Strategy Evaluation Report

Shrikanth Mahale

[Smahale6@gatech.edu](mailto:Smahale6@gatech.edu)

**Abstract—Manual Strategy and Learner based strategy for stock trading**

The goal of this project is to solve a trading problem using a learner which will learn the way we trade and make decision on our behalf on whether or not to trade a stock on a particular day. This goal is achieved by implementing an ensemble learner which uses Bag Learner and RT Learner. Available data is used as training (in-sample) data for the learner and new data was used as testing (out-of-sample) data to test the accuracy of the learner. The actions in a trading problem (BUY, SELL or DO Nothing) were converted to classes in the classification ensemble learning problem (+1,-1,0) , and then trained and tested on in-sample data and out-of-sample data respectively to get a class label which indicates an action(BUY, SELL or DO Nothing) in the trading problem. Similarly, a manual strategy is implemented in parallel experiment is conducted to compare the manual strategy with the learner-based strategy.

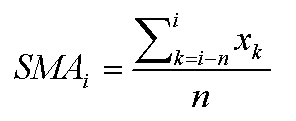
**Indicator Overview**

As mentioned in the introduction, we implemented a learner and a manual strategy and indicators played an important role in this implementation. In manual strategy 3 indicators were used to give us a signal of whether to buy or to sell. In the learner the 3 indicators were used as features which predicts a response of whether to buy or sell or do nothing (1,0, -1).

The same 3 indicators used for both Manual Strategy and Strategy Learner are given below

**Indicator 1: SMA (Simple Moving Average).**

A simple moving average is formed by computing the average price of a stock/security over a specific number of periods. As the name suggest, we calculated a rolling (moving) mean (average) over a specified time window (aka lookback window). In this project we used a lookback window of 14 days. PSMA was calculated as Price/SMA ratio.



**Indicator 2: Bollinger Band Percentage(BBP)**

Bollinger Bands are two volatility bands are two standard deviations plotted above (aka upper Bollinger Band) and below SMA (aka upper Bollinger Band). Bollinger Band Percentage is calculated using the formula given below.

Upper Band = SMA + 2 \* (σ Prices)

Lower Band = SMA - 2\* (σ Prices)

Bollinger Band % = (prices – Lower band)/ (Upper Band – Lower Band) .

The SMA for this BB was calculated with look back window of 14 days.

**Indicator 3: Volatility**

Volatility is further calculated using standard deviation within a rolling window of the Daily return given by mathematical formula below

Volatility = (over a given window)

Where Daily return is calculated using formula below

In this project we used a lookback window of 14 days

**Manual Strategy**

In order achieve Manual Strategy much more optimal than the benchmark strategy, different combination of the values of the above 3 indicators were experimented on what we call as in-sample data (JPM shares between Jan 1 2008 to Dec 31 2009) and the combination of indicator values below turned out to be the best combination explained in pseudocode format

**IF** *Price/SMA ratio < 1* **AND** *BBP < 0* **AND** *Volatility < 0.095*

**THEN** *Signal = 1 (Buy the share)*

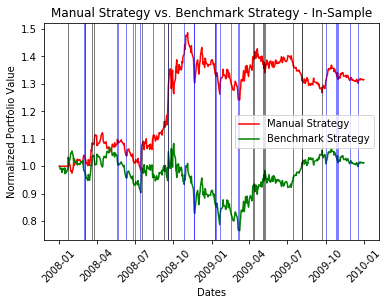
**ELSEIF** *Price/SMA ratio > 1.05* **AND** *BBP < 0* **AND** *Volatility > 0.01*

**THEN** *Signal = -1 (Sell the share)*

**ELSE** *Signal = 0 (Do nothing)*

This strategy does not qualify as effective strategy because in this strategy a trial and error method were used on the in-sample dataset using domain knowledge of indicators to get a strategy which is better than what we define a benchmark strategy. The approach will be a specialized combination of indicators for the data which we are experimenting on and once we change the data this approach may fail. In machine learning terminology, it is like saying we overfitted our approach manually to the in-sample dataset and this approach may fail with the out-of-sample data. We would need to try many combinations of the indicator values to turn this specialized approach into a generalized combination for any dataset.

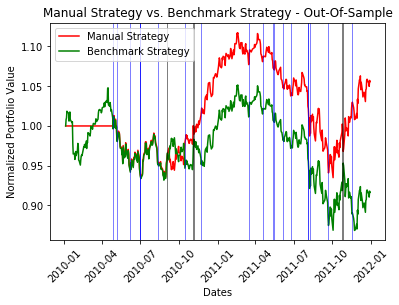
The Fig. 1 and Fig. 2 below justifies my explanation above. Fig 1 shows strategy for in-sample period and Fig-2 for out-of-sample period. The impact and commissions for both periods were set to 0.005 and 9.95 respectively.The blue line and the black vertical line show the number of BUY and SELL respectively in a time period



**Fig. 1**

Looking at Fig. 1 above, we notice that the Manual Strategy starts to outperform the Benchmark Strategy because of the values of the indicators were chosen to perform well in the in-sample data.

