# CS7646 ML4T Proj 8 Strategy Evaluation

Jingya Feng

jfeng89@gatech.edu

## 1.INDICATOR OVERVIEW

Based on 5 indicators I evaluated in Project 6, I chose normalized SMA (and SMA Crossovers) Bollinger Band Percentage, Stochastic Oscillator and Rate of Change Crosseovers to help optimize my Manual Strategy and Strategy Learner.

For this project I set lookback period as 20 for SMA, Bollinger Band and Rate of Change, which all required lookback window. I retrieved 2 * window sizes prices data, prior to the starting date for trading, to avoid getting NA values of indicators resulted from moving window.

All indicators were implemented throughout indicators.py as independent methods that used API as *get_<indicator_name>(start_date,end_date,syms,period)* and returned a dataframe whose values represent corresponding indicator values.

### 1.1 SMA and Price/SMA

SMA is calculated by summing all closing prices over the last "n" periods then divided by "n", which is look period I chose at 20. Prices are normalized by dividing the prices on starting date.

### 1.2 Bollinger Band (BB%)

Bollinger Band is a type of volaitlity indicator and is defined to be a range with 2 standard deviations away from a simple moving average and Bollinger Band % quantifies stock's adjusted close price relative to the upper and lower Bollinger Band. All could be returned through get_bb method, but B% is mainly used for my trading signals. They were calculated as formulas below:

$$Upper\ Bollinger\ Band = SMA + 2 * Rolling\ Standard\ Deviation$$

$$Lower\ Bollinger\ Band = SMA - 2 * Rolling\ Standard\ Deviation$$

$$B\% = \frac{Price - Lower\ Bollinger\ Band}{Upper\ Bollinger\ Band - Lower\ Bollinger\ Band}$$

### 1.3 Rate of Change (ROC)

Rate of Change is a momenntum indicator that measures the price percentage change between current price and the price a certain number of periods ago. I used *pandas.DataFrame.pct_change* on normalized prices and period of 20 to calculate ROC.

$$Rate\ of\ Change = \frac{Price_{current} - Price_{lookback}}{Price_{lookback}}$$

### 1.4 Stochastic Oscillator

Stochastic Oscillator (%K) is calculated as:

$$\%K = \frac{Closing\ Price - L14}{H14 - L14}$$

With industry standard at 14-period window, Stochastic Oscillator does not need to take the lookback period as a parameter. It does need additional adjustment of daily high , low, and closing prices of a stock. So inside *get_sto* method, obtained adjusted closing price and closing price were retrieved through *get_data* method to obtain adjustment ratios for each trading day, before calculating daily adjusted high prices and low prices. After having daily adjusted closing price, high price and low price on hand, I used rolling function as well to get L14 and H14 and final %K based on the equation above.

### 1.5 SMA Crosses and ROC Crosses

In addition to buy and sell signals, I created an price cross signal, utilizing 2 additional dataframes, SMA Crosses and Rate of Change Crosses, because cross over indexes are valid indicators for trend changes to help the strategy decide the timing to exit current positions. Crossover dataframes are constructed as following: when Price/SMA or Rate of Change are positive, they were classified as 1, otherwise 0. Then the differences were took between each row, yielding dates with nonzero values when Price/SMA or ROC changed from positive to negative (or vice versa)

## 2.MANUAL STRATEGY

### 2.1 Manual Strategy Construction and Rationales

The underlying rationale to construct trading strategy is to long when the stock demonstrates oversold signal, and to sell when the stock demonstrates overbought signal. For my manual strategy, Price/SMA, B%, and Stochastic Oscillator were used to indicate overbought/sold.

Since Price moves faster than SMA, when Price is higher than SMA, it usually means the stock is overheated and could imply a good timing to short, vice versa. So I set the threshold to be 5% for Price/SMA ratio: when Price/SMA > 1.05, short the stock; when Price/SMA < 0.95, buy the stock.
For B%, continuous touches on the upper band (B%>1) may indicates an overbought signal while touches on the lower band (B%<0) may suggest an oversold.
For Stochastic Oscillator, industry has a general belief that reading over 80 indicates overbought and below 20 indicates oversold and my manual strategy adopted this belief.
In addition, I also utilized Cross-over indicators, as discussed in previous section, to help determine when to exit current positions: when SMA and ROC crosses both happened, the stock had a higher potential to experience a trend change, implying a possible exit from current position.

