MC3 - P3: CS7646 Machine Learning for Trading

Saad Khan (skhan315@gatech.edu)

November 28, 2016

Introduction

The purpose of this project report is to use Technical Analysis and develop (i) manual rule-based and (ii) machine learning based trading strategies by creating market orders. These orders will then be tested using a market simulator to analyze the performance of a particular trading strategy.

NOTE

1. Graphs used in the report are expanded versions of the Python popup window plots. These were visually easy to understand so I used them, however, output plots of the python programs will be saved in the matplotlib format.

2. Assumptions for this project are included in the readme file. For indicator computation: It is assumed that price dataframe is already extracted using get_data().

Part 1: Technical Indicators

Technical indicators are metrics that are derived from generic stock information such as price and volume. These indicators can predict future price levels or general price direction. The indicators that I have chosen for this part of the report are (i) Price to SMA ratio, (ii) Momentum and (iii) Relative Strength Index.

Price to Simple Moving Average (SMA) Ratio

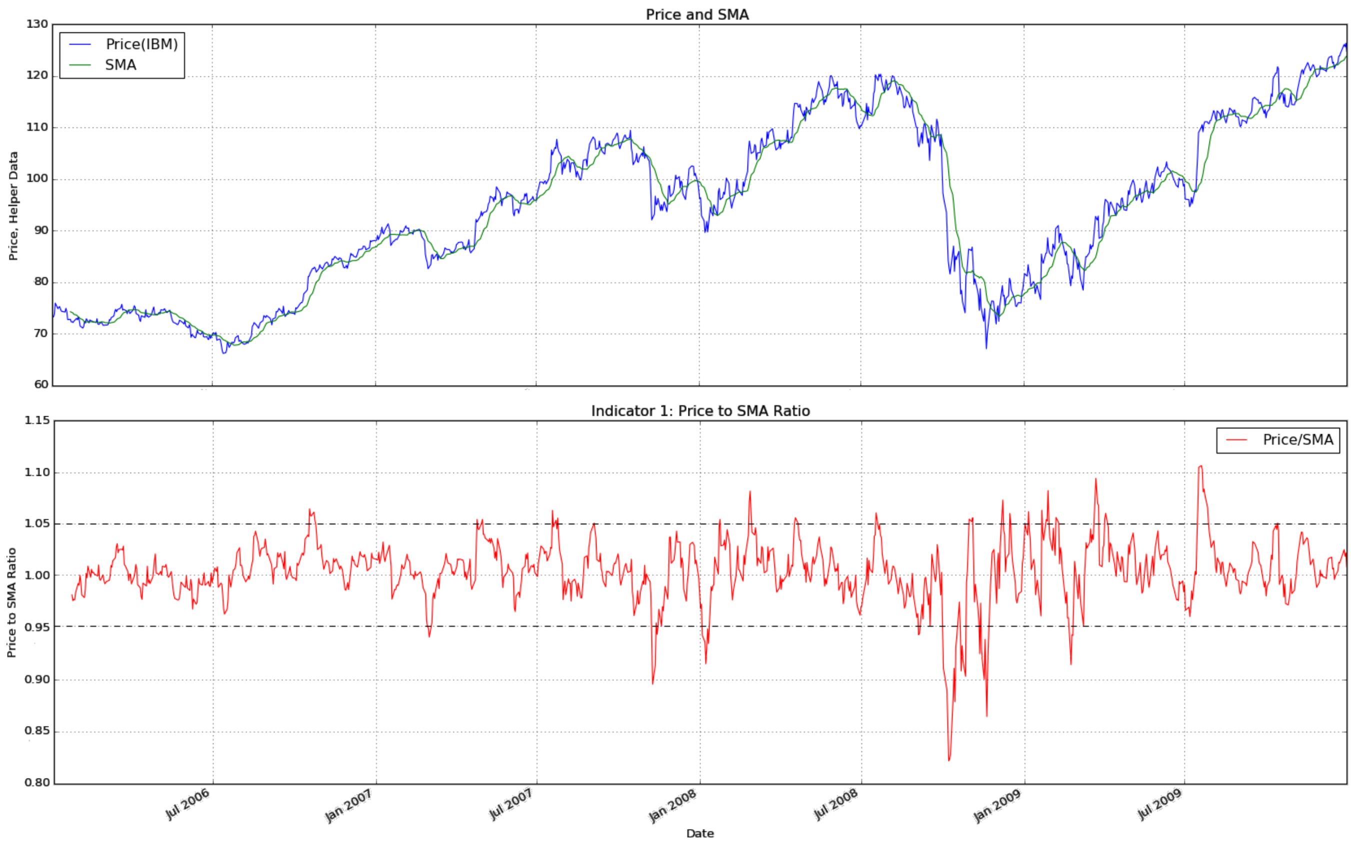

Figure 1: [TOP] : Price and SMA (Helper Data), [BOTTOM] : Indicator 1 - Price to SMA Ratio

Overview : Price/SMA is the adjusted close price of the stock at day [t] divided by the SMA. In this case where N = 14, Price/SMA today will be today's adjust close price of the stock divided by SMA of the past 14 days. One of the uses of this ratio for a strategy can be to determine when to close a position by classifying in 3 ways, if the ratio is greater than 1 then we are above the SMA, equals 1 indicates we are at the SMA and less than 1 indicates being below the SMA (David Byrd used for the strategy explained in the class).

