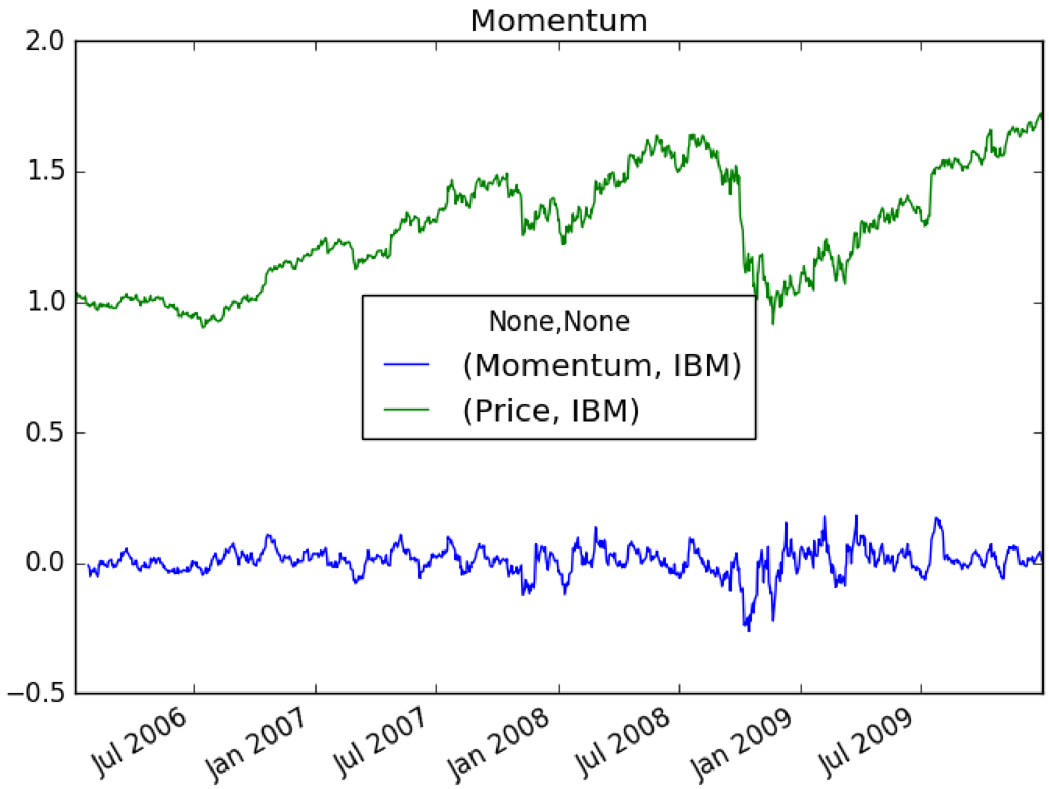
Part I: Technical Indicators

Momentum



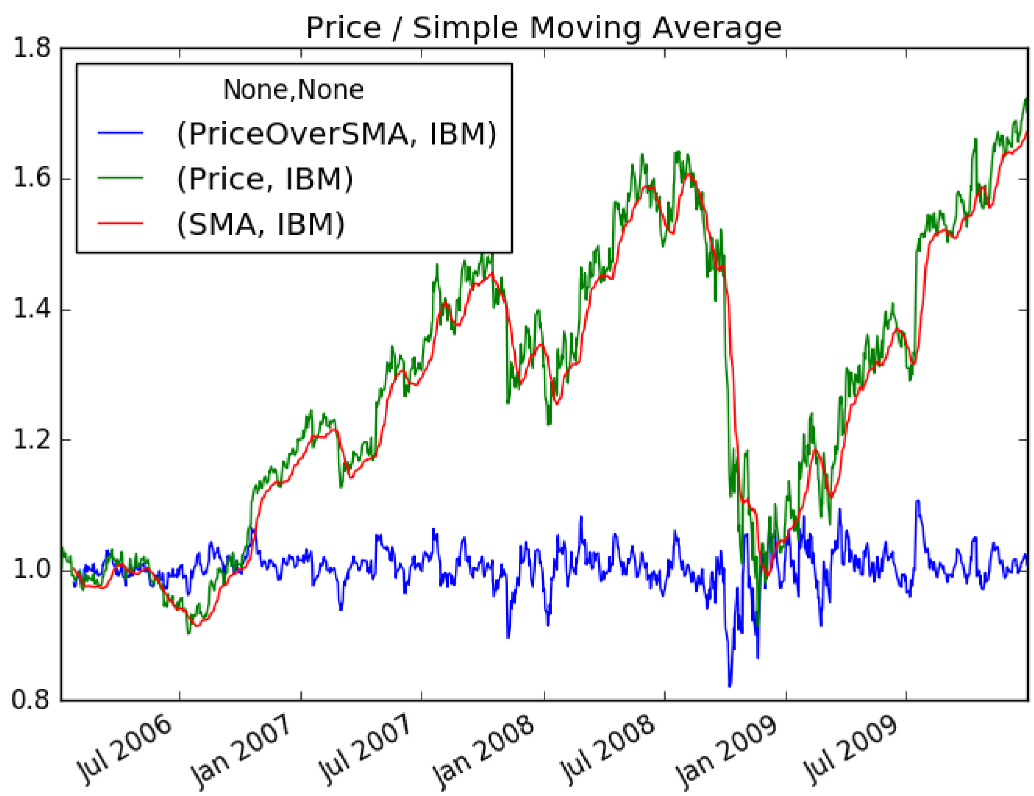
Overview:

Momentum is a measure of how much the stock has gone up or down in a fixed amount of past time. It is a useful indicator because we tend to expect prices to continue with momentum. In this case, we are looking back 14 days. Momentum can also be useful for identifying if we are in an uptrend or downtrend based on the value. A value