To summarize, buy was signaled when all conditions, Price/SMA ratio < 0.95, Bollinger Band % < 0 and Stochastic Oscillatator < 20, have been satisfied.
Sell was signaled when all conditions, Price/SMA ratio > 1.05, Bollinger Band % >1 and Stochastic Oscillatator >20, have been satisfied.
Exit position (desired holding set to 0) when both SMA Cross and ROC Cross have nonzero values.
We will have a dataframe called "order" with +1/-1/0 as desired holding. I then took the difference of order using .diff() to obtain the trade the strategy should implement each day, and multiplied order size at 1,000, which is the *df_trades* my Manual Strategy *testPolicy()* returned. Marketsimcode as an object will take in *df_trade* returned by Manual Strategy *testPolicy()* to compute portfolio value, obtain portfolio statistics and generate Figure 1 and Figure 2.

### 2.2 Manual Strategy Performance Evaluation

Following the indicators and rationales to construct the strategy, similar strategies are applied for the same stock (JPM) for period starting from Jan 1 2010 to Dec 31 2011 (Out-sample period) The following charts have visualized the effectivenss of my Manual Strategy construction for in-sample and out-sample periods.
In both charts, the purple line represented the benchnmark where a portfolio started with $100,000 cash and investing in 1,000 shares of symbol (JPM stock) on the first trading day and holding that position untill the end of the period. The blue and black lines represented LONG and SHORT trades conducted by the manual strategy.