Chart : Figure 1[TOP] shows the chart as per Dr. Balch's comments in piazza post **1184**, where stock price(Price[IBM]) and helper data (14 day SMA[IBM]) are plotted together while Price/SMA is plotted in Figure 1[BOTTOM] separately for ease of visualization. These charts were generated separately from [indicators.py], but combined in MS paint. The Price/SMA chart also includes 2 horizontal dashed lines at [1.05 and 0.95]. These are in conjunction with David Byrd's strategy explained in the class where Price/SMA dropping below 0.95 can be used to enter a long position while Price/SMA moving above 1.05 can be a signal to enter a short position.

Description : For Price/SMA, step by step procedure is mentioned below along with easy to follow pseudo code.

<table>
<tr><th>Steps</th><th>Description</th><th>Pseudo code</th></tr>
<tr><td>1</td><td>Create an array 'sma' that is same size and shape as the daily price array. This can be done using Pandas copy function.</td><td rowspan="6">
procedure PRICE/SMA
  $sma \leftarrow$ copy of $Price$
  for all days and symbols do assign 0's to sma
  for all days do Assign NaNs to sma for first N days
    for all symbols do
      for $i \leftarrow [today - N], [today]$ do
        $sum \leftarrow sum + Price(i)$
    $sma \leftarrow sum/N$
  $price\_sma \leftarrow$ copy of $Price$
  for all days and symbols do
    $price\_sma \leftarrow price/sma$
</td></tr>
<tr><td>2</td><td>Then initialize this array by using a nested FOR loop for each day and symbol and assign 0's everywhere.</td></tr>
<tr><td>3</td><td>Now populate this array, via a triple nested FOR loop by looping over all days, all symbols (just like we did in step 2) but now adding a 3rd loop for the lookback period from [today - N] to [today] while summing up the prices for each day.</td></tr>
<tr><td>4</td><td>Once done with the inner most loop for summing up the prices for last N days we simply divide this result by N itself (N = 14).</td></tr>
<tr><td>5</td><td>Now for finally computing Price/SMA, divide price for each day and symbol by respective SMA .</td></tr>
<tr><td>NOTE</td><td>As the first SMA value for each symbol is going to appear on the Nth day so an IF condition should be placed to assign NaN to SMA values for dates before the Nth day.</td></tr>
</table>

Momentum

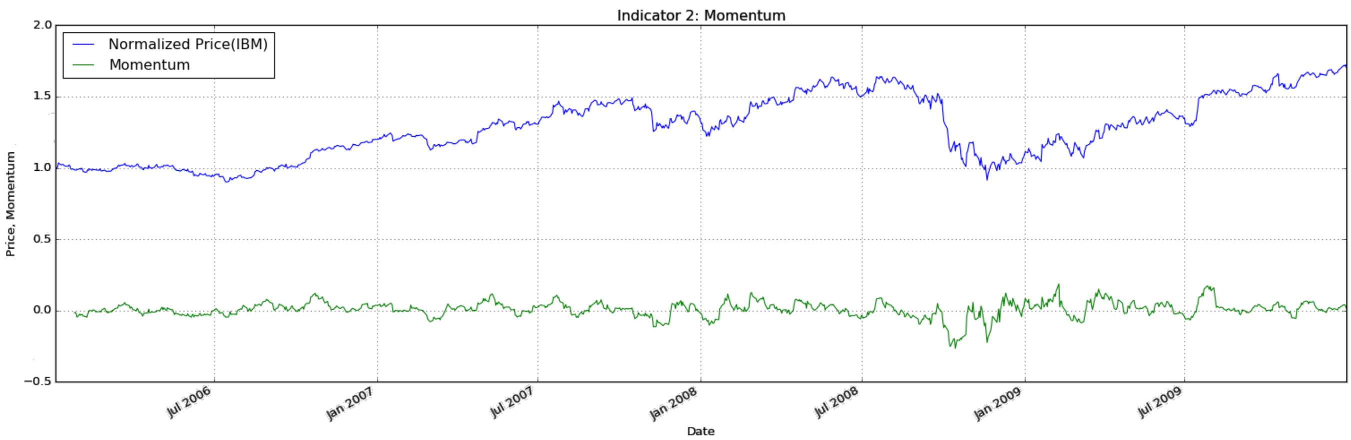

Figure 2: Indicator 2 - Normalized Price and Momentum

Overview : In technical analysis, momentum is considered as a means to help identify trend lines. A strategy based on momentum can take long or short positions hoping that the momentum will continue in the same direction after the position is entered. Momentum based investments utilize the trend created by the most recent price break such as past 10, 20 days and in this case it is based on last 14 days. This indicator was not presented in the class and was considered for this project as per the suggestions on the project wiki page.

Chart : Figure 2 shows the chart for Momentum with normalized price for the IBM stock (as per the piazza post **1345** for ease of visualization). It is clearly evident that momentum signifies the general direction of the stock, specially for the period at the end of 2008 when the market was not doing well.

Description : Momentum can be computed in a similar way like Price/SMA using the following steps. A pseudo code is also provided for reference.

