CS 7646: Quant strategy using machine Learning

[Arti Chauhan: Apr-10-2017]

Develop trading strategies using Technical Analysis and test them using your market simulator. Utilize your Random Tree learner to train and test a learning trading algorithm.

1.Technical Indicators

- ***a)*** *Is each indicator described in sufficient detail that someone else could reproduce it?*
- ***b)*** *Is there a chart for each indicator that properly illustrates its operation?*
- ***c)*** *Is at least one indicator different from those provided by the instructor's code?*
- ***d)*** *Does the submitted code indicators.py properly reflect the indicators provided in the report?*

1a. For manual rule_based trading strategy I used following four indicators

1. **Bollinger Band percentage (BB%):**

   This indicator essentially captures volatility in price of the stock. Bollinger Bands are volatility bands placed above and below moving average of price and computed as

   SMA = rolling average over n days.
   stdev = rolling standard deviation over n days.
   Upper Band = SMA + 2* stddev
   Lower Band = SMA - 2* stddev
   BB% = (Price – Lower Band)/ (Upper Band – Lower Band)

   Band widens when volatility increases and shrinks when volatility decreases. As such, it is used to determine if prices are relatively low or high.

   BB% interpretation

   - BB% > 1   : price moved from inside to outside, crossing upper band. It indicates price is relatively high /stock oversold.
   - BB% < 0: price moved from inside to outside, crossing lower band. It indicates price is relatively low /stock overbought.

2. **Rate of Change (ROC):**

   ROC is a momentum oscillator, which measures the speed at which price is changing. It is quite useful in identifying overall direction of underlying trend. Period n is set to appropriate value depending on the goal – higher n if goal is to identify long term momentum and smaller n for short term momentum.

   ROC = [(Price[t]/Price[t-n])-1]* 100

   ROC interpretation

   - ROC > 0 : surge in price. If ROC goes above a predefined threshold, it helps identify overbought condition.

- ROC <0 : decline in price. If ROC goes below a predefined threshold, it helps identify oversold condition.

3. **Price/SMA :**

This indicator identifies when price diverges from its SMA (moving average).

Price/SMA = Price[t]/ SMA(Price[t-n : t])

Price/SMA interpretation

- P_SMA > 0 : Price is higher than its previous n days SMA. When P_SMA goes above a predefined threshold, it signifies overbought condition.

- P_SMA < 0 : Price is lower than its previous n days SMA. When P_SMA goes below a predefined threshold, it signifies oversold condition.

4. **Slow Stochastic Oscillator (SSO)**

Stochastic Oscillator essentially measures the level of the closing price, relative to the high-low range over a given period of time and help identify overbought and oversold levels.

Computation:

Stochastic Oscillator computes %K and % D as follows

%K = (Current Close - Lowest Low)/(Highest High - Lowest Low) * 100

%D = 3-day SMA of %K

where

Lowest Low = lowest low-Price for the look-back period

Highest High = highest high-Price for the look-back period

Fast Stochastic Oscillator:

Fast %K = %K basic calculation mentioned above

Fast %D = 3-period SMA of Fast %K

Slow Stochastic Oscillator:

Slow %K = Fast %D

Slow %D = 3-period SMA of Slow %K

Interpretation:

The Slow Stochastic Oscillator is a smoothed version of Fast Stochastic Oscillator.

