Project 8 Report: Strategy Evaluation

Youssef Sultan
ysultan@gatech.edu

Abstract—In this report, two stock portfolio investment strategies are developed and comparatively reviewed. To ensure congruence in this empirical study, both strategies utilize the same technical indicators, starting cash, starting shares and stock. The difference between the two strategies is that one's trading rules are manually set, while the other's are set by artificial intelligence. Throughout the meta-analysis, the in-sample period is January 1, 2008 to December 31, 2009 and the out-sample period is January 1, 2010 to December 31, 2011. All experiments are conducted on the JPM stock. The benchmark is defined as holding 1000 shares at the start date of each period.

Introduction

The first strategy denoted as the "Manual Strategy" is a manual rule-based trader that makes trades or actions based on stock prices from the in-sample period. The stock prices in this period are then used to aggregate three technical indicators and based on these technical indicators' values, decisions to enter or exit positions in the stock are executed. The second strategy denoted as the "Strategy Learner" creates its own rules based on the data it learns during the in-sample period using the random forest classifier algorithm. In an effort to construct an uncorrelated forest of decision trees whose collective predictions are more accurate than any manual rule or single decision tree, it builds each tree using bootstrap aggregation and feature randomness. It is postulated that due to the adaptivity of artificial intelligence the Strategy Learner's ability to reduce variance in this problem statement will yield a higher portfolio value than the Manual Strategy.

Indicators

Momentum is based on price differences within a fixed time interval, it measures the volatility of stock prices in trend analysis and shows the strengths and weaknesses. This indicator is implemented by aggregating the rate of change within the input price data and then subtracting the price data from the rate of change. This indicator will return a value that may cross above or below zero. Since the rate of change is based on the change of an $n - day$ window, in this case, the parameter $n = 20$.

Simple Moving Average (SMA) is based on the average of prices of a stock within a fixed time window. In this use case, the indicator presented is the stock price divided by the SMA which gives a ratio that can indicate when to buy or sell based on the zero line. The Price divided by SMA is simply implemented by dividing the average price for the $n - day$ window selected for each consecutive day of the stock by the price and in this case, the parameter $n = 20$.

Stochastic Oscillator is momentum driven and shows insight into whether the state of the security is in an overbought or oversold position. This is aggregated by taking the minimum and maximum prices of the rolling $n - day$ window which can be seen in the following equation:

$$\%K = \frac{100 * (Recent\ Close - Lowest\ Price\ of\ n{-}Day\ Window)}{(Highest\ Price\ of\ n\ Day\ Window - Lowest\ Price\ of\ n{-}Day\ Window)}$$

In this case, the parameter $n = 14$. Generally, there are two lines involved, one which expresses the current state of that security in terms of price known as %K, and the other which is a moving average of the previous line described known as %D. In this implementation, only %K is used. The values returned from both indicators are from 0 to 100 for each stock price of each date.

All indicators are implemented in the same format for each learner and are specifically optimized for vectorization which benefits run-time.

Manual Strategy

The if-then rules specified for this strategy combine each indicator together based on their nature. Each indicator has one thing in common; they all tell us when to have a long, short, or hold position in the security. The approach used is that if and only if each indicator's values signal a short position in unison, a short position is taken. If a short position is not indicated by the thresholds, then the short position is filled and current shares of the security return to 0 from -1000 and stay at 0 until all indicators signal another short signal. There are also times when the short position is filled and a subsequent long position is also taken.

This method was determined to be performant based on trial and error of the in-sample period and testing different threshold values and seeing which values yield a higher portfolio value overall.