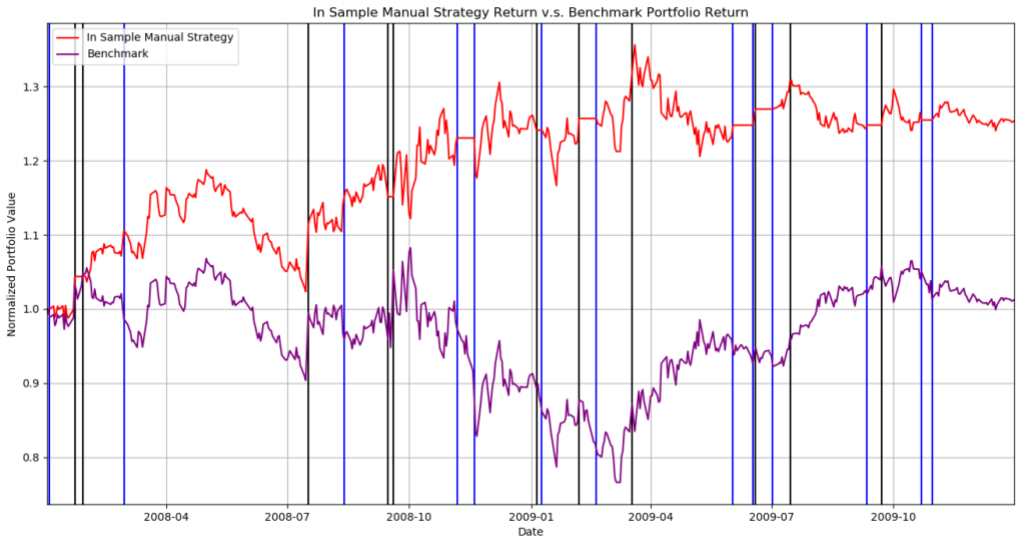


***Figure 1***—In Sample Manual Strategy versus Benchmark

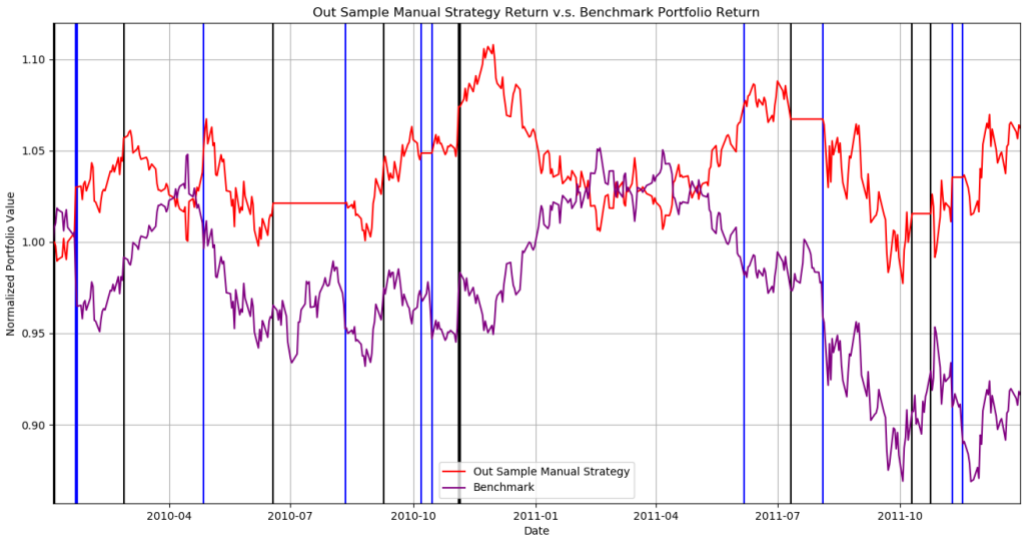


***Figure 2***—Out Sample Manual Strategy versus Benchmark

***Table 1*** —In Sample Performance Metrics Comparison: Benchmark versus Manual Strategy

| Name | CR | ADR | STD | SR |
|---|---|---|---|---|
| Manual Strategy | 0.2536 | 0.000526 | 0.012502 | 0.668302 |
| Benchmark | 0.0123 | 0.000168 | 0.017004 | 0.156918 |

***Table 2*** —Out Sample Performance Metrics Comparison: Benchmark versus Manual Strategy

| Name | CR | ADR | STD | SR |
|---|---|---|---|---|
| Manual Strategy | 0.0624 | 0.000146 | 0.007216 | 0.321964 |
| Benchmark | -0.0834 | -0.000137 | 0.00481 | -0.256813 |

As above performance metrics comparison table demonstrated, In Sample Manual Strategy has substantially outperformed the benchmark: its cumulative return (CR) is almost 21x benchmark CR and also has higher Sharpe Ratio. For Out Sample period, benchmark has lost money whereas Manual Strategy could still make profits through trading.

### 2.3 Manual Strategy In/Out Sample Difference Explanations

We can infer from Figure 2 that during out sample period, Manual Strategy does not consistently beat the benchmark; there are periods when Manual Strategy did not make trade at

all, probably due to the fact that the criteria to trigger buy/sell/exit signals are too strict to be fufilled during that period.
We can also observe that Manual Strategy did not perform as well for Out Sample as for In Sample. Factors that could lead to differences include the general financial market environment fluctuations: The phoneomenal global recession started in 2008 in our in-sample period, and as a U.S. bank, JPM is one of the companies impacted the most. Therefore, the company, the industry and the whole financial market may have completely different performances during Out Sample period, in which the indicators selected might now work ideally as hypothesized. In addition,from Efficient Market Theory, technical analysis can only make money when market is inefficient. In real-life, it is clearly impossible to purely rely on technical indicators to make trading decisions and expect the strategy to make profits consistently.

## 3.STRATEGY LEARNER

### 3.1 Strategy Learner and Random Tree Learner Set Up

Strategy Learner is set up to be a class that takes parameter verbose, impact and commission. It also has parameters Random Tree Learner (with leaf sizes) to help learn based on training data, N to indicate N day return when deriving trade data as Y_train, and Y_BUY and Y_SELL set up within the class to indicate the threshold for Strategy Learner to make trades.
For my Strategy learner, I used leaf size =5 for Random Tree Learner, N = 1 (to construct trades based on next day's return) and Y_BUY & Y_SELL = 2*impact (0 for experiment 1, adjustable with different measurements for experiment 2)

The following functions were all implemented:

- *get_indicators():* a helper method that takes symbol, starting date, ending date and looback window to return concatenated selected indicators as X for training and testing phase.
- *add_evidence():* formulates *X_training*, classifies trade data as *Y_training*, and interacts with learner object through calling *learner.add_evidence* to supplement Random Tree Learner with testing data to build the tree
    - *X_training*: 3 indicators (Price/SMA, Bollinger Band% and Stochastic Oscillator) and 2 crossover indicators concatenated together as X_train for JPM during the In-Sample period, obtained through get_indicator().
    - *Y_training:* Following the suggested classification trader approach on project Wiki, Y_train should be a dataframe of values +1/-1/0 representing the ideal holding the learner should take at the date, based on the future N day return. Besides N, Strategy Learner also has hyperparameters Y_BUY and Y_SELL to help adjust the threshold for Strategy Learner to make trades which will be reflected through Y_train. They were all set up based on the commission parameter the Strategy Learner received, to facilitate experiment 2.
      To determine the ideal holding, daily return are discretized with help from pandas function *.diff(periods=N)* based on daily prices. Then daily_return

dataframe was shifted N periods ahead, as the classification is calculated using data from the future.
The ideal holding is constructed in a similar manner as in Manual Strategy: if the daily return is larger than Y.BUY, we should go LONG; if the daily return is smaller than Y.SELL, we should go SHORT. It means that with no market impact, we should go LONG when next day's return is positive and vice versa.

- *testPolicy():* learning phase, formulates X_test with given parameters (stock symbol, start trading date and end trading date), calls learner's query method using parameters and returns trade Dataframe learned.

As training data has been converted as classification method (-1/+1/0), Random Tree Learner was also adjusted from taking mean to taking mode during the tree building process.
Similar to what I did for Manual Strategy, I also used Marketsimcode to obtain portfolio values and statistics.

## 4. EXPERIMENT 1

### 4.1 Experimental Setup

Experiment 1 compared the results of Manual Strategy and Strategy Learner trading JPM during in-sample period (from Jan 1st, 2008 to Dec 31st, 2009) with starting cash at $100,000. Both market impact and commision have been set up to 0 for experiment 1. Both Strategies are assumed to be taking only Long, Short and Cash positions with order size = 1000. The maximum positions a strategy can hold is also +/- 1000. Benchmark of holding JPM stocks was also included as a reference.

Experiment 1 only contains 1 method Run() to finish all tasks for experiment 1, including trading order generation, portfolio statistics comparison and charts generation. Within the Run() method, 2 Marketsimcode objects were initiated firstly with in-sample and out-sample period to take in corresponding Manual Strategy, Strategy Learner and Benchmark orders (ms_order_in/out returned by ms.testpolicy(), sl_order_in/out returned by sl.testpolicy(), and b_orders_in/out constructed mannually as a dataframe) Corresponding trading strategy portfolios and benchmark portfolio results were obtained through .compute_portvals() in marketsimcode objects and plotted.