Steps	Description	Pseudo code

1	You can start by creating a copy of the price array (lets call it 'momentum').	**procedure** MOMENTUM
		$momentum \leftarrow$ copy of $Price$
		for all days and symbols **do** assign 0's to $momentum$
		for all days **do** Assign NaNs to momentum for first N days
		for all symbols **do**
		for $i \leftarrow [today - N], [today]$ **do**
		$momentum \leftarrow (Price/Price(i)) - 1$

2	Then clear out this array by assigning 0's everywhere (just like previously done for Price/SMA)	
3	Now populate this array using a triple nested FOR loop for all days and all symbols (just like we did for Price/SMA) but now in the 3rd loop (inner most loop) you need to compute the momentum expression below (source: wikipedia). $rate\ of\ change = \frac{close_{today} - close_{N\ days\ ago}}{close_{N\ days\ ago}}$	
4	This can be done by first computing the ratio of price[today] to the price[today - N] days (N = 14) and subtracting 1.0 from this ratio for all days and symbols. This will give you the momentum indicator which will be used as part of the strategies later in this report.	
NOTE	*Momentum for each symbol is going to appear on the Nth day so an IF condition inside the outer most loop should be placed to assign NaNs.*	

Relative Strength Index (RSI)

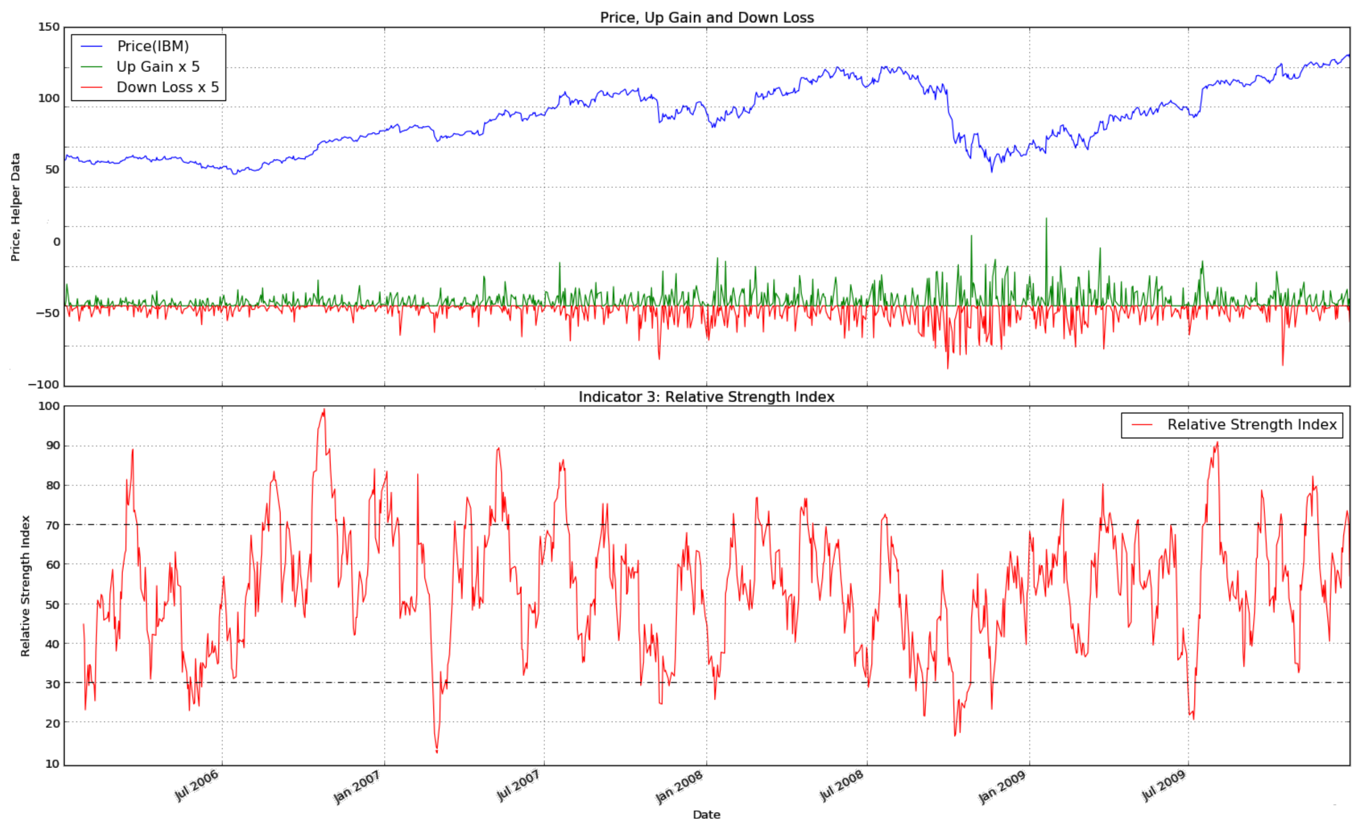

Figure 3: [TOP] : Price and Up Gain and Down Loss (Helper Data), [BOTTOM] : Indicator 3 - Relative Strength Index

Overview : Relative strength Indicator can be used as an oscillating indicator as it is predicting when the stock is reaching a high or a low. It is typically scored from 0 to 100 and stock is categorized as being overbought (RSI ¿ 70) or oversold (RSI ¡ 30). By definition it is on days when stock goes up how much does it go up and on days it goes down how much does it go down. RSI can be used for a divergence based strategy where the stock in consideration can be bought or sold in contrast to the RSI of the market (say S&P 500).

Chart : Similar to the Price/SMA chart, Figure 3[TOP] shows the stock price history and the helper data (14 day) up gain and down loss. Up gain and down loss are multiplied by 5 just for plotting purposes in order to better understand the trend. Figure 3[BOTTOM] shows the RSI values for the in-sample period with horizontal dashed lines at 70 and 30 showing the trigger points. RSI below 30 might indicate to initiate a long while above 70 might indicate entering a short position.

Description : RSI can be computed in the following way.