A sample of the approach can be seen in the following pseudocode:

```
if price/sma > 0:
  if momentum < 0:
    if %K >= 55:
      Short
    elif price/sma < 1.05:
      if momentum > .7:
        if %K >= 55:
          Short
        else:
          Long
...
elif momentum < .5 or == 0:
  Long
else:
  pass # take no positions/actions on the market
```

During the meta-analysis of the stock data it was seen that the benchmark in the in-sample period had a low performance with a low cumulative return. This inferred that the stock itself was not performing well, so with this knowledge being known; the approach was tuned more toward short positions. These positions are then filled with no further actions taken if the requirements aren't met. Also, in order to make sure the trading frequency was low, trades were only executed if and only if no trades had been executed in the past 8 days. This lag can be seen at the beginning of each graph, where the line is stagnant or straight in the first 8 days in which no positions in the market are taken on those beginning days as a safety measure. It can be seen in *Figure 1* and *Figure 2* that the manual strategy performed better than the benchmark during the in-sample period and out-sample period. The manual strategy did not perform as well in the out-sample period because the rules and thresholds were tuned only based on the in-sample period. During the out-sample period, it can be seen based on the benchmark that there are new trends that were not evident in the in-sample period which explains the reduction in performance. This is done to simulate how this manual strategy would react to unseen data. In both figures, the black vertical lines indicate short entry points, while the blue vertical lines indicate long entry points.

Figure 1 - Benchmark vs Manual Strategy (in-sample) period

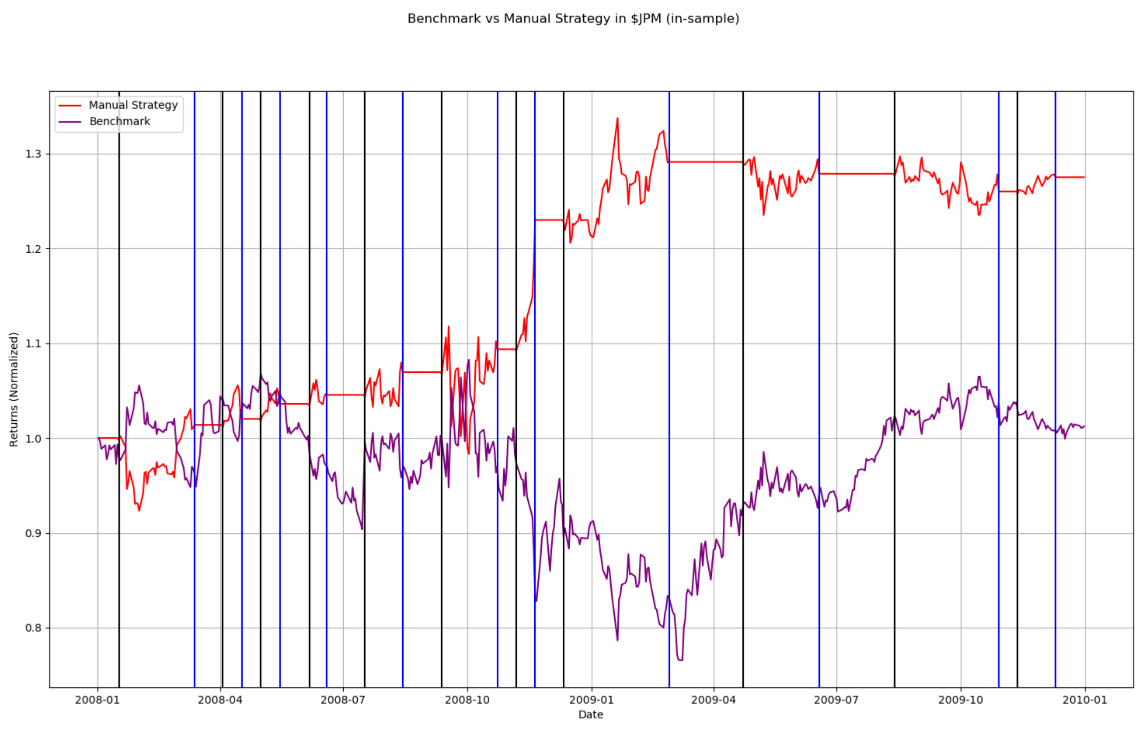
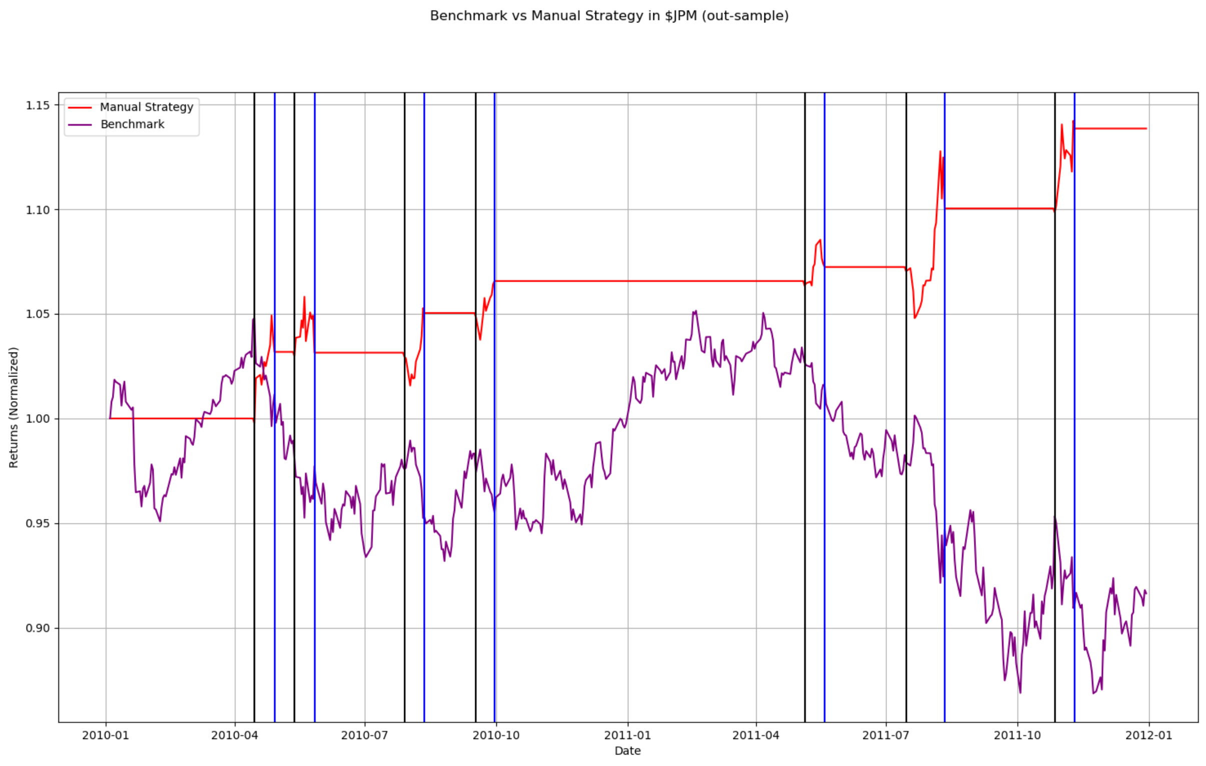