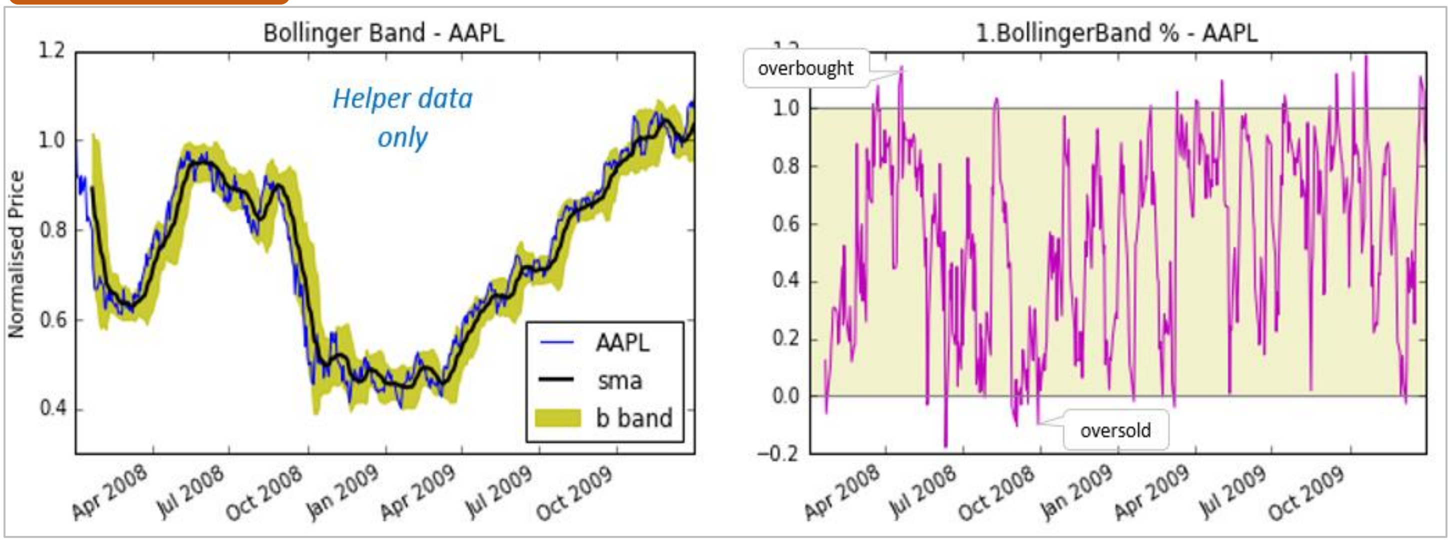
Notice %K in the Slow Stochastic Oscillator equals %D in the Fast Stochastic Oscillator

- SSO > Threshold (>80) : overbought.
- SSO < Threshold (<20) : oversold

1b. Charts for above mentioned four indicators are provided below

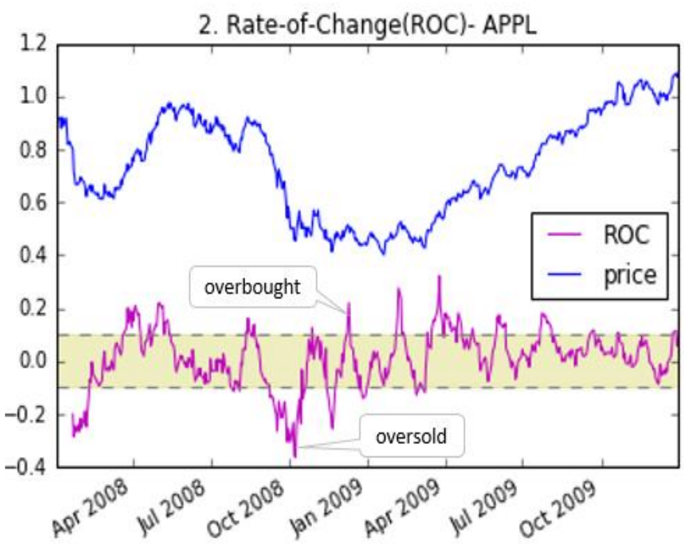
- Indicators are shown in magenta color.
- Light yellow shaded area under indicators, marks region of no-action.
- When indicator goes north of this yellow, it signals overbought condition and if it goes south it signals oversold.
- For BB% and SSO, I have shown helper data for indicators on a separate chart (on left) for sake of readability.