Steps	Description
1	Start by copying the price array and call it 'rsi'.
2	Then using a triple nested FOR loop with the outer most loop traversing all days and assigning NaNs to RSI values for days earlier than the lookback period (N = 14).
3	The second loop is run for all the symbols and initializes 2 variables, up_gain and down_loss to 0.
4	For the inner most loop from dates [today - N] to [today], maintain a variable 'delta' and assign the difference of today's price and yesterday's price to it. You will be doing this N times.
5	For a +ve delta, add delta to the variable up_gain and for -ve delta , subtract it from the variable down_loss.
6	Outside this loop put another IF-ELSE block inside the 2nd loop. If down_loss is 0 then RSI for that day and symbol is 100, otherwise (ELSE) assign a variable 'rs' = (up_gain / N) / (down_loss / N).
7	To compute RSI for each day, use the following formula (source: wikipedia) $$RSI = 100 - \frac{100}{1 + RS}$$
NOTE	*Again as rsi or each symbol is going to appear on the Nth day so an IF condition inside the outer most loop should be placed to assign NaNs.*

Pseudo code

```
procedure RSI
    rsi ← copy of Price
    for all days do Assign NaNs to rsi for first N days
        for all symbols do
            up_gain ← 0
            down_loss ← 0
            for i ← [today − N], [today] do
                delta ← Price − Price(i)
                if delta ≥ 0 then
                    up_gain ← up_gain + delta
                else
                    down_loss ← down_loss − delta
            if down_loss = 0 then
                rsi ← 100
            else
                rs ← (up_gain/N)/(down_loss/N)
                rsi ← 100 − (100/(1 + rs))
```

Part 2: Manual Rule-Based Trader

NOTE : Benchmark considered for part 2, 3 and 4 is the performance of a portfolio starting with $100,000 cash, then investing in 500 shares of IBM and holding that position for the entire period.

Overview : The rule based strategy was developed in conjunction with the piazza post **1357** and the indicators chosen were Momentum[IBM], RSI[IBM] and RSI[SPY]. The [rule_based.py] program that generates orders file for this part of the project extends the [indicators.py] program from the previous section and adds on an IF-ELSE conditions based 10 day holding strategy(description below).

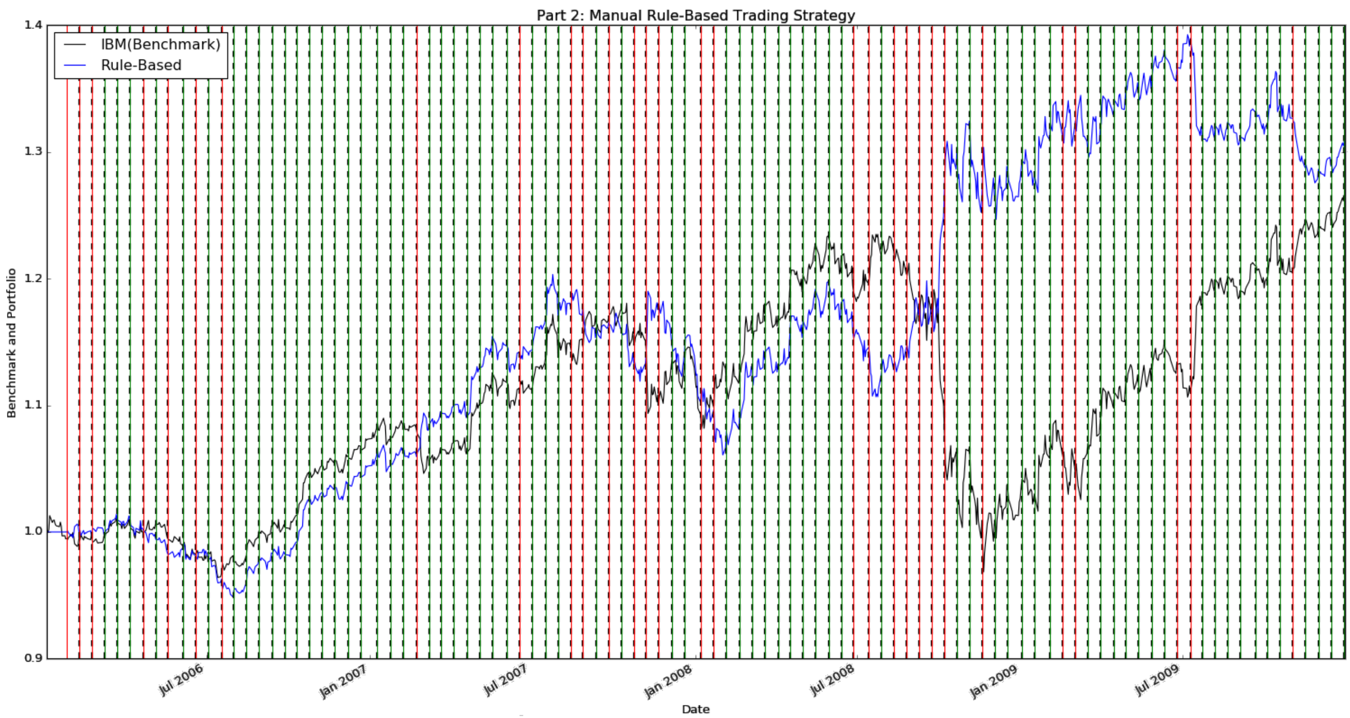

Figure 4: Manual Rule Based Trading Strategy - In-Sample period

Chart : Figure 4 shows the comparison of the bench mark and the rule based portfolio. Strategy implemented virtually trades every 10 days, i.e as soon as it exits a position, it enters a new one the same day. The exit positions are shown with dashed black lines as per the piazza post **1261** in order to clearly visualize same day exits and entries. The portfolio as a result of this strategy starts off very closely with the benchmark followed by drastic improvement in 2008. Although, the portfolio kept on improving from end of 2008 up till mid of 2009, it eventually declined towards the end of the in-sample period, however, still performed better than the benchmark.

