Part 1: Technical Indicators

In this part, I use 3 different indicators: momentum, Bollinger, and RSI (Culter's version).

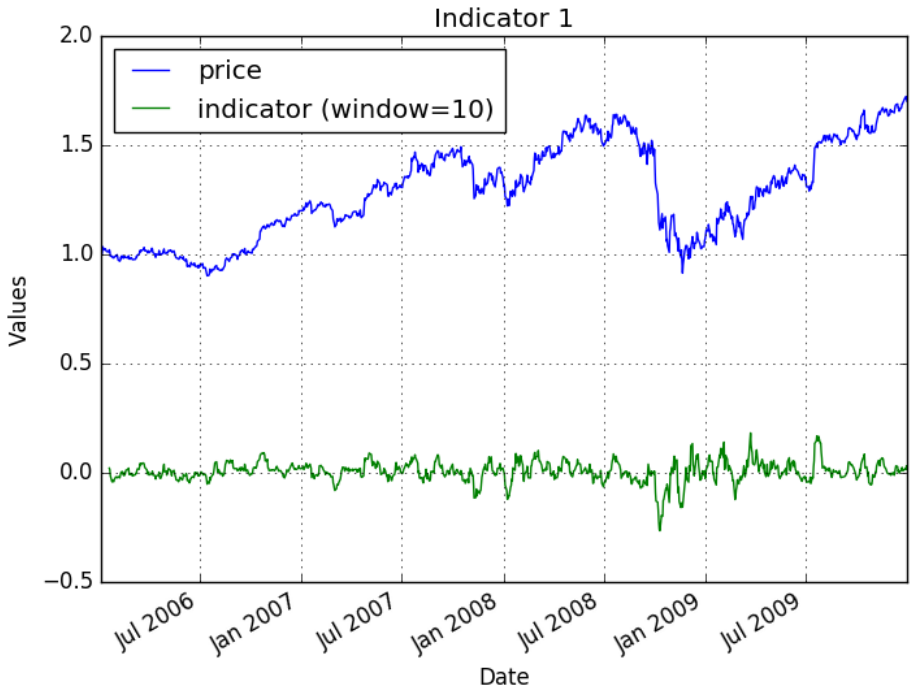
1. momentum:

Momentum shows the relationship between a value and a corresponding previous value. When calculating the momentum, we should firstly calculate the ratio of current value and the value N days ago, and then decrease this ratio by 1.

That is, the momentum of time t can be calculated by: momentum[t] = (price[t]/price[t-N]) - 1, where N is the amount of days you want to look back.

In my code, I assign the N to be 10, which means we will divide the price at each day by the one 10 days ago and then minus the quotient for each day by 1.

The chart is shown as follows:



2. Bollinger

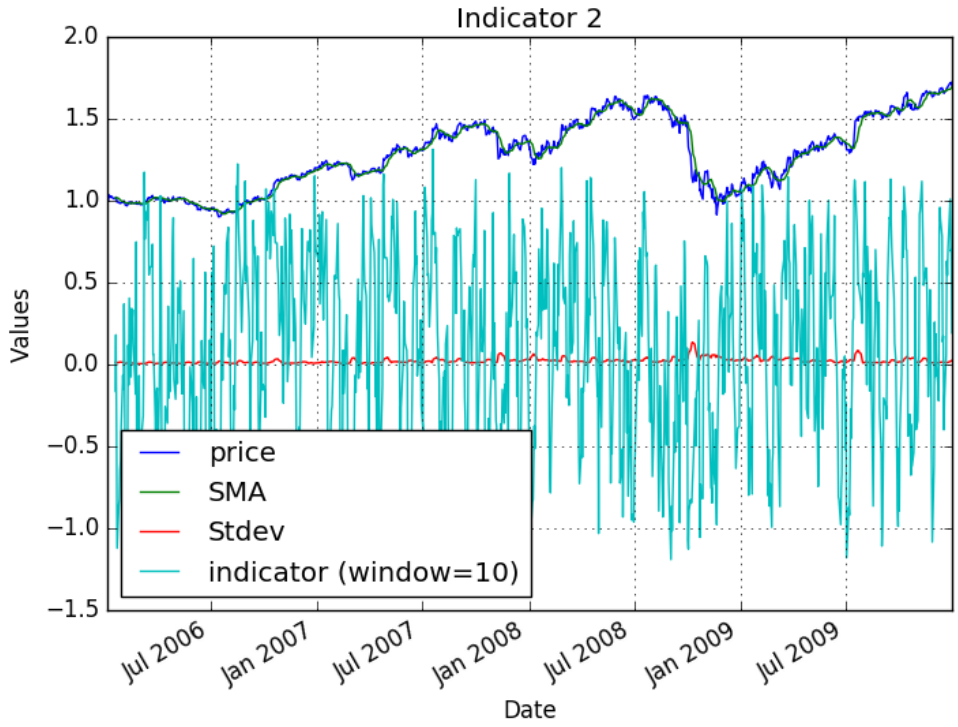
Bollinger shows the relative level of price, which can help us know wether the price is too high or too low.