Figure 2 - Benchmark vs Manual Strategy (out-sample) period



Strategy Learner

In order to frame this problem statement for our machine learning algorithm, our algorithm and our problem statement needed to be understood thoroughly. The algorithm learns on input data X that is classified with a target variable Y, it then makes predictions on new incoming data of their classification Y' or target variable based on what it has learned. The problem statement is to allow this machine learning algorithm to learn the patterns of each indicator value and its classification to make predictions or classify future indicator values. Since there are three options, short, out, or long this turns into a multiclass-classification problem where each class can be represented as -1, 0, or 1 respectively.

The dataset in the learning phase is derived from the in-sample period containing each date only where all 3 indicators have values, this can be denoted as our X or input variable as they are all vectorized. Since we do not have a Y, or classification of what the best trade would be at each indicator value of each date, we have to classify each record through labeling. In order to classify each record in a way that would yield the highest portfolio value overall, we apply a simple rule that will classify this training data on when to go short, out or long based on the previous 7-day cumulative return of the stock price data of these dates. This can be seen in the following pseudocode:

```
for each date of our constructed dataset with indicator values:
  if 7-day cumulative return of the stock at that date > 0:
    Y at that date = 1 # long position
  if 7-day cumulative return of the stock at that date < 0:
    Y at that date = -1 # short position
  else:
    Y at that date = 0 # out position
```

