Report: Project 6

CS7646 Machine Learning for Trading, 2019 Spring

Hui Xia (hxia40)
903459648
Georgia Institute of Technology

1. Part 1: Technical Indicators:

In this report, four technical indicators, i.e. Price/Simple Moving Average(SMA) ratio, Relative Strength Index (RSI), Money Flow Index (MFI), and the CBOE Volatility Index (VIX), are used. The details on how they were coded, and the purpose of using each are described below.

1.1. Price/SMA ratio

1.1.1. Introduction

SMA is a well-known and widely-used **trend indicator** in stock technical analysis. The simple moving average over a range of time (i.e. the 'lookback') provide the analyst a 'smoothened' trend, i.e. it helps the analyst to remove price action noises. It is generally considered that the longer the 'lookback' is used, the more accurate it describes the trend. However, using longer lookback will also result in a 'slower' reaction to fast-changing price actions. In practice, 9, 14, 25, 50, 100, and 200-day SMA are often used, depend on the analysis/trading style of the user.

Since this project requires the student to make multiple trading decisions over a relative short time period (i.e. two years), in this report, the relative position of the stock price against its 14-day SMA is used to demonstrate the trend of the stock price movement.

1.1.2 Implementation algorithm

Using the in-sample data as an example, how to realize this indicator is described below, followed by the resulted plot (**Figure 1**) drawn using this algorithm.

a) Read the stock price as a dataframe by the '`get_data'` function from '`util.py'`, using from January 1, 2008 to December 31, 2009 at the '`date_range`' argument 'JPM' at the '`symbols`' argument, 14 at the '`lookback`' argument, 'True' at the '`addSPY`' argument, and 'Adj Close' at the '`colname`' argument.

b) First forward fill, then backward fill the stock price dataframe. Drop the 'SPY' column from the dataframe.

c) Normalize the first day price of the target stock ('JPM') by dividing all stock price values to its price at the first trading day.

d) Calculate 14-day SMA using the `dataframe.rolling().mean()` function. Calculate Price/SMA ratio using the calculation result of dividing the stock price by the 14-day SMA.

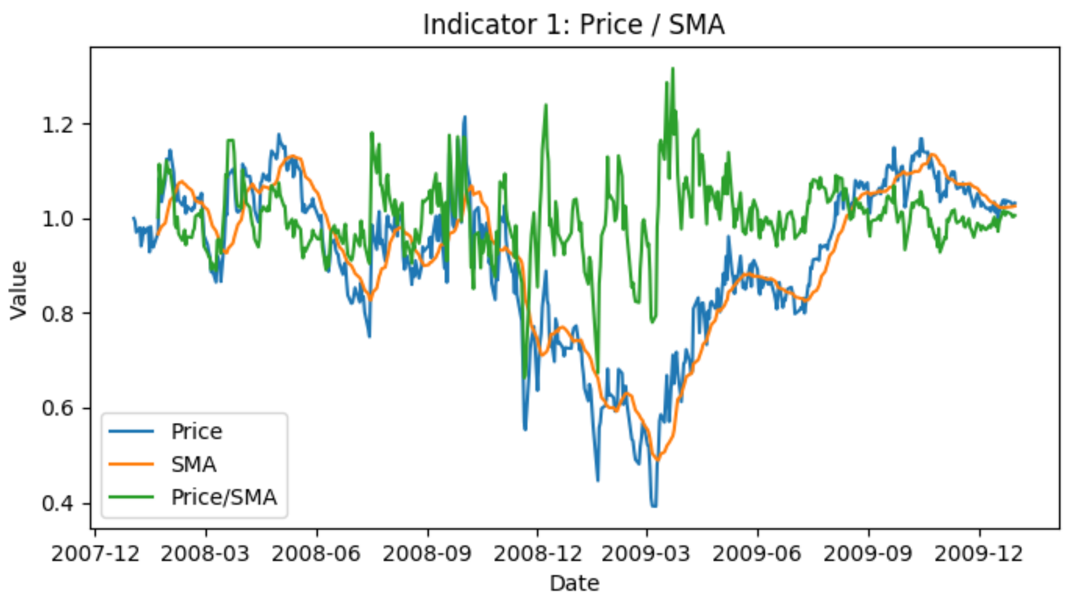

Figure 1: Line plot of price/SMA, SMA, and the stock price. The latter two have been normalized to 1 at the beginning to accommodate with the plot.