Description : Moving further from the point where we have data frames for the indicators, following are the rules used for the manual strategy accompanied by a pseudo code.

Steps	Description
1	You can start by initializing 4 variables, 'holdings' , 'period10' , an empty 'orders' list and 'flag10' . 'Holdings' will maintain the current position [+500, -500, 0], 'period10' will check for the days that have passed after entering a position, 'orders' is a multi-dimensional array holding the BUY and SELL requests to be converted to an orders file while 'flag10' is a boolean check that is set to TRUE as soon a position is entered and is set to FALSE as it is exited.
2	After this, a double nested FOR loop should loop over all days and symbols (from day = N to last date of the in-sample period).
3	The inner loop should be set for 3 IF conditions, first 2 for the exit cases and last one for the entrance case. The exit IF conditions should check if the 10 day holding period (period10) has been reached along with holding of either +500 or -500 shares.
4	If the value of 'holdings' is +500 we exit the position by updating the 'holdings' = 0, setting 'flag10' to FALSE and updating the ' list with (SELL, 500). If the 'holdings' is -500 we execute a BUY by setting 'holdings' to 0, seting 'flag10' to FALSE and updating the 'orders' list with (BUY ,500)
5	The 3rd IF block should check for 'flag10' variable and should only be executed, if 'flag10' is FALSE otherwise we do nothing as this highlights the fact that we are in a state where we have entered a position and 10 days have not passed yet.
6	Once inside the the 3rd IF condition, check for the rules for entering either a long position or a short posiiton.
7	For entering a LONG position (BUY, 500) we check if rsi[IBM] is less than 30 and rsi[SPY] is greater than 30. If this critera is not met then we check if momentum[IBM] is greater than 0. If any of the criteria is met, we buy 500 shares of IBM and update 'holdings' to +500, set 'flag10' to TRUE, set 'period10' to the date 10 days from the current date and finally update 'orders' list with (BUY, 500).
8	For entering a SHORT position (SELL, 500) we check if rsi[IBM] is greater than 70 and rsi[SPY] is less than 70. If this critera is not met then we check if momentum[IBM] is less than 0. If any of the criteria is met, we sell 500 shares of IBM and update 'holdings' to -500, set 'flag10' to TRUE, set 'period10' to the date 10 days from the current date and finally update 'orders' list with (SELL, 500).
9	Once loop traversing is finished, 'orders' list will hold the sequential BUY and SELL orders.
NOTE	Care must be taken to setup a condition right after the rules evaluation to make sure whenever the trader enters a position, it should never have holdings other than [+500, 0, -500]

Pseudo code

procedure RULE BASED STRATEGY
    orders = [...] # empty list
    $holdings \leftarrow 0$, $period10 \leftarrow -1$, $flag10 \leftarrow False$
    for $i \leftarrow [Nthday], [End]$ **do**
        for all symbols **do**
            if $holding = +500$ **and** $i = period10$ **then**
                $orders \leftarrow (SELL, 500)$, $holdings \leftarrow 0$, $flag10 \leftarrow False$
            if $holding = -500$ **and** $i = period10$ **then**
                $orders \leftarrow (BUY, 500)$, $holdings \leftarrow 0$, $flag10 \leftarrow False$
            if $flag10 = False$ **then**
                if $(rsi[IBM] < 30$ **and** $rsi[SPY] > 30)$ **or** $momentum[IBM] > 0$ **then**
                    if $holding = 0$ **then**
                        $orders \leftarrow (BUY, 500)$, $holdings \leftarrow +500$, $flag10 \leftarrow True$, $period10 \leftarrow i + 10$
                else if $(rsi[IBM] > 70$ **and** $rsi[SPY] < 70)$ **or** $momentum[IBM] < 0$ **then**
                    if $holding = 0$ **then**
                        $orders \leftarrow (SELL, 500)$, $holdings \leftarrow -500$, $flag10 \leftarrow True$, $period10 \leftarrow i + 10$

Part 3: Machine Learning Based Trader

Overview : The first step was to convert the leaf condition for the [RTLearner.py] from mean() to mode() for classification. Secondly, in order to maximize the portfolio, the YBUY = -0.02 and YSELL = -0.022 threshold were set very close to each other so that we have very few 'DO NOTHING' classifications and maximize the portfolio.

Chart : Figure 5 shows the comparison of the benchmark and the 2 strategies. Trades occur virtually every day and the ML strategy out performs the benchmark and the rule-based strategy by a very big margin.

Description : The approach for the ML based system was 2-fold, (i) training the classifier and (ii) implementing the strategy. Figure 6 shows how training data was structured before supplying it to the Learner. The strategy developed for ML system to produce the orders file was very similar to the rule based system as the learning part was already completed by the learner, the decision making was based on the predicted results rather than thresholds for the indicators. Below Figure 5 is mentioned the approach along with the pseudo code that can help reproduce this.