greater than 0 indicates a general uptrend in the last 14 days and a value less than 0 indicates a general downtrend in the last 14 days.

Implementation Details:

All that's needed to calculate momentum is the history of prices and a determined lookback period (n). The momentum at time t can be calculated as: momentum[t] = (price[t]/price[t-n]) -1, where n = 14 days in our case.

Simple Moving Average (SMA)



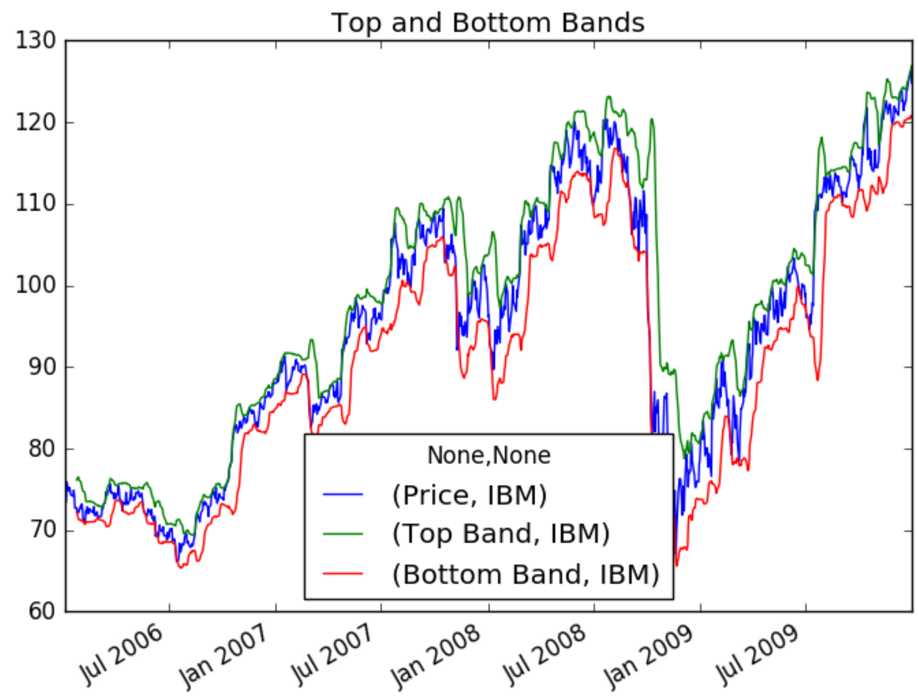
Overview:

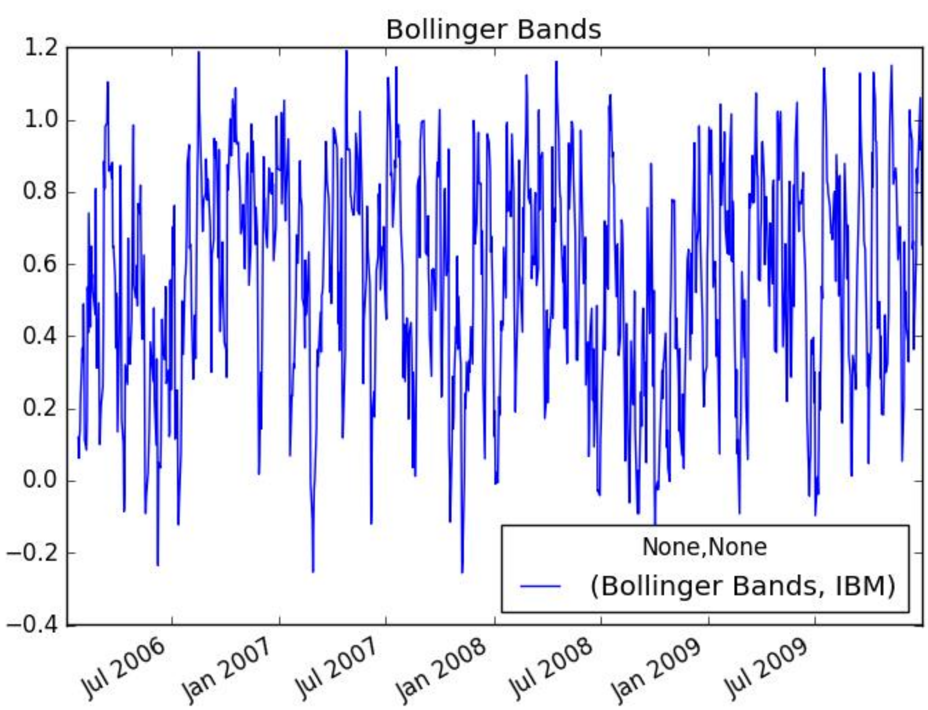
Simple moving average is essentially the mean of the price back n days. In this case we are looking back 14 days. By itself, SMA is a useful indicator in that it essentially smooths out the price values and can be viewed as a proxy for value. This especially useful when comparing the current price with the SMA value, since a current price well below the SMA value can indicate an arbitrage opportunity in favor of going long (and vice-versa). Even more useful is the ratio of price over SMA which makes it more obvious when the stock is over or undervalued. Our price/SMA ratio denotes that the stock is overvalued when it is greater than 1 and undervalued when it is less than 1.

Implementation Details:

Once again, all that's needed to calculate price/SMA ratio is the time history of price data and a lookback period. SMA is calculated as the mean of the price data back n days: price[t-n:t].mean(), where n is 14 in our case. Thus to calculate price/SMA, the full equation is: price[t]/price[t-n:t].mean().