1.2. RSI

1.2.1. Introduction

RSI is a well-known and widely-used **reversal indicator** in stock technical analysis. It compares the magnitude of average price gains and losses over a given time period ('lookback') to evaluate the price performance (in the range from 0 to 100) of an asset. Usually, if the value of RSI is higher than 70, the asset is considered as 'overbought' and the chance of a bearish reversal is increased. On the other hand, if the value of RSI is lower than 30, the asset is considered as 'oversold' and the chance of a bullish reversal is increased.

Similar to the case for SMA described above, since that this project requires the student to make multiple trading decisions over a relative short time period (i.e. two years), in this report, 14-day RSI is used to signal bullish price reversal and potential long entry point. RSI is calculated using the equation below:

$$RSI \ = 100 - \frac{100}{(1+RS)} \tag{1}$$

Where RS is calculated as:

$$RS = \frac{asset\ gain/_{lookback}}{asset\ loss/_{lookback}} \tag{2}$$

Where lookback is a pre-defined time period (which equals to 14 in this report) the RSI is calculated upon, and asset gain and asset loss are the total gains and losses during the lookback period. Specially, if there is no asset loss, the value of RSI is set to 100.

1.2.2. Implementation algorithm

Using the in-sample data as an example, how to realize this indicator is described below, followed by the resulted plot (**Figure 2**) drawn using this algorithm.

a) Read the stock price as a dataframe by the '`get_data`' function from '`util.py`', using from January 1, 2008 to December 31, 2009 at the '`date_range`' argument 'JPM' at the '`symbols`' argument, 14 at the '`lookback`' argument, 'True' at the '`addSPY`' argument, and 'Adj Close' at the '`colname`' argument.

b) First forward fill, then backward fill the stock price dataframe. Drop the 'SPY' column from the dataframe.

c) Calculate 'up gain' and 'down loss' using `dataframe.rolling().where()` and `dataframe.rolling().sum()`. The former is used to distinguish upwards and downwards movement in a given lookback period, and the latter is used to calculate the sum thereof.

d) Calculate RS using the 'up gain' and 'down loss', under **Equation (2)**.

e) Calculate RSI using RS, under **Equation (1)**.

f) For demonstration purposes, normalize the first day price of the target stock ('JPM') by dividing all stock price values to its price at the first trading day, then times 50.