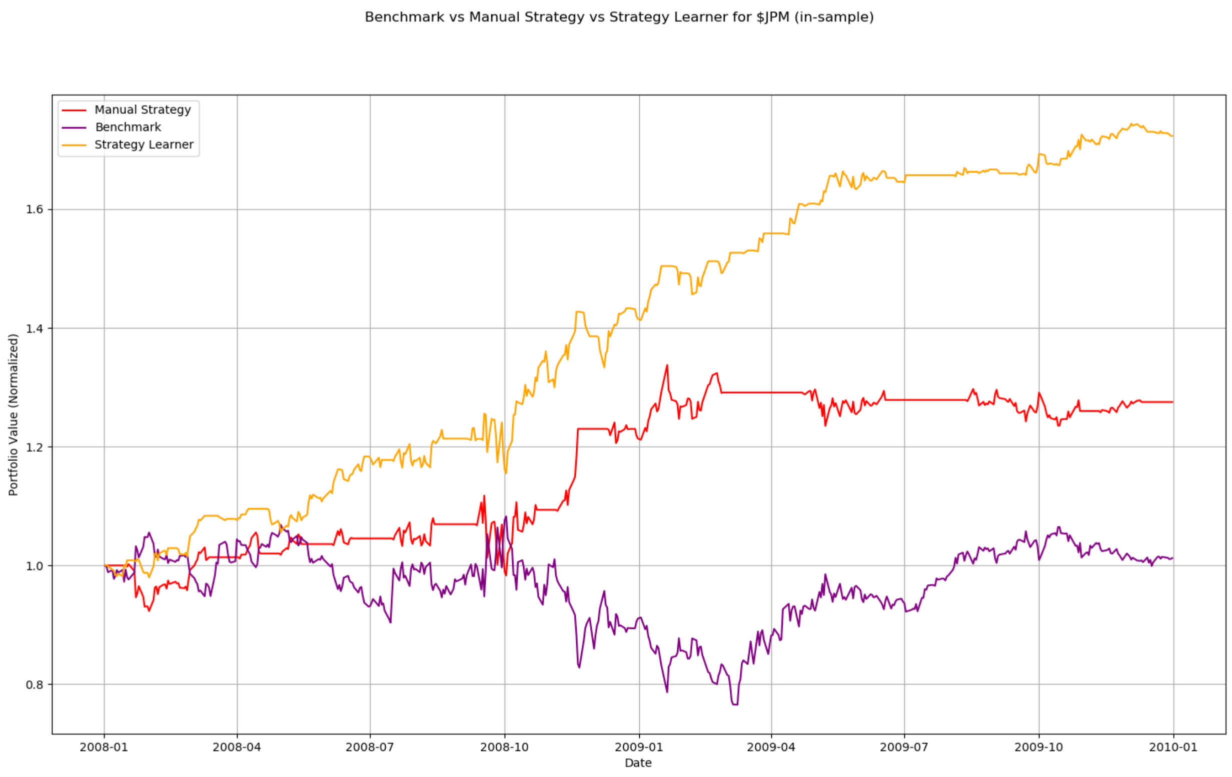
Once our training data was constructed, the algorithm could then learn and create its own rules of when to make predictions based on this compiled dataset. This algorithm uses a specific technique known as bagging, where it creates multiple decision trees or bags with random samples of the data, learns from each sample and combines the predictions of each sample to create collective robust predictions of all of the data.

On the contrary, the performance of the portfolio from the predictions can change based on the amount of bags set during the initialization of the algorithm, this is one of the hyperparameters. Another hyperparameter that can affect predictions is the leaf size which controls the depth or complexity of the decision tree. In order to generalize for different patterns of input data *X*, a grid search was implemented. This implementation will try different hyperparameter values during training, calculate the cumulative return of the predictions on the in-sample data with the tried values and finally return the hyperparameters with the maximum cumulative return. These hyperparameters are then used in the initialization of the model ensuring maximum performance during training. In this approach, the input data or indicator data was not discretized. Since they were in a format that was learnable by the model no additional post-processing was necessary.