When calculating the Bollinger value, I will normalize the daily price, and then calculate the rolling mean (SMA) and rolling standard deviation with the window size being 10

days. Then the Bollinger value for each day can be calculated: firstly calculate the difference of the current price and the rolling mean (SMA), and then divide it by 2 and rolling standard deviation.

That is, bb_value[t] = (price[t] - SMA[t])/(2 * stdev[t]). Here the SMA is the rolling mean of price with window = 10 days, and stdev is the rolling standard deviation with window = 10 days.

The chart is shown as follows:



3. RSI

RSI is the Relative Strength Index. However, there are many versions of RSI, and for this project I selected the Culver's version.

When calculating the RSI for one day, I will calculate how much does the price goes up (if the price didn't rise it will be 0) for each day in the past 9 days and today, as well as how much does the price goes down (if the price didn't go down it will be 0) for each in the same period.

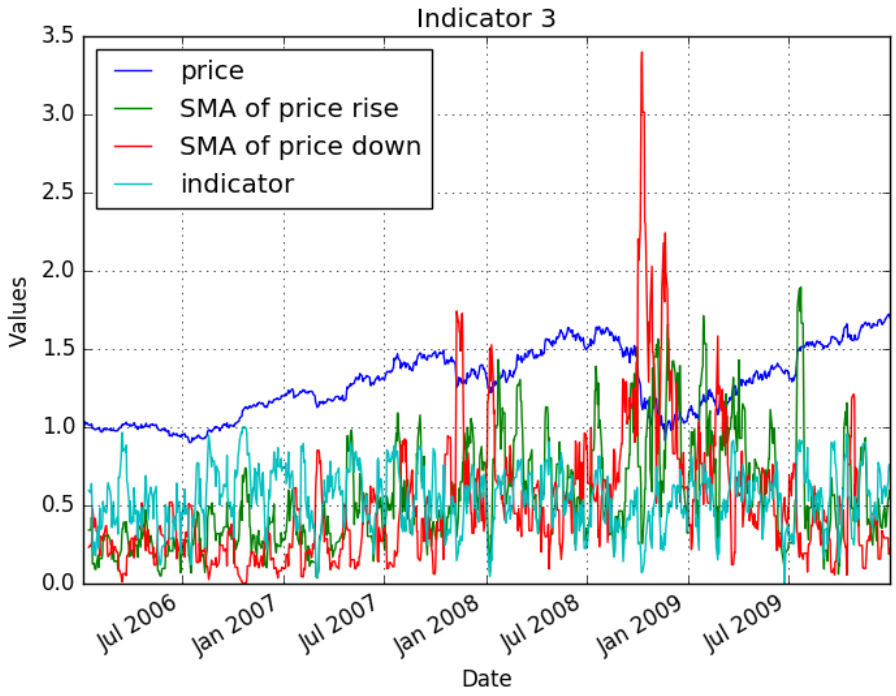
*/*for example, assume the prices we have are: 1, 3, 1, 6, 4 …Then the price increases should be 2,0,5,0,… and the price decreases should be 0,2,0,2,….*/*

Then I will calculate daily average increase for the period and the daily average decrease for the period by:

average daily increase = SUM(all the price increase in the period)/total days

average daily decrease = SUM(all the price decrease in the period)/total days
And the Cutler's RSI can be calculated by:
RSI= average daily increase / (average daily increase + average daily decrease)

The chart is shown as follows:



Part 2: Manual Rule-Based Trader

Here's my annual trading strategy: (some of the numbers below are actually the parameters of the trading strategy) as is described before, each day I have three indicators: momentum, Bollinger, RSI.