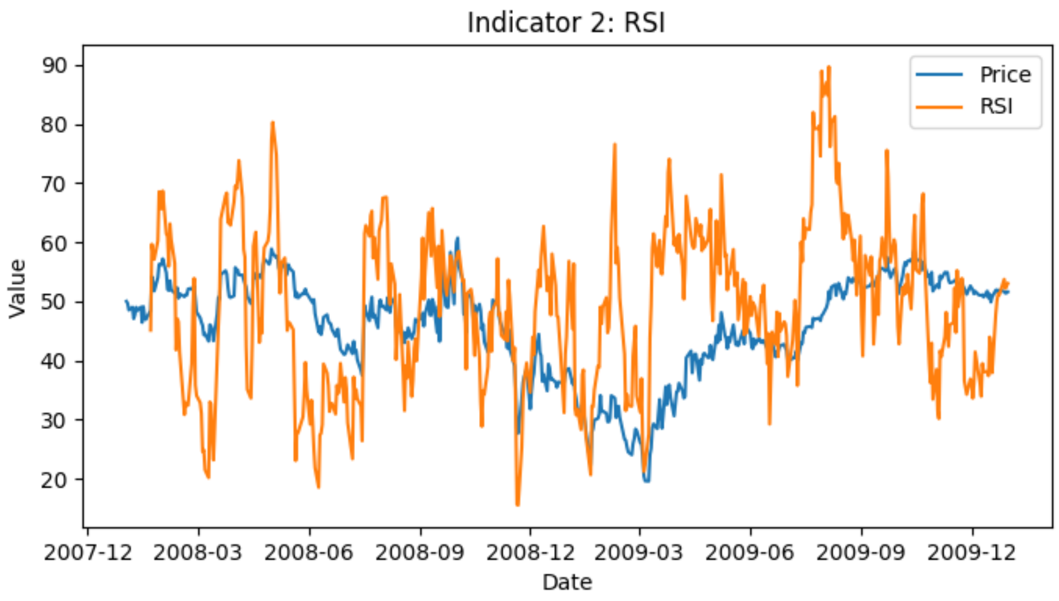

Figure 2: Line plot of the RSI and the stock price (which has been normalized to 50 at the beginning to accommodate with the plot).

1.3. MFI

1.3.1. Introduction

MFI is another well-known **reversal indicator** in stock technical analysis. It compares the magnitude of average price and volume gains and losses over a given time period ('lookback') to evaluate the price performance (in the range from 0 to 100) of an asset. Usually, if the value of MFI is higher than 70, the asset is considered as ‘overbought’ and the chance of a bearish reversal is increased. On the other hand, if the value of MFI is lower than 30, the asset is considered as ‘oversold’ and the chance of a bullish reversal is increased.

Similar to the cases for SMA and RSI described above, since that this project requires the student to make multiple trading decisions over a relative short time period (i.e. two years), in this report, 14-day MFI is used to signal price bearish reversal and potential short entry point. MFI is calculated using the equation below:

$$MFI = 100 - \frac{100}{(1+Money\ Flow)} \tag{3}$$

Where Money Flow is calculated as:

$$Money\ Flow = \frac{(asset\ gain \times gain\ volume)/_{lookback}}{(asset\ loss \times loss\ volume)/_{lookback}} \tag{4}$$

Where lookback is a pre-defined time period (which equals to 14 in this report) the MFI is calculated upon, asset gain and asset loss are the total gains and losses during the lookback period, gain volume and loss volume are the total volume during the asset gains or losses, respectively. Specially, if there is no asset loss, the value of MFI is set to 100.

1.3.2. Implementation algorithm

Using the in-sample data as an example, how to realize this indicator is described below, followed by the resulted plot (**Figure 3**) drawn using this algorithm.

a) Read the stock price as a dataframe by the ‘`get_data`’ function from ‘`util.py`’, using from January 1, 2008 to December 31, 2009 at the ‘`date_range`’ argument ‘JPM’ at the ‘`symbols`’ argument, 14 at the ‘`lookback`’ argument, ‘True’ at the ‘`addSPY`’ argument, and ‘Adj Close’ at the ‘`colname`’ argument.

b) Read the stock volume as a dataframe by the ‘`get_data`’ function from ‘`util.py`’, using from January 1, 2008 to December 31, 2009 at the ‘`date_range`’ argument ‘JPM’ at the ‘`symbols`’ argument, 14 at the ‘`lookback`’ argument, ‘True’ at the ‘`addSPY`’ argument, and ‘Volume’ at the ‘`colname`’ argument.

c) First forward fill, then backward fill the stock price and the volume dataframe. Drop the ‘SPY’ column from them.

d) Calculate ‘up gain’ and ‘down loss’ for both dataframes using `dataframe.rolling().where()` and `dataframe.rolling().sum()`. The former is used to distinguish upwards and downwards movement in a given lookback period, and the latter is used to calculate the sum thereof.

e) Calculate Money Flow using the ‘up gain’ and ‘down loss’ from both the stock price and the volume dataframes, under **Equation (4)**.

f) Calculate MFI using Money Flow, under **Equation (3)**.

g) For demonstration purposes, normalize the first day price of the target stock (‘JPM’) by dividing all stock price values to its price at the first trading day, then times 50.