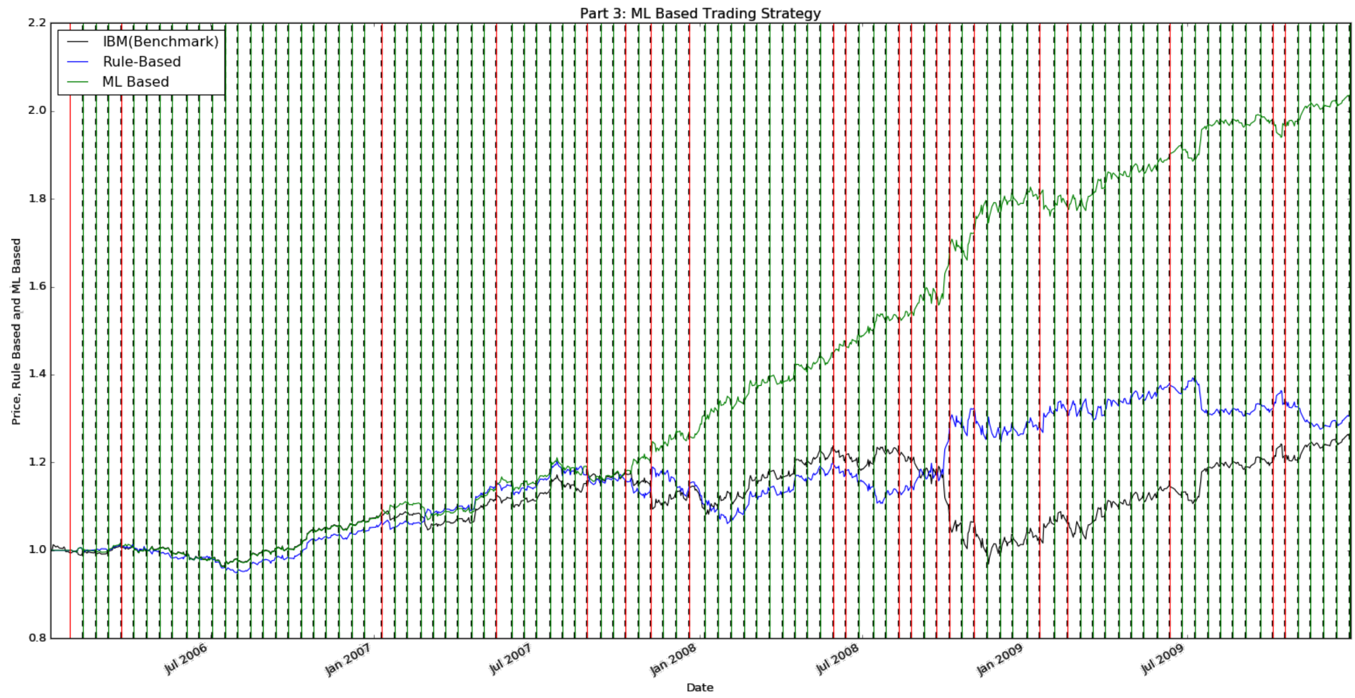

Figure 5: Machine Learning Based Trading Strategy - In-Sample period

Steps	Description
1	You can start by initializing the same 4 variables, 'holdings' , 'period10' , an empty 'orders' list and 'flag10' . In addition to this you will also need the prediction from the learner.
2	All the steps for generating the orders for the ML based appriach are exactly same as the rule based approach except the the entry IF condition for the stock, i.e. the 3rd IF condition (referring to rule based approach and as shown in the pseudo code below)
3	Once in the 3rd IF block, check for 'flag10' variable and only be execute if 'flag10' is FALSE otherwise we do nothing. When it is FALSE check for the
4	For entering a LONG position (BUY, 500) we check if prediction is BUY, i.e. +1 and there is no holding then we buy 500 shares and update 'holdings' to +500, set 'flag10' to TRUE, set 'period10' to the date 10 days from the current date and finally update 'orders' list with (BUY, 500).
5	We do the same for entering a SHORT position (SELL, 500) by checking if prediction is -1 and we don't hold 500 shares, then we update 'holdings' to 0, set 'flag10' to TRUE, set 'period10' to the date 10 days from the current date and finally update 'orders' list with (SELL, 500).
6	Once loop traversing is finished, 'orders' list will hold the sequential BUY and SELL orders.
NOTE	Again, care must be taken to setup a condition right after the rules evaluation to make sure whenever the trader enters a position, it should never have holdings other than [+500, 0, -500].

Pseudo code

```
procedure ML Based Strategy
    orders = [...] # empty list
    holdings ← 0, period10 ← −1, flag10 ← False
    for i ← [N^th day], [Last Date] do
        for all symbols do
            if holding = +500 and i = period10 then
                orders ← (SELL, 500), holdings ← 0, flag10 ← False
            if holding = −500 and i = period10 then
                orders ← (BUY, 500), holdings ← 0, flag10 ← False
            if flag10 = False then
                if predY(i) = +1 andholdings = 0 then
                    if holding = 0 then
                        orders ← (BUY, 500), holdings ← +500, flag10 ← True, period10 ← i + 10
                else if predY(i) = −1 andholdings = 0 then
                    if holding = 0 then
                        orders ← (SELL, 500), holdings ← −500, flag10 ← True, period10 ← i + 10
```

Part 4: Comparative Analysis

NOTE : Benchmark considered for this part is the performance of a portfolio starting with $100,000 cash, then investing in 500 shares of IBM and holding that position for the entire period.