Bollinger Bands





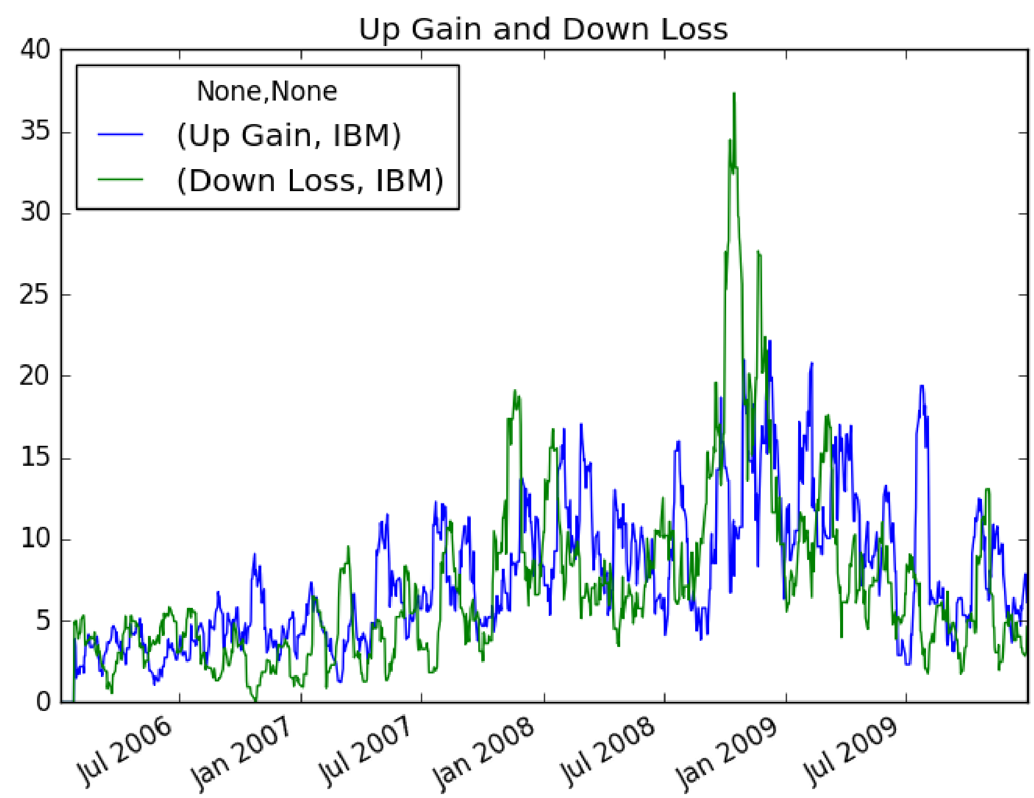
Overview:

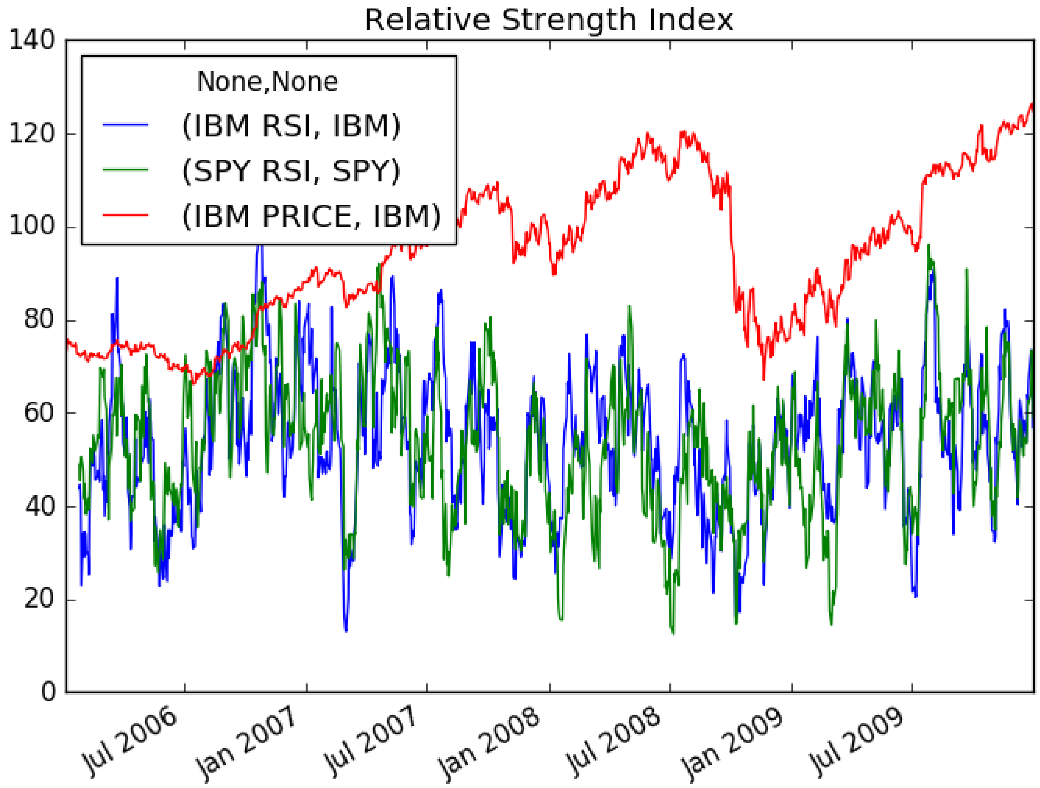
Bollinger Bands are a smarter way to trade on SMA since it takes the local volatility into account. The top chart shows the upper and lower bands which are essentially 2 standard deviation away from the SMA. The bottom chart shows the value of the Bollinger Band indicator when has a value greater than 1 when price goes over the top band (overbought) and less than 0 when it goes below the bottom band (oversold). This is a useful indicator since the price deviating from the SMA is less predictive when there is high volatility.