Experiment 1

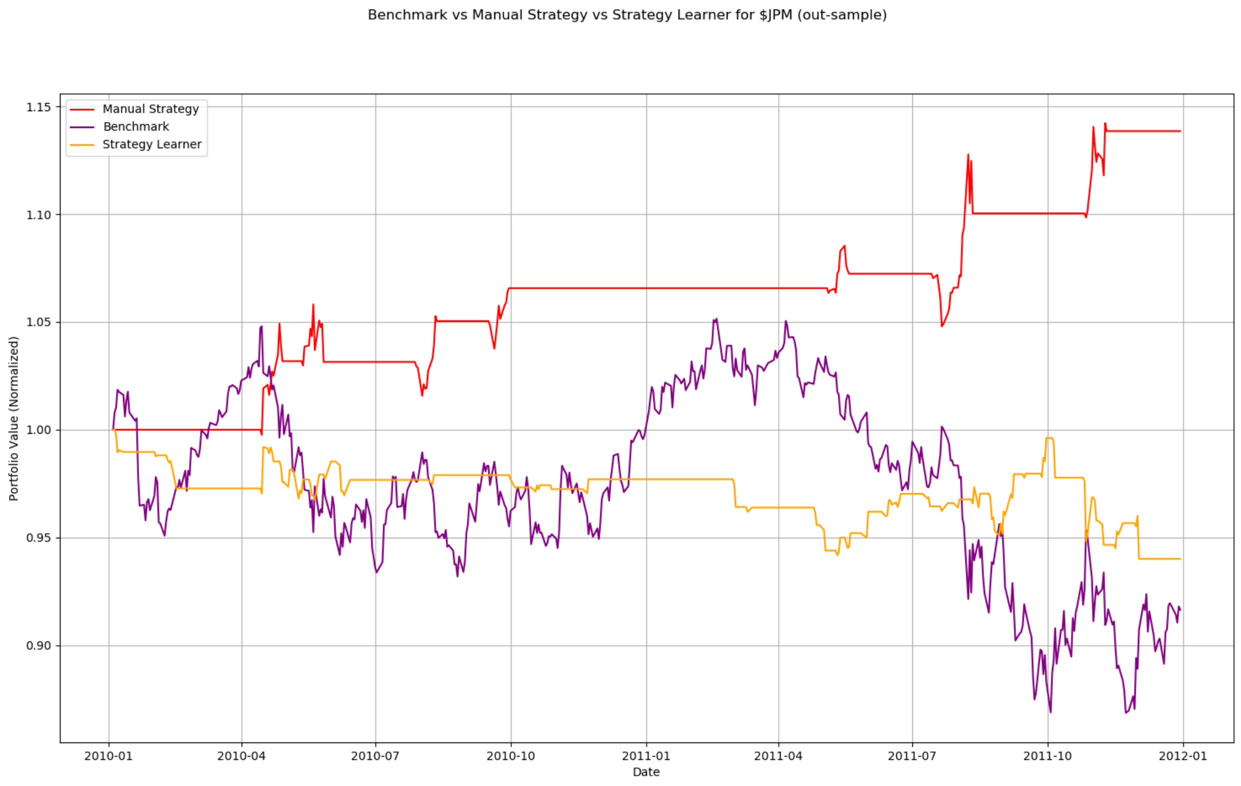
In experiment 1 the strategy learner is compared to the manual strategy during the in-sample period which can be seen in *Figure 3* and in the out-sample period

Figure 3 - Benchmark vs Manual Strategy vs Strategy Learner (in-sample) period



which can be seen in *Figure 4*. The benchmark is also included.