Chart : Figure 7 shows the comparative trends of the bench mark, the rule based portfolio and the ML based portfolio for the out of sample period. It can clearly be seen that the ML based strategy has performed very well against the unknown data with the final portfolio value of more than $13,000 compared to the benchmark. On the other hand, rule based strategy

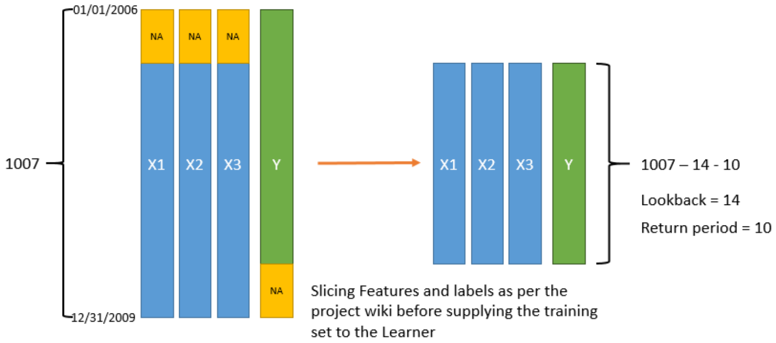

Figure 6: Training Set for the Learner

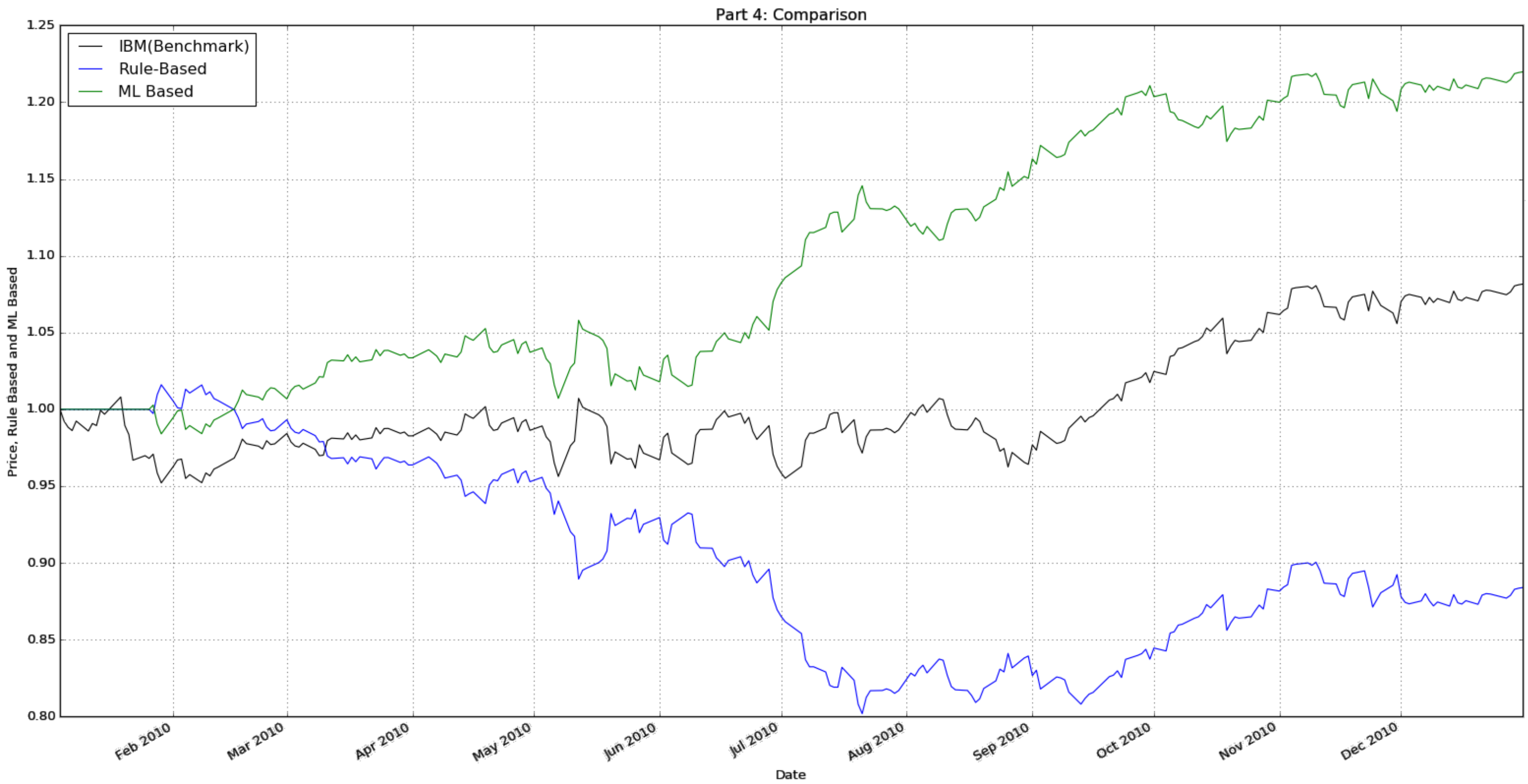

Figure 7: Comparative Analysis - Out of Sample period

fared pretty poorly against the unknown data loosing almost $20,000 at the end of the out of sample period compared to the benchmark.

Poor Performance of the Manual Strategy vs. Machine Learning Strategy: The main reason, I can think of regarding the poor performance of the manual strategy is that it heavily relied on divergence from the market (SPY). With indicators such as RSI[IBM] and RSI[SPY] and diverging rules, rule based approach did favor during the in-sample period as shown in Figure 8[LEFT] where towards the end IBM and SPY diverge (including SPY in plots as well - just for illustration purposes). Although, rule based approach does take advantage of the divergence but not that much compared to ML approach. In contrast if we look at Figure 8[RIGHT] for the out of sample period, rule based approach never kicks in as IBM and SPY perform almost the same throughout 2010. In fact, the diverging nature of the rules constructed, work against the portfolio for the 1st half of 2010 as SPY performs better than IBM. This caused the value of the portfolio to go down and it never recovered back from there, where as, random tree learner does not get stuck and takes advantages of both RSI[IBM] and RSI[SPY], beating the benchmark by a significant margin for both in-sample and out of sample periods.