## 4.2 Experimental Results

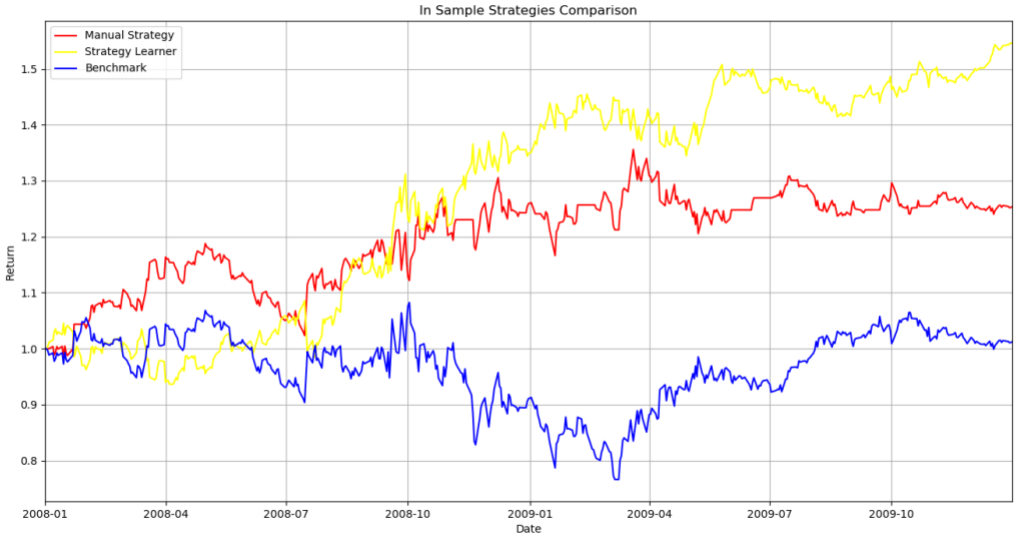


***Figure 3***—In Sample Manual Strategy, Strategy Learner, and Benchmark Comparison

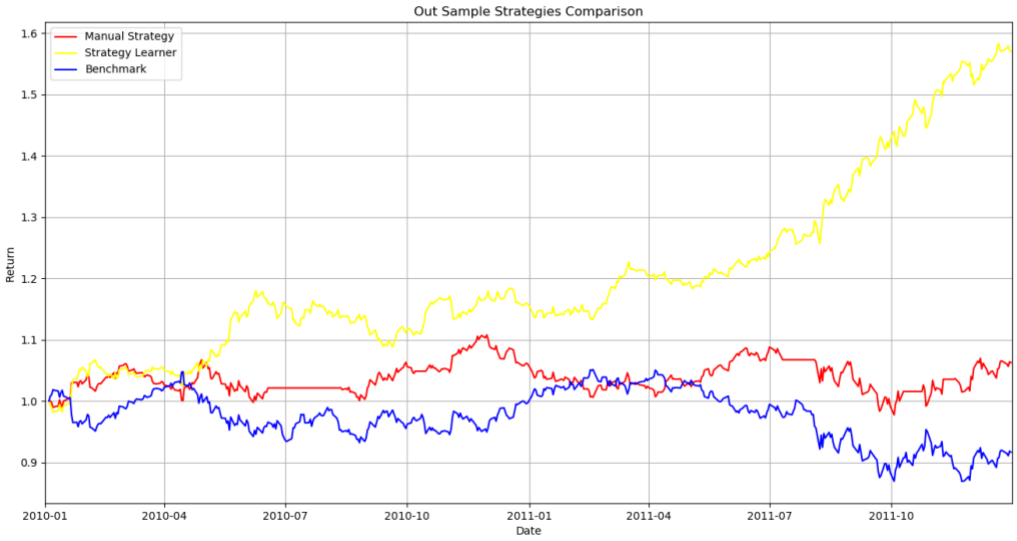


***Figure 4***—Out Sample Manual Strategy, Strategy Learner, and Benchmark Comparison

Inferred from charts above, Strategy Learner has outperformed than benchmark and manual strategy for in-sample period and out-sample period. Strategy Learner has reached the cumulative return over 50% for in-sample period. The result for in-sample period are expected because we trained the learner using in-sample data, which means trades made during in-sample period are ideal trades to generate desired returns. Therefore they should generate higher return than the benchmark and generally the manual strategy. For out-sample period, it is not definite that Strategy Learner will always outperform.

## 5.EXPERIMENT 2

### 5.1 Experimental Setup

Experiment 2 explored how variations of impact would affect trading-related metrics. Market impact is defined to be the amount the price moves against the trader data at each transaction. The price would increase by the impact when buying and decrease by the impact when selling. Hypothetically, a higher impact value would lower the volume traded and negtaively affect the trading strategy performance.

Firstly, commission and impact have already been taken care of in marketsimcode compute_portval() when computing portfolio values: Cash used for each trade is calculated based on price * (1 +/- market impact)+commission. However, impact and commission should also be taken into consideration at Strategy Learner during adding evidence phase. When calculating daily return, dataframes with market impacted buy prices values and sell prices values were firstly calculated, where buy prices equal to daily adjusted closing prices * (1+impact) and sell prices equal to daily adjusted closing prices * (1-impact). Then daily buy return and daily sell return are calculated as followed: daily buy return would be next day's daily sell prices – today's daily buy prices, and daily sell return would be next day's daily buy prices – today's daily sell prices, as a trader always needs th opposite trade to close up the position. Finally, daily total return in dollar values are daily buy return + daily sell return and are shifted with similar manner when impact = 0 to indicate future N day return