Implementation Details:

In order to get the normalized Bollinger band indicator you first need to get the top and bottom bands which are located 2 standard deviations away from the SMA. Note that the standard deviation is a rolling standard deviation with a lookback period of 14. We won't go into the specific implementation details of rolling standard deviation since it can be easily calculated with a pandas function. To be clear: topband[t] = SMA[t] + 2 * (std[t]) and bottomband[t] = SMA[t] – 2 * (std[t]). Finally the indicator we're after can be calculated as follows: bb[t] = (price[t] – bottomband[t]) / (topband[t] – bottomband[t]). Simple back of the envelope calculations confirm that a price less than bottom band produces a negative value and a price greater than the distance between the bands produces a value greater than 1.

Relative Strength Index (RSI)





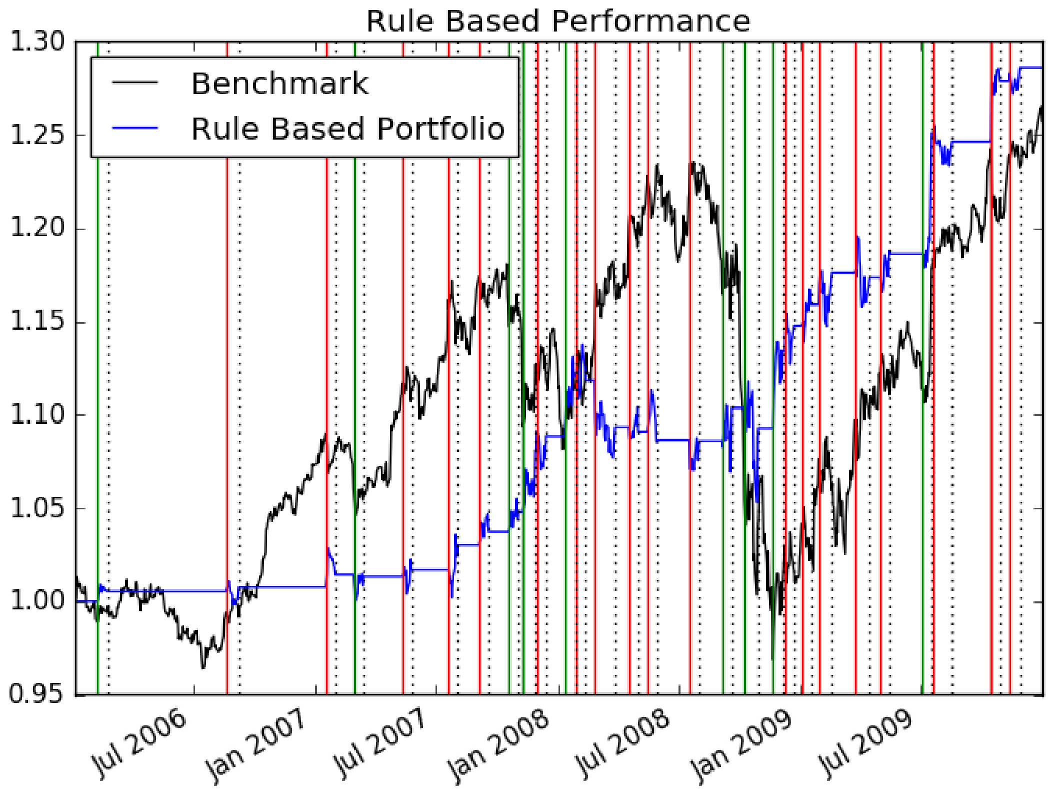
Overview:

Relative Strength Indicator is a momentum indicator useful for determining if a stock is overbought or oversold. It essentially looks at the ratio of the average gains over the average losses over a lookback period, which in this case is 14 days (both illustrated in the top chart). This ratio is normalized between zero and 100 where RSI = 0 indicates no gains and RSI =100 indicates 0 losses. Generally an RSI over 70 is overbought and under 30 is oversold. Note that we have included the RSI value for SPY since that is used to get a measure of whether the market is over or underbought.

Implementation Details:

This indicator is by far the most rigorous to calculate, but fairly simple none the less. First, we calculate the daily returns for each day of the trading period where the price of the current day minus the previous day is each day's daily return. Then we get the up gain and down loss for each day by separately summing the positive and negative returns for the past 14 days. Then the relative strength is the ratio of the average up gain over down loss for each day: rs = (up_gain / 14) / (down_loss / 14). Finally, RSI is normalize by: RSI[t] = 100 – (100/(1+rs[t])). Be wary of divide by 0 when the downloss is 0.

Part 2: Manual Rule-Based Trader



Overview:

The overall strategy of the manual trading rules is buy when the stock appears to be oversold relative to the market and to sell when the stock appears to be overbought relative to the market. If our indicators are correctly identifying that stock is oversold or undervalued, then it makes sense to buy since the market will eventually correct and the value will go up. The capital assets pricing model states that the stocks are completely controlled by the market so there should be no residual return. The basis of this class is to actively trade, attempting to disprove that theory by finding residual returns that are unique to a particular security relative to the market, also known as alpha. To this end we use a number of indicators to determine if the IBM stock is overbought or undersold. A low price/SMA ratio, low Bollinger Band percentage, and low RSI value all indicate that the stock is oversold. In addition, a low momentum value indicates that the stock is also trending downward at the time, making it even more likely that the stock is oversold at that time and will continue in that direction. If at the same time we see high RSI value for the market indicating that the market as a whole is not oversold, then we can assume that there is a residual value for the stock that can be capitalized on by buying the stock. Unsurprisingly, the reverse logic can be used to find selling or shorting opportunities. These divergences don't last forever however, so when the SMA crosses over the price, then we exit our position since we no longer have any indication of the stock being sufficiently under or oversold.