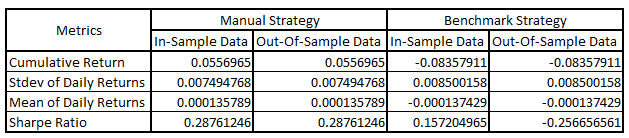
The combinations of indicators that were used on in-sample data was also used on the out-of-sample data (Jan 1 2010 to Dec 31 2012) and was plotted as shown in Fig. 2 below. If we look at Fig. 2, we look at Fig. 2 below unlike Fig.1 the Manual Strategy starts outperforming the little late as (around end of 2010) because the indicator combination that was chosen was not a good fit for this data.



**Fig. 2**

Another observation we see when we compare Fig. 1 to Fig. 2 is that the number of vertical blue lines and black lines in Fig-2 is lesser than that of Fig. 1, which means that number of BUYS and SELLS for the JPM shares for the in-sample data is higher as compared to out-of-sample data. We have more DO NOTHING in out-of-sample data as compared to in-sample data.

Given below are the tables that summarizes the performance of stock and Manual Strategy for both In-Sample and Out-Of-Sample periods.



**Strategy Learner**

The strategy learner was implemented using Ensemble learner which used a combination of RT Learner and Bag Learner. Indicators used in Manual Strategy was used as features in this learner. This was accomplished using steps given below.

Step 1: Setting up of the Learner

* RT Learner and Bag Learner code was implemented using the code used in project 3 with minute changes since this learner would be a classification learner and not a regression learner.
* A leaf size of 5 in RT Learner and 50 Bags in Bag Learner or this project was chosen in order to avoid overfitting as observed in project 3.

Step 2: Training Phase

* This step in performed in the add\_evidence function of StrategyLearner.py file and same in-sample dataset used in Manual Strategy was also used in Training of the learner.
* The 3 indicators Price/SMA, BBP and Volatility were used as features (X) for the RT learner. All the three indicators had a look back of 14 days.
* The response y was supposed to have values -1 or 0 or +1 which denotes SELL or DO NOTHING or BUY respectively. In order to achieve this a ratio of N-Day (N = 10) prices over the current prices was calculated using in-sample dataset as this would give us general indication of whether stock prices are going up or down and also relates well with the chosen indicators. The idea here is if the N-Day ratio is below 0.01 then y = -1, if the N-Day ratio is above 0.01 () then y = 1, else y = 0.
* If we take market impact into consideration then the final value of y in training set is calculated using pseudocode below

**IF** *N-Day ratio > (0.001+Impact)*

**THEN** *y = 1 (Buy)*

**ELSEIF** *N-Day ratio < (-0.001-Impact)*

**THEN** *y = -1 (Sell)*

**ELSE** *y = 0 (Do Nothing)*

Step 2: Testing Phase

* We use our learner built in training phase on the out of sample data to predict on indicators and give the value of Y after the bag learner taking the mode of 50 trees.
* Based on the value of y and the number of shares in holdings, we decide whether to buy or sell 1000 or 2000 shares as we can have max net holding of 1000 shares either LONG or SHORT. This activity is carried out in testPolicy() function.

**Adjustment of Data**

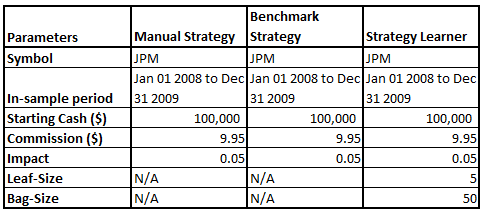
There was no need for adjustment of data because the ensemble technique of bag learner along with RT Learner works well with the data available in this assignment. Just because Random Forest gives us value easily and we take the mode of all possible values in the 50 bags, we generally end up taking the best action. As a result, no manipulation of the data like discretization or standardization is needed

**Experiment 1**

Following steps were performed in order to implement experiment 1.

