to produce a more consistent learner or doing the opposite to try to produce a faster-learning model. A relatively high learning rate of 0.5 was used to speed up convergence. A low gamma value of 0.3 was used to emphasize immediate and short-term rewards for the model. A random action rate of 0.8 and a random action decay rate of 0.99 were used to encourage high exploration of actions.

Dyna was not used in this model, as the actions of the learner did not affect the state of the model (calculated from stock price), so it did not make sense for a transition matrix to attempt to predict future states based on state-action pairs. However, since the training data is relatively sparse, the training data was fed to the model multiple times to ensure that it converged (40 epochs were used in this case).

4 EXPERIMENT 1 (MANUAL STRATEGY / STRATEGY LEARNER)

This experiment seeks to show the differences in performance between the Manual Strategy and the Strategy Learner. Assumptions made were that each model learns from only in-sample data, stock position can only be +1000 or -1000, and leverage is uncapped.

It's initially hypothesized that the Strategy Learner will outperform the Manual Learner because the Strategy Learner iteratively optimizes on the same parameters that are manual optimized in the Manual Strategy along with exploring other combinations of indicators that may not be intuitive and thus overlooked during the development of the Manual Learner.

The results shown in the figure below support this hypothesis, as the Strategy Learner outperforms the Manual Strategy by a significant margin. This result is expected in nearly all cases when using in-sample data, because the Q-learning model used in the Strategy Learner here uses the same basic learning method (trial-and-error and iterative optimization) that is used to develop the Manual Strategy. This means that in most cases, the best Manual Strategy can only match the Strategy Learner and not exceed its performance. There is a small probability

(dependent on hyperparameters) for the Q-learner to diverge from a solution and perform worse than a Manual Strategy, but this probability can be reduced to a negligible amount.

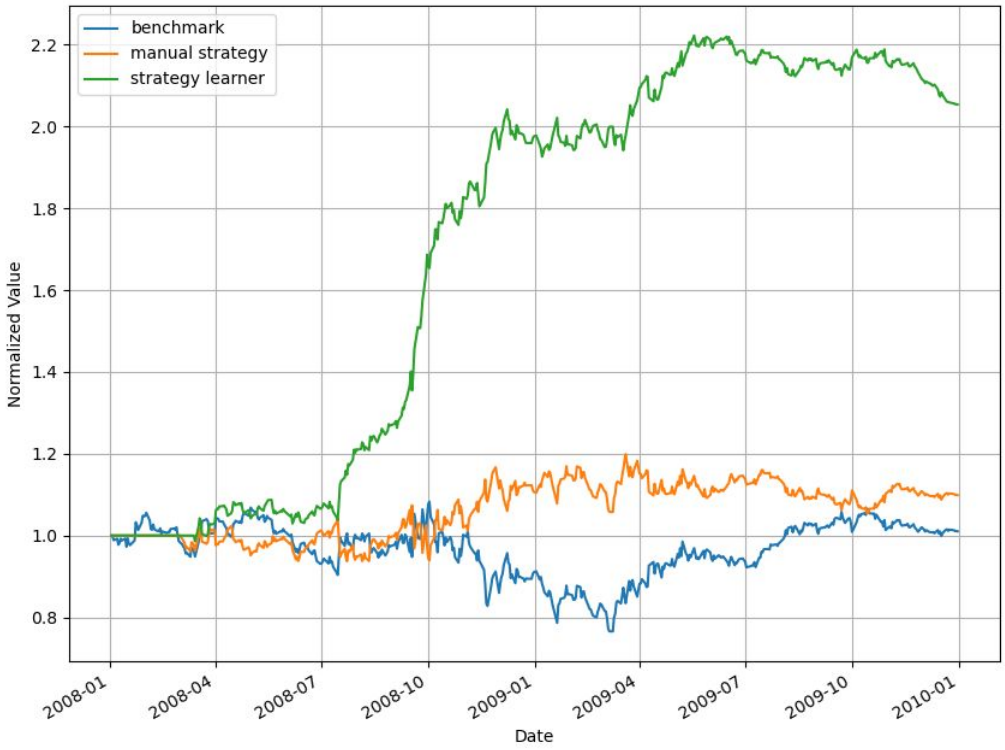

Figure 4—Strategy Learner and Manual Strategy performance against benchmark.

The hyperparameters used in the Q-learner of the Strategy Learner are listed in the table below.