Not only for daily return calculation, the threshold to determine when to make trades Y_BUY and Y_SELL are also adjusted to be 5*Impact value in Strategy Learner. When

Experiment 2 also contains contains 1 method Run() to finish all tasks. Within the Run() method, I selected 3 impact values from 0, 0.001 to 0.01 and will observe changes of performance metrics, including cumulative return, average daily return, standard deviation, and sharpe ratio, based on these different impact measurements. All other parameters (symbol, In Sample and Out Sample period, commssions, maximum positions etc.) except for impact remained the same. For each measurement, an Marketsimcode object was initiated with in-sample period to take in corresponding Manual Strategy, Strategy Learner and Benchmark orders (ms_order_in/out returned by ms.testpolicy(), sl_order_in/out returned by sl.testpolicy(), and b_orders_in/out constructed mannually as a dataframe) Corresponding trading strategy portfolios and benchmark portfolio results were obtained through .compute_portvals() in marketsimcode objects and plotted.

## 5.2 Experiment Result

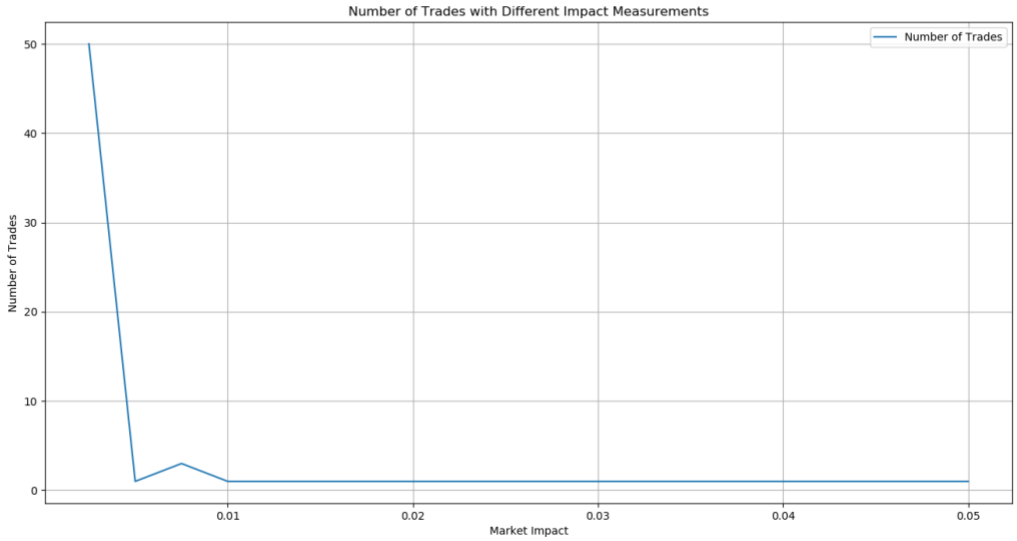


***Figure 5***—Number of Trades with Different Impact Measurments


***Figure 6***—Portfolio Value with Different Impact Measurments

***Table 3*** —In Sample Strategy Learner Performance Metrics Comparison w/ Different Impact Values

| Impact Measurements | CR | ADR | STD | SR |
|---|---|---|---|---|
| 0.0025 | 0.318435 | 0.000636 | 0.013202 | 0.764432 |
| 0.005 | -0.012324 | 0.000095 | 0.015453 | 0.097297 |
| 0.0075 | 0.014124 | 0.000146 | 0.015384 | 0.15075 |
| 0.01 | -0.012348 | 0.000095 | 0.015482 | 0.097524 |
| 0.05 | -0.012541 | 0.000098 | 0.015717 | 0.099372 |

From Figure 5, we can see that higher impact leads to fewer trades made, which in turn may have negative influence on trading strategy's profitability, as demonstrated in Figure 6. There are no trade placed for certain impact measurements which could be furthur explored.

## 6 REFERENCES


1. Investopedia. Retrieved Nov 21, 2022, from https://www.investopedia.com/terms/s/stochasticoscillator.asp
2. Desjardins, J. (2019, March 11). *12 Types of Technical Indicators Used by Stock Traders*. Visual Capitalist. Retrieved Nov 23, 2022, from https://www.visualcapitalist.com/12-types-technical-indicators-stocks/