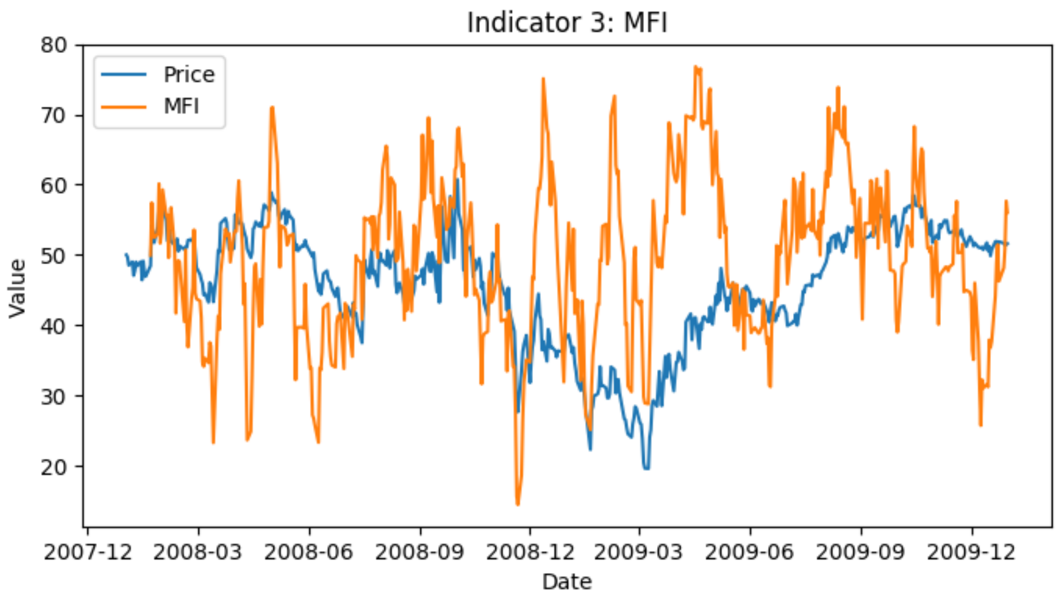

Figure 3: Line plot of the MFI and the stock price (which has been normalized to 50 at the beginning to accommodate with the plot).

1.4. VIX

1.4.1. Introduction

Different than other technical indicators that are usually calculated from the stock price and/or volume, VIX is published by Chicago Board Options Exchange (CBOE). VIX is a measure of expected price fluctuations in the S&P 500 Index options over the next 30 days. In layman’s words, VIX measures how fiercely the market expect the S&P 500 Index to move in the next 30 days. Higher VIX means the market expect the S&P 500 Index to (usually bearish) move further away from its current Value.

Since that most investors hate risk, higher VIX can also be explained as the market is more “fearful”. In this report, high value (>50) of VIX is used to prevent any long entries, as the market lacks confidence. On the other hand, low value (<20) of VIX is used to prevent any short entries, as the market is confident, and any attempt of shorting tend to cause less gain (or even loss) than expected.

1.4.2. Implementation algorithm

Using the in-sample data as an example, how to realize this indicator is described below, followed by the resulted plot (**Figure 4**) drawn using this algorithm.

a) Read the stock price as a dataframe by the ‘`get_data`’ function from ‘`util.py`’, using from January 1, 2008 to December 31, 2009 at the ‘`date_range`’ argument ‘$VIX’ at the ‘`symbols`’ argument, 14 at the ‘`lookback`’ argument, ‘True’ at the ‘`addSPY`’ argument, and ‘Adj Close’ at the ‘`colname`’ argument.

b) First forward fill, then backward fill the stock price and the volume dataframe. Drop the ‘SPY’ column from them.

c) For demonstration purposes, normalize the first day price of the target stock (‘JPM’) by dividing all stock price values to its price at the first trading day, then times 50.