Figure 4 - Benchmark vs Manual Strategy vs Strategy Learner (out-sample) period

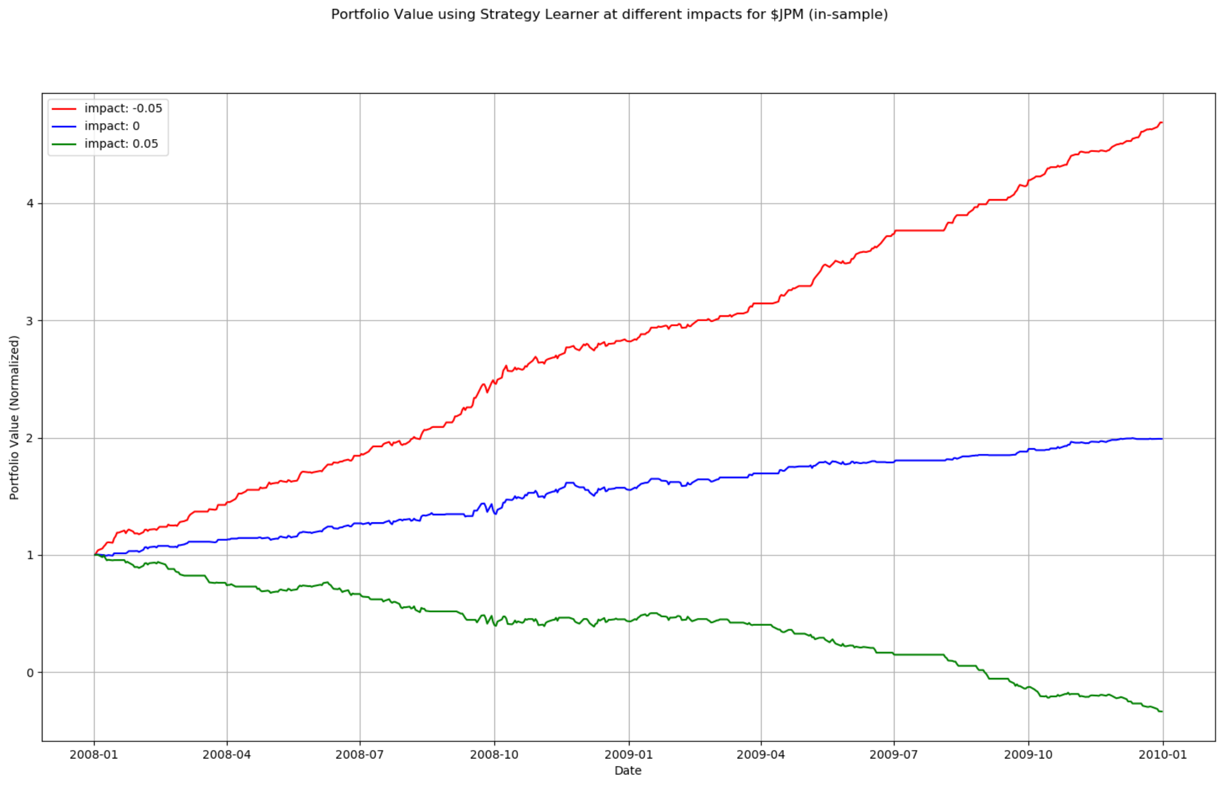


The initial hypothesis is that the strategy learner would perform better than the manual strategy. It can be seen that during the in-sample period the portfolio value of the strategy learner performs better than the manual strategy and the benchmark. However, during the out-sample period, the strategy learner fails to generalize and performs worse than the manual strategy. The manual strategy uses the manual rules set within the period to make trades. Also, the strategy learner is trained on the in-sample period data using the random forest classifier. All of these different perspectives' portfolio values are aggregated using a market simulator. The market simulator aggregates the value of the portfolio based on the current shares held at the specific date of the stock for all dates. Due to the nature of the grid search in the training phase of the strategy learner, there are no specific hyperparameters to set specifically to repeat the experiment as the model is designed to find those parameters on its own. The manual strategy and strategy learner use the same indicators that are mentioned above. The result of the strategy learner and manual strategy would indeed occur every time this experiment is performed on in-sample data due to the fact that both the manual strategy and strategy learner are tuned to perform the best on in-sample data.

Experiment 2

In experiment 2 the value of impact is changed to see how this affects the in-sample trading behavior. The impact is defined as the value that changes the price of a stock due to a specific trade. The strategy learner is trained and its portfolio value is tested three different times, each with a different impact value to see the effect of impact on the portfolio value and sharpe ratio. The sharpe ratio is a measurement that assesses risk-adjusted return which accounts for standard deviation in its mathematical formula. It is hypothesized that as the impact value in the market increases, the portfolio value and sharpe ratio will decrease. It can be seen in *Figure 5* that as the impact decreases, the portfolio value increases. This means that if the trading of a stock caused a decrease in the price, the portfolio value would go up since the stock is cheaper which aligns with the hypothesis as the inverse relationship can be seen in both *Figure 5* and *Figure 6*.