If RSI>0.7 (a parameter): SELL
If RSI<0.1 (a parameter): BUY

-> If we don't sell or buy(if 0.1<=RSI<=0.7): look at Bollinger
If Bollinger > 0.725 (a parameter): SELL
If Bollinger < -0.725 (a parameter): BUY

-> If we still don't sell or buy(if 0.1<=RSI<=0.7 && -0.725<=Bb<=0.725): momentum
If momentum >0.016 (a parameter): BUY
If momentum <-0.02 (a parameter): SELL

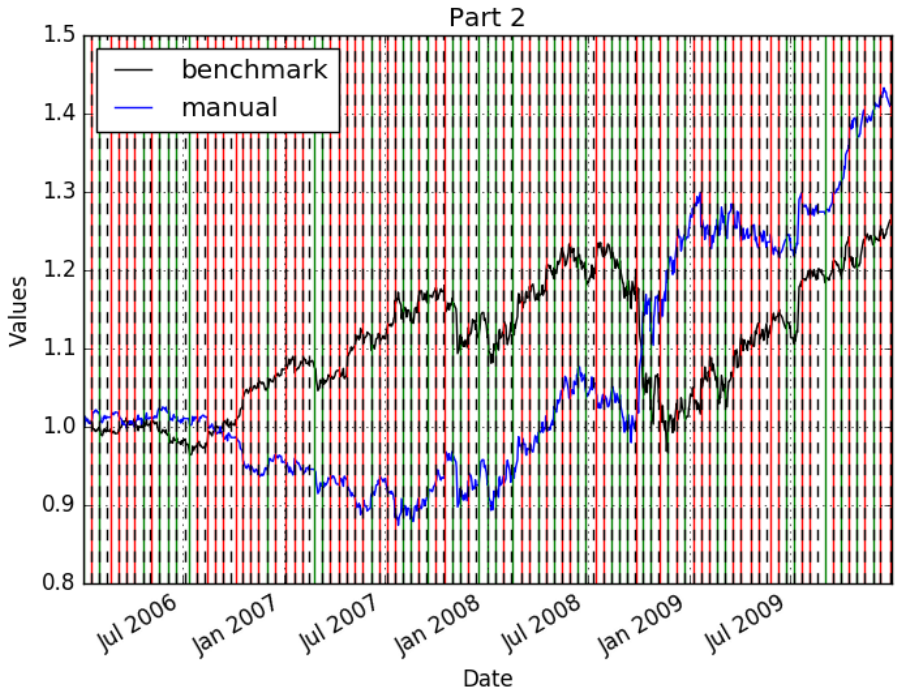
Firstly, I will check the RSI, which reflects the strength of the market. If RSI>0.7 then I would take a short position, because a very high RSI indicates the over-buy may be severe and the price would probably goes down later; if RSI<0.1 then I will take a long position, because a very low RSI indicates the over-sell may be severe and the price would probably goes up later.

If the RSI doesn't let us buy or sell, I will look at the Bollinger value. If Bollinger value > 0.725, I will short the stock because this indicates that the current price is much higher than the average price for the past few days, and it may drop down later since it's too high; If Bollinger value<-0.725, I will long(buy) the stock, since the price is much lower than the average and may rise up soon.

If the Bollinger value still doesn't let me buy or sell, I will check the momentum value. Since both the RSI and Bollinger didn't let me buy or sell, this means the price is neither too high than what it should be nor too low, and I will buy it if I think it is rising, and short it if it is going down. If momentum > 0.016 I will buy the stock, because this means the price will probably rise since it is not too high than what it should be; if the momentum < -0.02 I will short the stock, because this means the price will probably goes down.

Finally I will get the suggestions from my manual strategy. After modifying it to suit the "10-day" rule posed by this project, I can have the finally trading orders.

The chart is shown as follows:



for the in-sample period, the strategy can provide a cumulative return of `0.4094`, while the benchmark is giving `0.2572`

Part 3: ML Trader

1. brief introduction of previous regression learner

Here's the brief introduction of my previous RT learner and Bag learner, which are finally modified to be classification learner for this problem.

The random tree and bag learner are the ones from MC3_P1. When training a random tree I will let the leaves store the **mean** value of the data for it, which means leave.value=avg(Y1,Y2,…,Yn) where Ys are the labels of all the data which the leave will represent.