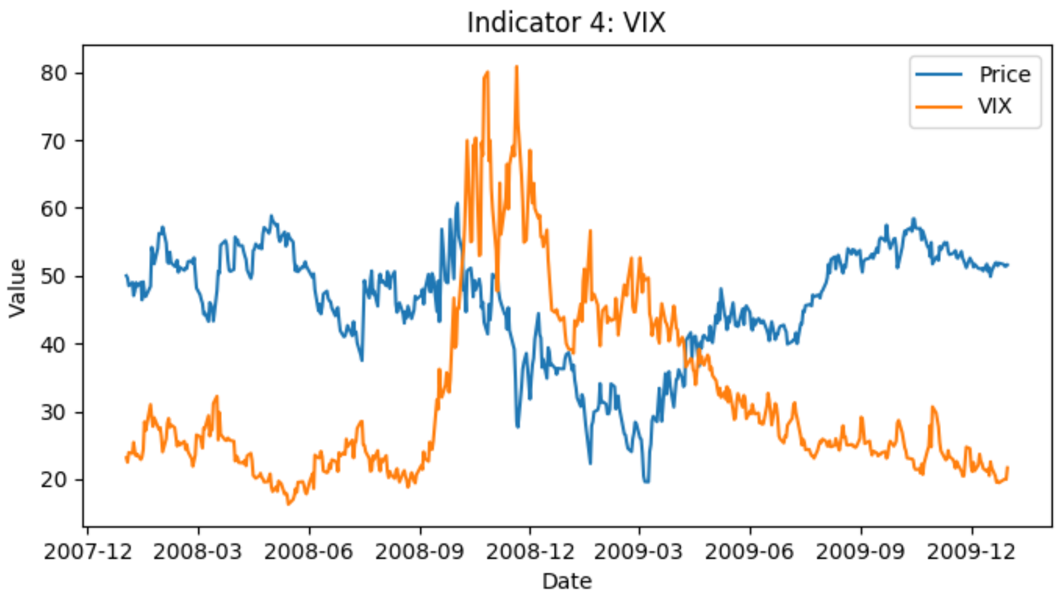

Figure 4: Line plot of the VIX index and the stock price (which has been normalized to 50 at the beginning to accommodate with the plot).

2. **Part 2**: **Comparative Analysis**

2.1. Experimental Methods

If I can see the future, with the given limitations, the best trading strategy will be long 1000 shares if the stock makes a bullish move or does not move, and short 1000 shares if the stock makes a bearish move the next trading day. I achieved this using the following steps:

a) Read the stock price as a dataframe by the ‘`get_data`’ function from ‘`util.py`’, using from January 1, 2008 to December 31, 2009 at the ‘`date_range`’ argument ‘JPM’ at the ‘`symbols`’ argument, 14 at the ‘`lookback`’ argument, ‘True’ at the ‘`addSPY`’ argument, and ‘Adj Close’ at the ‘`colname`’ argument.

b) First forward fill, then backward fill the stock price dataframe. Drop the ‘SPY’ column from the dataframe.

c) Normalize the first day price of the target stock (‘JPM’) by dividing all stock price values to its price at the first trading day.

d) Calculate and find out the ‘winning day’ and ‘losing day’ from the stock price dataframes using `dataframe.rolling().where()`.

e) Create a dataframe `df_trade` that adjusts the holdings to long 1000 shares if the stock moves up, and short 1000 shares if the stock moves down the next trading day. Return the `df_trade` to `marketsim.py`.