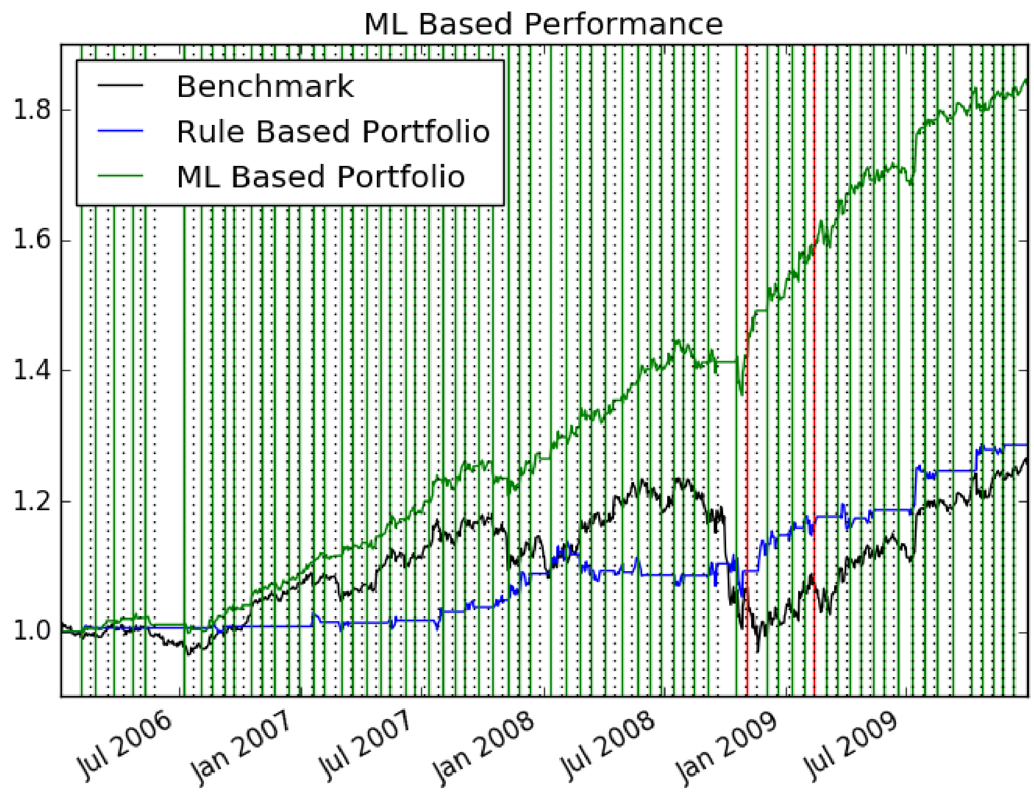
Implementation Details:

The strategy can be summed up as follows:

- If
  - The price/SMA ratio is less than 1
  - The Bollinger bands value is less than 0.2
  - The RSI is less than 30
  - The momentum is less than -0.048
  - The SPY RSI is GREATER than 25
- Then
  - The stock is likely oversold and there is a likely a divergence from the market so go LONG
- Else if
  - The price/SMA ration is greater than 1
  - The Bollinger bands value is greater than 0.9
  - The RSI value is greater than 50
  - The momentum is greater than 0.052
  - The SPY RSI is LESS than 70
- Then
  - The stock is likely overbought and there is likely a divergence from the market so go SHORT
- Exit positions when price crosses SMA
- Always hold long or short positions for 10 days to coincide with project requirements

  The exact values set for the indicators were tweaked to optimize performance on the test set and in general, relaxing the requirements for buying or selling resulted in more trades as well as better performance up to a limit. This is why I call out that divergence from market is only likely but not guaranteed during buys and sells. For instance there could be cases where The RSI for the stock is 27 and the RSI for the market is 27, meeting both the requirements for going long but without a divergence in the stock and the market.

Part 3: ML Trader



Overview:

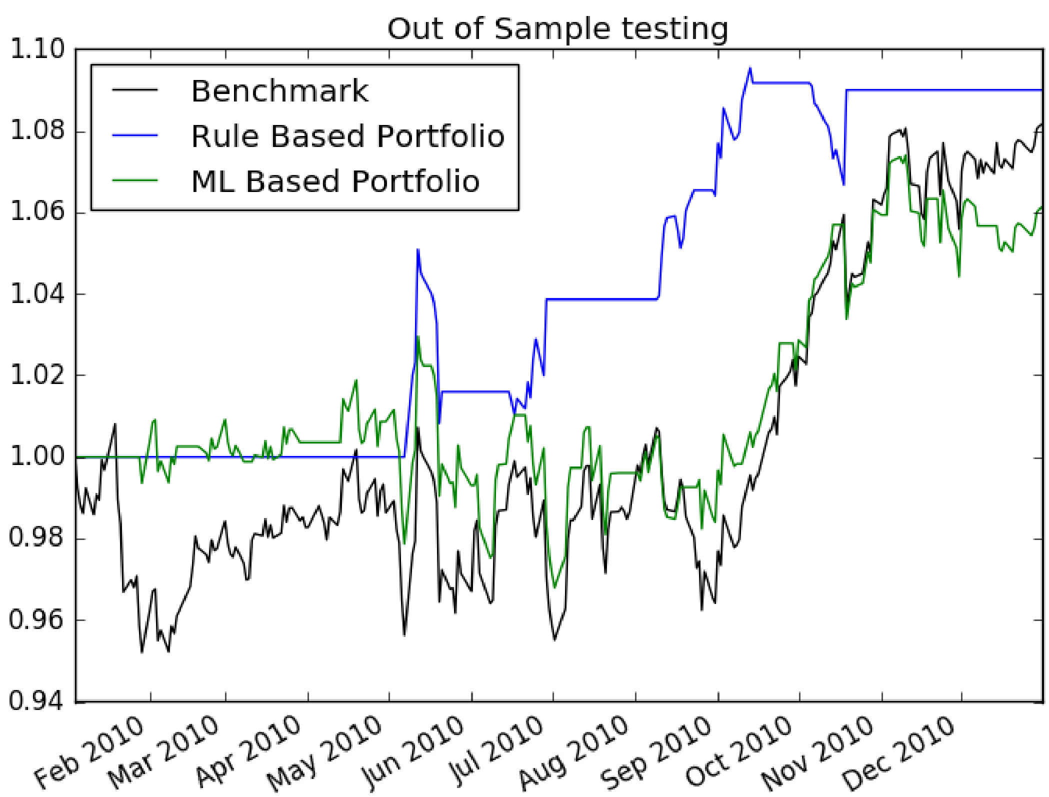
The ML approach utilizes a random tree learner with classification in order to determine if the portfolio should go long, short, or exit on any particular day. The x-training data for the learner is the indicators for each day and the y training data is based on the 10 day future return for each day. Days that have a sufficiently high future return are given a value of 1 for buy and sufficiently low returns are given a value of -1 for selling. The random tree learner is run with this x and y training data and a leaf size of 5 with no bagging. Then the category for days in the training set was predicted using the learner with the x training data from the indicators. Note that the values returned from the random tree learner for predicted Y values were generally not whole number since the leaf size of 5 would average category values. Therefore, values that were greater than .5 were rounded to 1 and less than -0.5 were rounded to -1 for a sale. This allowed for categorization learning. It should be noted that the values of leaf size and bagging were tweaked, however a small leaf size and no bagging resulted in the best results over the training data set, most likely due to over fitting.

Implementation Details:

- Get the x training data from the indicators over the training dates
- Get the x testing data from the indicators over the training dates
- Get the y training data from the 10 day returns for each day over the training dates
    - If the return is greater than 1.04 then set y to buy or 1

- If the return is less than .95 then set the y to sell or -1
- Train the learner on the x and y training data with leaf size of 5
- Get predicted y values by querying trainer with x testing data
- Convert predicted y values to buy or sell values by rounding up from .5 to 1 and down from -.5 to -1. Everything else is set to 0.
- Ensure that orders are held for at least 10 days before exiting position
- Then diff the orders so that there is only an order when target shares change.

Part 4: Comparative Analysis



In-Sample			
	Stock	**Manual**	**ML**
Cumulative Return	0.257	0.285	0.838
Sharpe Ratio	0.584	0.93	2.23
Out-Sample			
	Stock	**Manual**	**ML**
Cumulative Return	0.081	0.0899	0.199
Sharpe Ratio	0.76	1.44	2.31

Discussion

Summary of results:

While the manual approach still outperformed the benchmark in out of sample testing, the ML approach not only dropped significantly in performance from its in sample testing, it even fell short of baseline.

Why performance is worse out of sample:

The manual approach performed comparably in and out of sample. The ML approach was optimized to work best on the training data. Clearly it had overfit on the training data even with the leaf size of 5 and therefore struggled when dealing with the out of sample tests.

Why manual and ML strategies perform substantially differently:

Once again, the major drop in performance for the ML approach is due to overfitting during training.

Why ML strategy is more susceptible to overfitting:

I could only overfit the manual approach so much because I optimized the parameters by hand and the core strategy was based on the logical operation of market forces related to under and over-valued stocks. The ML strategy was only tweaked in order to get maximum performance on the training data set which in some sense incentivized an approach that was overfit to the training data with smaller leaf sizes and without bagging to introduce randomness.