Table 2—Hyperparameters used in the Q-learner of the Strategy Learner.

Hyperparameter	**Value**
Learning rate, α	0.5
Discount factor, γ	0.3
Random action rate, rar	0.8
Random action decay rate, radr	0.99

dyna	0

5 EXPERIMENT 2 (STRATEGY LEARNER)

This experiment seeks to illustrate the effect of varying impact on the performance of the Strategy Learner. The different values for impact used were 0, 0.005, 0.01, and 0.1. It's hypothesized that increasing impact will cause the learner to make fewer trades, as increasing impact reduces the reward for actions that result in a trade being made. Overall performance should also be reduced, as increasing impact also increases the cost of trading, eating into return.

As illustrated in the figure below, increasing impact decreases learner performance. A very high impact of 0.1 causes the learner to perform much worse than the benchmark itself. The number of trades made by each learner were also affected, but not in a clear way. The learner with the highest impact made the most trades, while the number of trades made by the other three learners do not directly correlate to each other. However, the differences in impact are not large compared to the daily changes in stock price. It's possible that results generated here require many more runs to eliminate enough noise to see how impact values affect number trades.

In running this experiment, the same hyperparameters were used in the Q-learner as were used in the previous experiment.

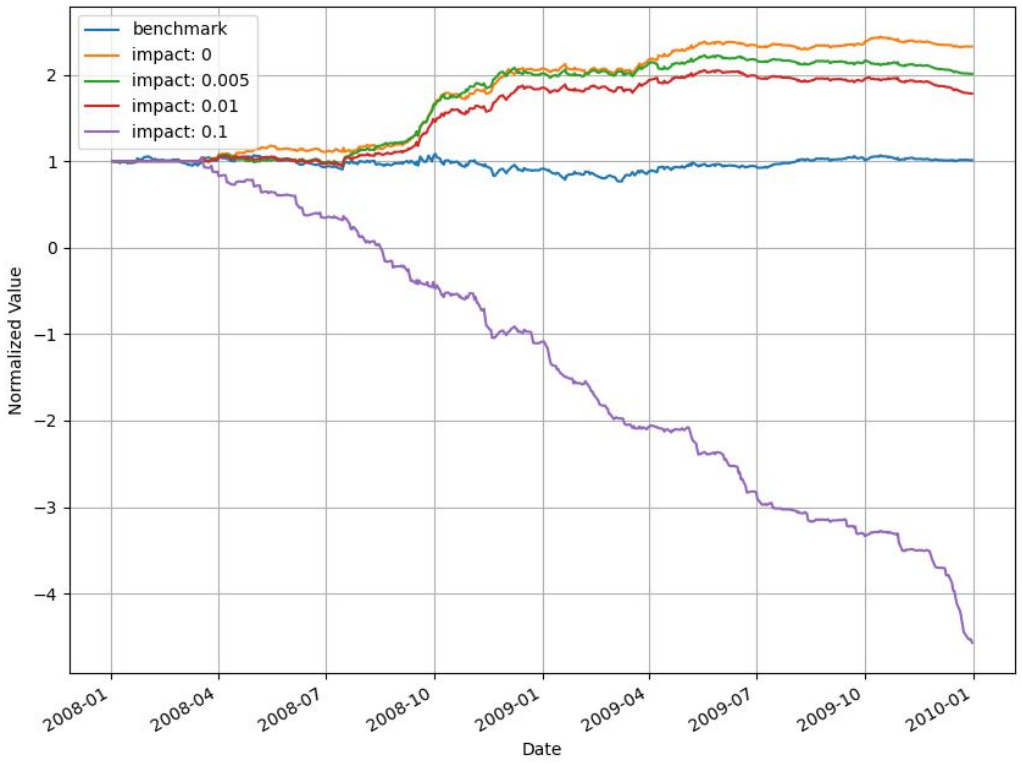

Figure 5—Effect of impact on Strategy Learner performance.

Table 3—Learner behavior as influenced by varying impact value.

	Cumulative return				**Number of trades**			
Impact	**Run #1**	**Run #2**	**Run #3**	**Mean**	**Run #1**	**Run #2**	**Run #3**	**Mean**
0	91.9%	137%	132%	120%	98	82	98	92.7
0.005	105%	106%	101%	104%	70	96	90	85.3
0.01	91.2%	45.7%	55.6%	64.2%	84	86	100	90
0.1	-650%	-574%	-506%	-577%	114	100	96	103.3