2.2. Result

Figure 5 demonstrates the benchmark and the value of the theoretically optimal portfolio. The cumulative return of the benchmark and the theoretically optimal portfolio are 5.7861 and 0.0123, respectively. The standard deviation of daily return of the bench mark and the theoretically optimal portfolio are 0.0045 and 0.0170, respectively. The mean of daily returns of the bench mark and the theoretically optimal portfolio are 0.00382 and 0.00017, respectively.

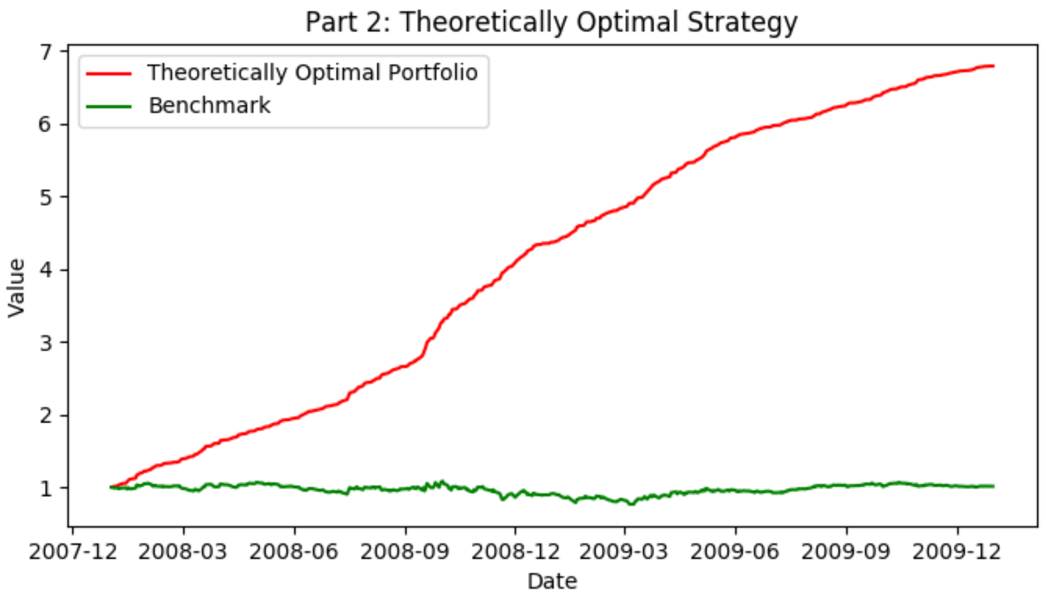

Figure 5: Performance of benchmark (normalized to 1.0 at the start, green line) and the theoretically optimal portfolio (normalized to 1.0 at the start, red line) over the in-sample period. (Transaction costs: Commission $0.00, Impact 0.000)

3. **Part 3**: **Comparative Analysis**

3.1. Experimental Methods

Using the technical indicators described in **Part 1**, a set of trading rules is implemented via calling `indicators.py` at `ManualStrategy.py`. The target of these rules is to make long entry when the market is oversold (when RSI or MFI < 30), and to make short entry when the market is overbought (when RSI or MFI > 70). The long and short entry signals are defined by RSI and MFI, respectively. The reason to choose MFI but not RSI for short entry signal is, it generally needs to be more conservative to short an accredited company's stock (e.g. our benchmark, 'JPM'), and thus volume confirmation is needed to make a short entry. The price/SMA ratio is used to confirm the trend reversal. In this project, the author expects to short the bullish trend (when price/SMA ratio > 1) and long the bearish trend (when price/SMA ratio < 1). Additionally, VIX is used to encourage the manual strategy to be more conservative by not entering long position when market is fearful (when VIX >= 50) and not entering short position when market is greedy (when VIX <= 20). The detailed trading strategy is:

Long entry: when RSI < 30 & price/SMA ratio < 1 & VIX < 50

Short entry: when MFI > 70 & price/SMA ratio > 1 & VIX > 20

Long exit: when RSI > 55

Short exit: when MFI < 55

Then, similar to **Part 2**, a dataframe `df_trade` that adjusts the holdings to long 1000 shares of the target stock if long entry appears, short 1000 shares if the short entry appears, and exit the respective position if long or short exit appears. The `df_trade` dataframe is then returned to `marketsim.py`.