When building a random forest, I will train many different random trees based on the training set. When making a query, the result will be the average of the results from every random tree.

2. modify the **regression** Random Tree and Random Forest to **classification** Random Tree and Random Forest

What I did is actually change the **mean** of the Ys to be **mode** of the Ys.

When building a random tree, the value for each leaf will be the class which most of the data for this leaf belongs to. That is, if most of the data for this leaf belongs to class 1, then the leaf will belong to class1.

When using the Bag learner (random forest) to deal with a query, I will make many queries by using each tree, and classify the test data to be the class which most trees think it should be. That is, if most of the random trees classify a data to be class2, then the random forest will classify the data to be class2.

3. How to use the random forest and random tree

When training and testing the random forest, I will use the indicators of the data to be the "X" for training or testing.

Now, the problem is how to build the Y for training. My way is: for each day, check the price 10 trading days later. If the price later is higher than 1.2*current price, then the Y will be "+1", which means buy; if the price 10 days later is lower then 0.8*current price, then the Y will be "-1", which means sell; otherwise the Y will be "0" for "do nothing".

That is:

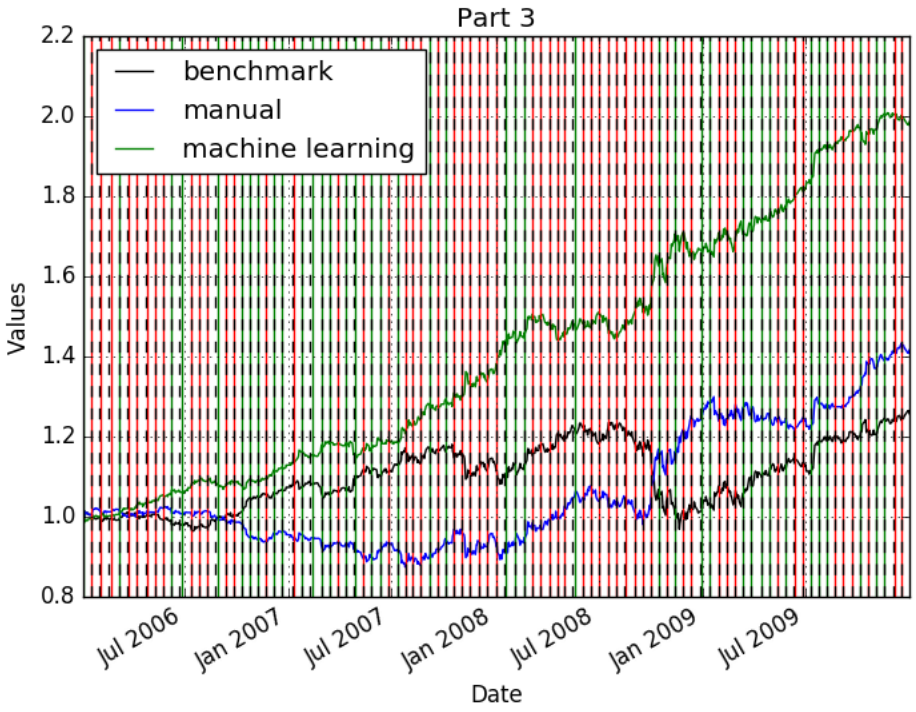
```
ret[t] = (price[t+10]/price[t]) - 1.0
if ret[t] > 0.2:
        Y[t]=+1
else if ret[t] < -0.2:
```

```
        Y[t]=-1
else:
        Y[t]=0
```

Note that some of the data won't have indicator or Y, my way is to cut off the first several days (head) and last several days (tail) of the X and Y for training set, in order to make the training data meaningful to the learner.

After this, when training the random forest, I can simply use the X and Y without NaN rows to train the learner. When doing the test (both in-sample and out-sample), I can pass the edited X to let the random forest gives its suggestion, and modify it to suit the "10-day" rule for this project, and finally get the trading orders.

The chart is shown as follows: (leaf_size=5, bag=7)



the ML trader is providing a in-sample cumulative return of `0.9965`, while the benchmark is only `0.2572`.