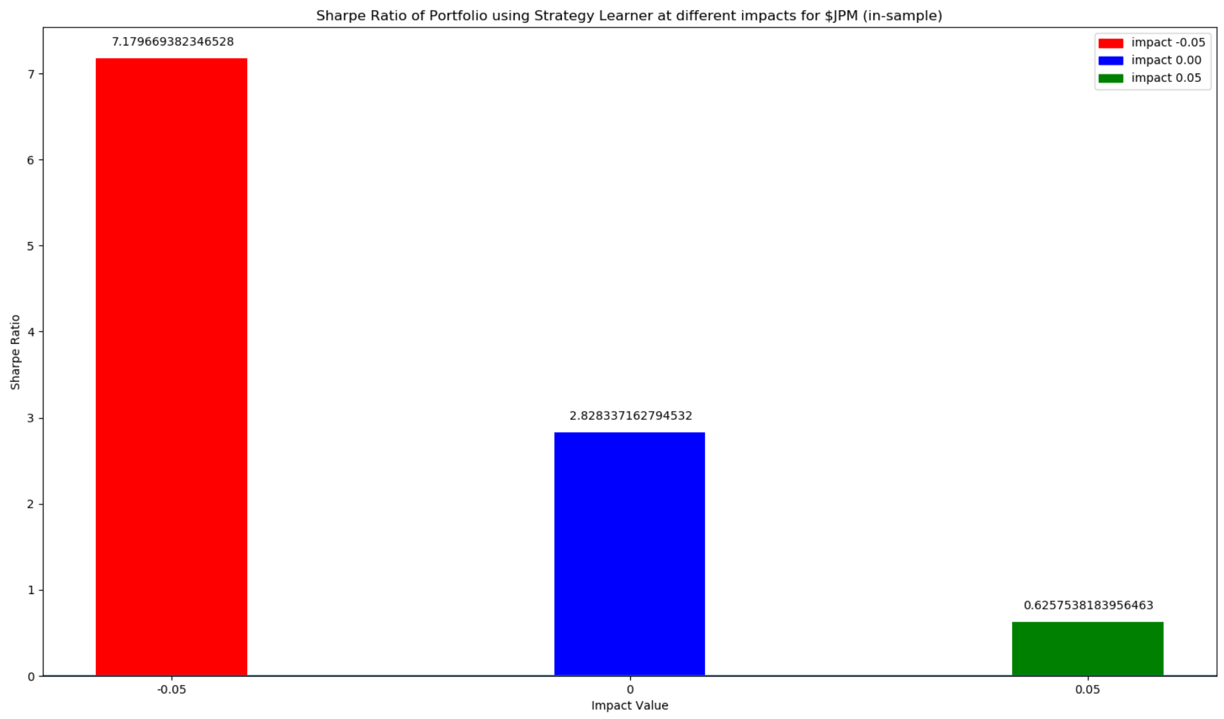
Figure 5 - Portfolio Value using Strategy Learner at 3 different impacts (in-sample) period



In order to reproduce this example, the strategy learner would have to be trained on JPM stock data in the respective periods where the market simulator accounts for each of the three impacts and the portfolio value calculation would then also have to account for these impacts respectively in their parameters initialized.

Each separate portfolio value is generated for each impact and is then plotted in *Figure 5*. The sharpe ratio is then aggregated from each different portfolio and plotted in *Figure 6* to show the differences.

Figure 6 - Sharpe Ratios of different Portfolio Values using Strategy Learner at 3 different impacts (in-sample) period



Additional graphs

Summary statistics of the manual strategy in the in-sample period

Type	Cumulative Returns	STD of Daily Returns	Mean of Daily Returns
Benchmark	0.012325	0.017041	0.000169
Manual Strategy	0.275097	0.011226	0.000545

Summary statistics of the manual strategy in the out-sample period

Type	Cumulative Returns	STD of Daily Returns	Mean of Daily Returns
Benchmark	-0.083579	0.0085	-0.000137
Manual Strategy	0.138553	0.003821	0.000265