Overall Performance Comparison :
Figure 9 tabulates different performance factors for the in-sample and out of sample orders. It can clearly be seen, in conjunction with the plots above, that ML based approach was the best performer. In fact the cumulative return ratio of ML/benchmark was 4.0 which beats the rubric requirement of 1.5 by a considerable margin. Although, rule based approach beat the benchmark for the in-sample period, its poor performance against the unknown data is also evident here with cumulative return ratio of rule based/benchmark of -1.42. A more visually informative representation for Sharpe ratio, cumulative return and portfolio value is shown below in Figure 10, again highlighting the poor performance of rule based portfolio for the out of sample period.

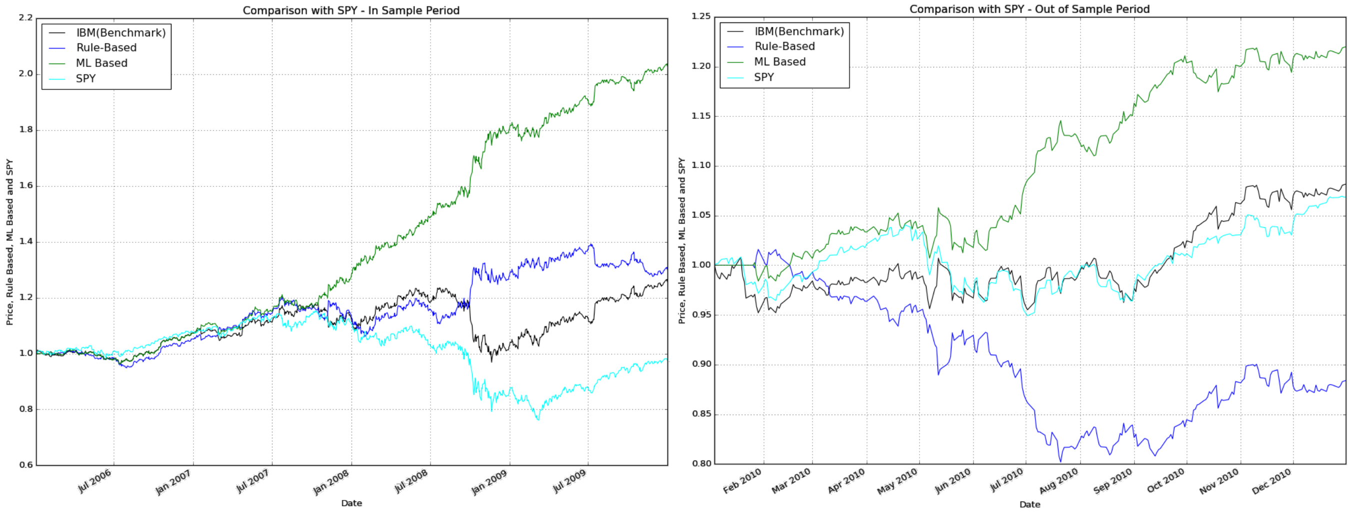

Figure 8: Comparative Analysis - [LEFT] : In Sample period, [RIGHT] : Out of Sample period

Sample Period	Strategy	Sharpe Ratio	Cumulative Return	Standard Deviation	Average Daily Return	Final Portfolio Value	C.R. Strategy / C.R. Benchmark
In Sample Period 01/01/2006 - 12/31/2009	*Benchmark*	0.58	0.26	0.0068	0.00025	125715	--
	Rule Based	0.72	0.30	0.0062	0.00028	130040	1.17
	ML Based	2.27	1.03	0.0050	0.00072	202860	4.00
Out of Sample Period 01/01/2006 - 12/31/2009	*Benchmark*	0.76	0.08	0.0070	0.00034	108155	--
	Rule Based	-0.98	-0.12	0.0075	-0.00046	88390	-1.42
	ML Based	2.11	0.22	0.0061	0.00081	121980	2.70

Figure 9: Comparative Analysis - [TOP] : In Sample period, [BOTTOM] : Out of Sample period

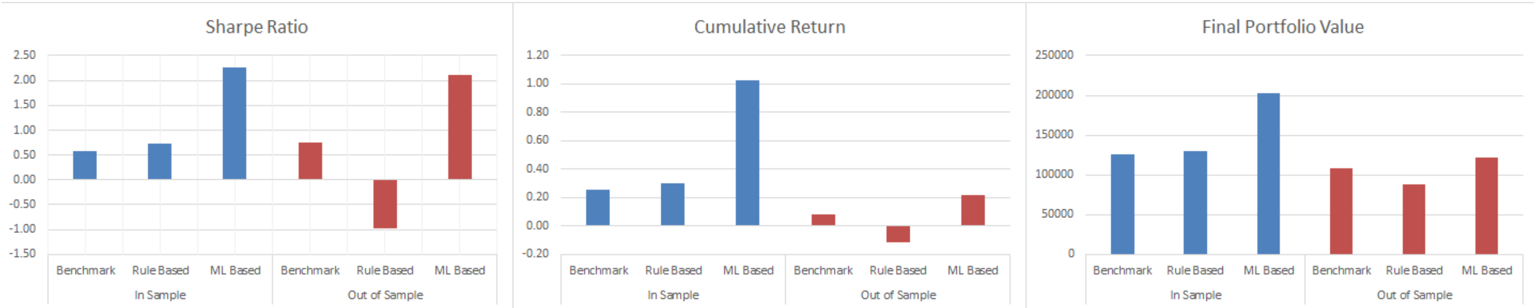

Figure 10: Comparative Analysis - [TOP] : In Sample period, [BOTTOM] : Out of Sample period

Conclusion

This project was extremely challenging considering the amount of time given to handle the tasks involved. Never the less, a great learning experience and a special thanks to some of the fellow course mates on piazza.