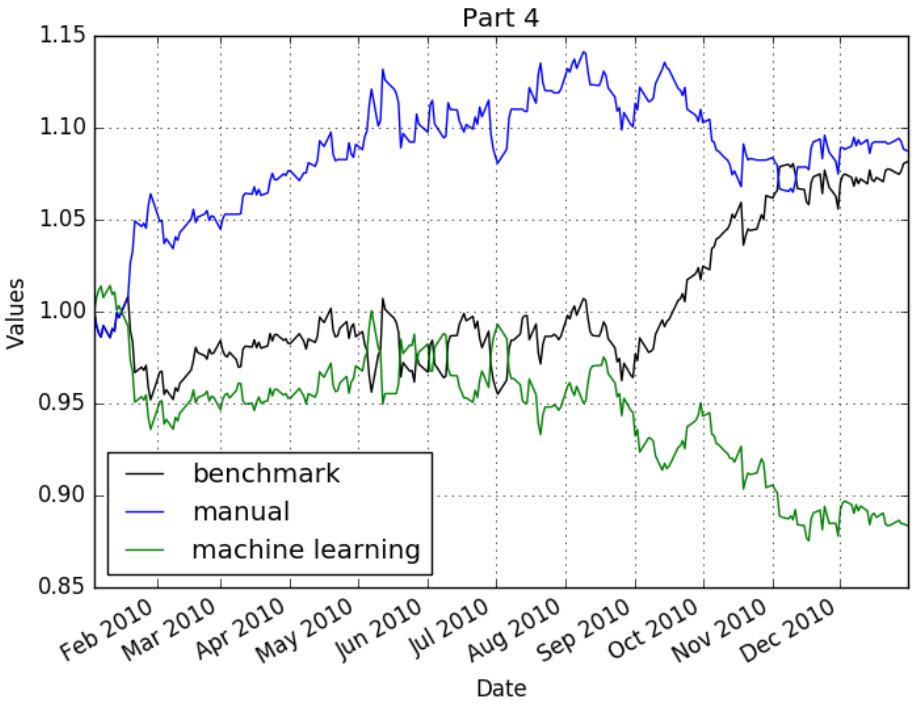
Part 4: Comparative Analysis

I select 4 factors for comparison: Sharpe ratio, cumulative return, standard deviation, average daily return.

Below is the table about the performance of benchmark, manual strategy and machine learning for in-sample and out-sample period.

	Sharpe ratio	Cumulative return	Standard deviation	Average daily return
In-sample: benchmark	0.5848	0.2572	0.0068	0.00025
In-sample: manual	0.8462	0.4094	0.0069	0.00037
In-sample: ML Trader	2.2504	0.9965	0.0050	0.00070
Out-sample: benchmark	0.7605	0.08155	0.0070	0.00034
Out-sample: manual	0.9507	0.09225	0.0064	0.00038
Out-sample: ML Trader	-1.0069	-0.11535	0.0073	-0.0005

The chart is shown as follows:



In general, the manual strategy performs well, and can beat the benchmark in both the in-sample and out-sample period, even though the advantage is less during the out-sample period. The machine learning trader can perform very well in the in-sample period, but perform very badly in the out-sample period.

1.
The problem that Machine Learning trader performs worse in the out-sample period may be caused by over-fitting of the random forest. Since the random forest algorithm is an instance-based learning algorithm, it is using the past situation to perform trading. In the out-sample period, it is using the knowledge in the past to decide what to do in the future. If the IBM is not performing in the same way before, and the machine learning trader fits the training set very well, then it may not perform well in the out-sample test, since it makes decision based on what happened before.

2.
For the manual strategy, even though it still beats the benchmark in out-sample period, the advantage is not as big as the one in the in-sample period. In other words, it is not performing so well like before (but it still beats the benchmark). The reason is similar with “over-fitting” for a machine learning algorithm: in the manual strategy, there are 6 parameters in total. After many different trials, I find a set of parameters which is very suitable for the in-sample period. However, now the 6 parameters fits the training set well, which means they may not perform well on other time periods or stocks.

3.
In comparison, the machine learning trader is much more susceptible to this problem, because it uses the instances in the training set to make a decision (in other words, it depends on the training set too much). For the manual strategy, even I kept modifying the parameters in order to improve the in-sample cumulative return, it will be less susceptible to the same flaw since it doesn’t depend on the training set as much as random forest. Meanwhile I just kept changing the parameters for a while and finally use relatively better ones, so it will not be susceptible to the flaw so much.