3.2. Result

Figure 6 demonstrates the benchmark and the manual strategy portfolio. Obviously, the performance of the benchmark is similar to what has been shown in **Part 2.2**. The cumulative return, the standard deviation of daily return, and the mean of daily return of the benchmark strategy is 0.0123, 0.0170, and 0.00017, respectively. On the other hand, the manual strategy performs better (i.e. higher return and lower standard deviation). The cumulative return, the standard deviation of daily return, and the mean of daily return of the manual strategy is 0.3594, 0.0081, and 0.00064, respectively.

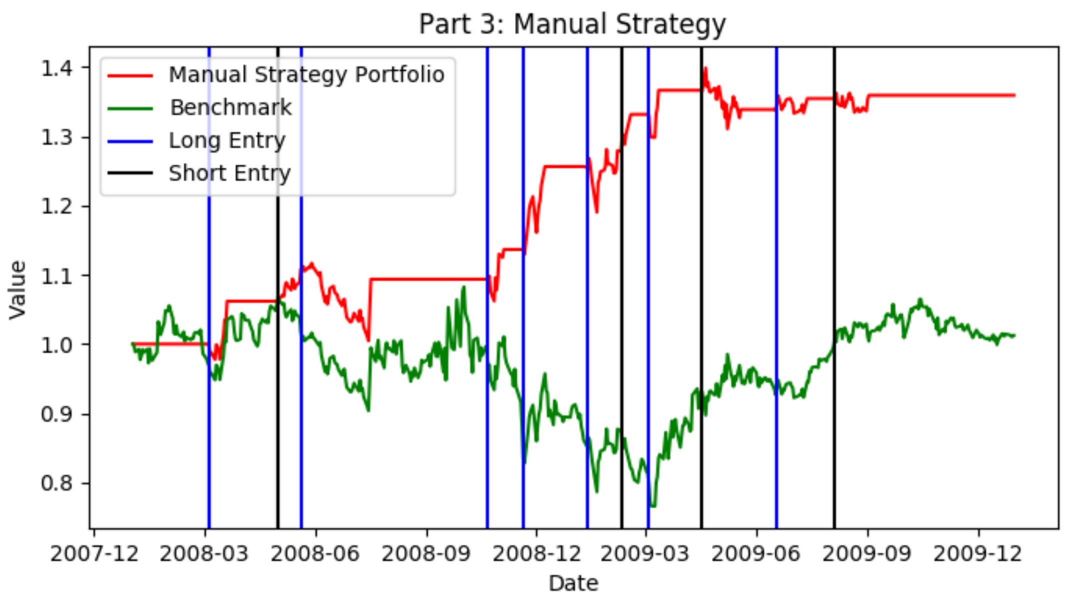

Figure 6: Line plot of benchmark (normalized to 1.0 at the start, green line) and the manual strategy portfolio (normalized to 1.0 at the start, red line) over the in-sample period. (Transaction costs: Commission $9.95, Impact 0.005)

4. Part 4: Comparative Analysis

4.1. Experimental Methods

Using the exact algorism described in **Part 3.1**, Except that when reading the stock price as a dataframe by the '`get_data`' function from '`util.py`', using from January 1, 2010 to December 31, 2011 at the '`date_range`' argument.

4.2. Result and Discussion

Figure 7 demonstrates the performance of the benchmark and the manual strategy in the out-of-sample period. The performance of the stock and the manual strategy for both in-sample and out-of-sample periods is summarized in **Table 1**.