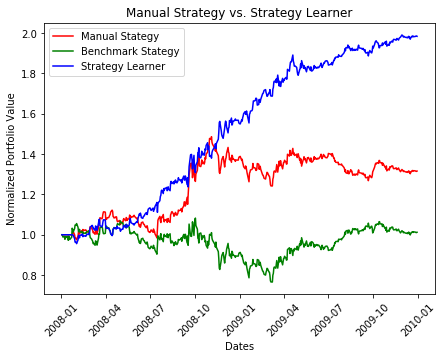
1. Manual strategy was executed with the same combination of Indicators explained in the Manual Strategy section on the JPM shares of the prices dataframe. This was executed calling the ManualStrategy.py file which in turn called the Indicators.py and Marketsimcode.py file.
2. Similarly, benchmark strategy was executed where a dataframe with benchmark conditions (starting with 100,000 investing in 1000 shares of the symbol in use on the first trading day, and holding that position) was used to calculate portfolio values. This was also executed calling the ManualStrategy.py file.
3. A learner-based strategy explained in Strategy Learner section above was then executed, using the StrategyLearner.py file which in turn calls RTLearner.py and BagLearner.py file.
4. Portfolio values, Cumulative Returns, Average Daily Returns, Stdev of Daily returns and Sharpe ratio was calculated after completing the above three steps for the 3 strategies.

Parameters below were used for each strategy the above steps.



The initial hypotheses before executing this result was that strategy learner would out-perform the manual and benchmark strategy because of the reasons mentioned in the Manual Strategy section

After executing all the strategies, their respective normalized portfolio values were plotted as shown in Fig.3 below

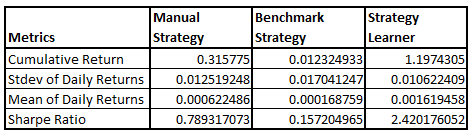


**Fig. 3**

Looking at Fig. 1 above we see that, manual and learner strategies show that while both beat the benchmark, but overall, the Strategy Learner has a better performance because in manual strategy we are making decisions solely on the basis of 3 indicator values whereas strategy our learner uses ensemble methods along with indicators which reduces overfitting and gives better performance on data.

We can expect this same relative result almost all the time with an in-sample period, as a machine learning learner will always perform well on data it has seen before and trained on. In fact, if the learner is trained on the same data repeatedly, it can quite often and soon converge to an optimal strategy.

Given below are some stats associated with experiment 1



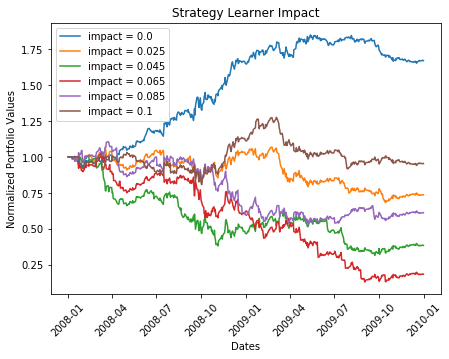
**Experiment 2**

**Hypothesis**

As Impact goes up profit goes down because of which trader becomes less willing to buy or sell stocks. This is because Impact is a cost that a buyer or seller has to incur while transacting. So, taking this aspect into consideration, the hypothesis is that as Impact increases the learner will be more resistant in in making trades and will end up with a lower portfolio value.

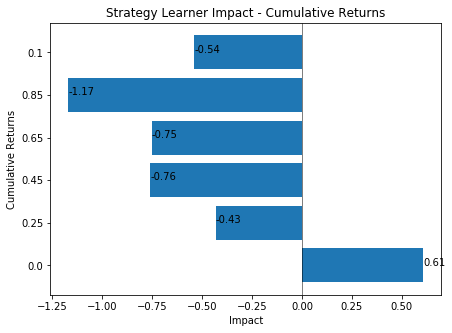
**Explanation**

To conduct this experiment, 6 different strategy learners were created using the code in *ManualStrategy.py* with impact values of 0.0, 0.025, 0.045, 0.065, 0.085 and 0.1 during in-sample period. Commission was set to 0.00 for all learners. Normalized portfolio value which would be one of our metrics of comparison was computed for all learners and plotted against in-sample dates as shown in Fig. 4 below.



**Fig. 4**

The other metrics of comparison would be cumulative return plotted below as shown in Fig. 5



**Fig 5.**

Looking at Fig.4 above, the hypothesis appears to be partly true but completely true. The learner with lowest impact = 0.0 shows best performance with and increased portfolio value by 1.75 times and the learner with impact = 0.085 has the lowest performance (even lower that the learner with Impact = 0.1). Looking at Fig. 5 we see the same performance for Cumulative returns, learner with impact = 0.0 has highest Cumulative returns and learner with impact 0.85 has the lowest Cumulative returns.

This must have happened because the impact is accounted in the learner in such a way that it makes the learner more conservative in making trades as it waits for the future n day market return to be either better than YBUY + impact to buy or lower than YSELL – impact to sell. As a result of this the learner doesn’t maximize the value by making certain trades and instead will choose to hold more often. This shows that it the learner implemented takes a conservative approach and does not trade much for impact = 0.1 preventing huge losses but there are situations where the learner does trades which are not very profitable such as with impact = 0.085