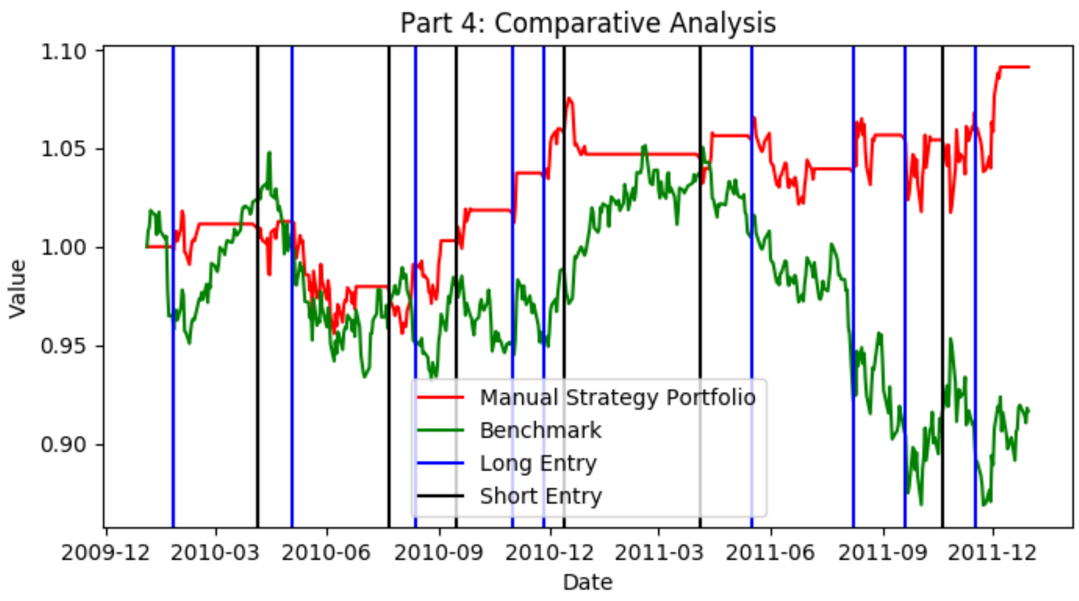

Figure 7: Line plot of benchmark (normalized to 1.0 at the start, green line) and the manual strategy portfolio (normalized to 1.0 at the start, red line) over the out-of-sample period. (Transaction costs: Commission $9.95, Impact 0.005)

Table 1. Performance of the stock benchmark ('JPM') and the manual strategy portfolio during in-sample and out-of-sample periods. (Transaction costs: Commission $9.95, Impact 0.005)

	Cumulative Return		Standard deviation of daily return		Mean of daily return	
	In-sample	Out-of-sample	In-sample	Out-of-sample	In-sample	Out-of-sample
Benchmark	0.0123	-0.0836	0.0170	0.0085	0.00017	-0.00014
Manual Strategy	0.3594	0.0913	0.0081	0.0058	0.00064	0.00019

Table 1 indicates that the manual strategy's performance during in-sample period is better than it during out-of-sample period in all three measures, i,e, cumulative return (0.3973 v.s. 0.1502), standard deviation of daily return (0.0080 v.s. 0.0057), and mean of daily return (0.00070 v.s. 0.00029). This is reasonable, as the manual strategy is designed (i.e. having the strategy 'tweaked') to obtain high performance during the in-sample period. Thus, the manual strategy performs worse during the out-of-sample period, due to potential overfitting to the in-sample data.

Additionally, the pattern of the stock price movement during the in-sample and out-of-sample periods are different. Considering that the 2008-2009 stock market crash happened during the in-sample period. This observation suggests that expletively designing trading strategy based on limited sample will not necessarily grant best out-of-sample performance. It is noticeable that the standard deviation during the in-sample period is higher than their out-of-sample counterparts, which is likely also due to the large price movements during the 2008-2009 stock market crash.

Table 1 also suggests that the manual strategy's performance during both in-sample and out-of-sample period is better (i.e. lower standard deviation and higher return) than the stock benchmark. This indicates that although the manual strategy could be overfitted, it still provides helpful trading decisions, and thus is an effective strategy.