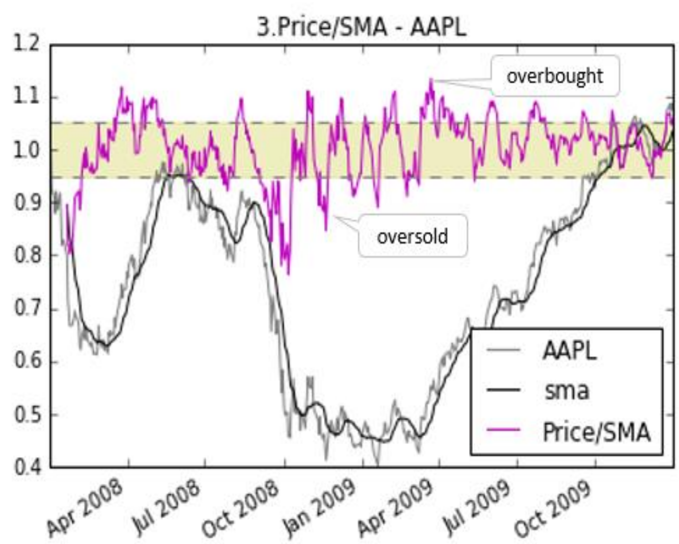
1. Bollinger Band %



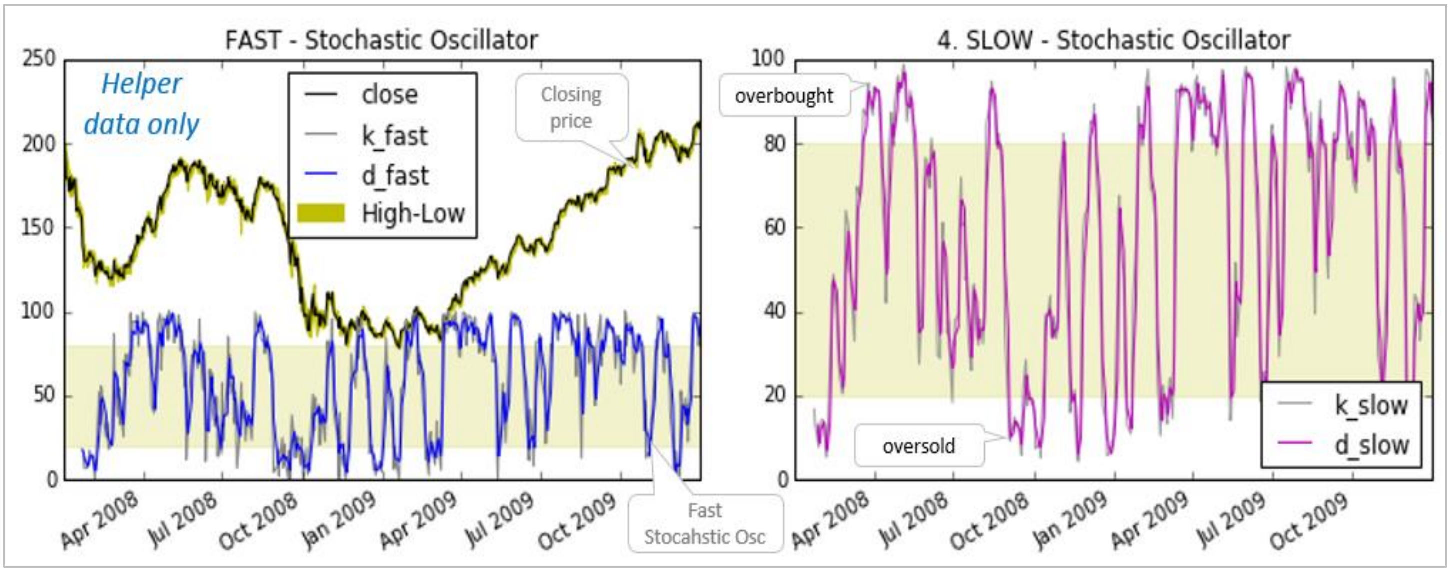
2. ROC

3. Price/SMA





4. Slow Stochastic Oscillator



1c. Indicator#4 ‘Slow Stochastic Oscillator’ is a new indicator that I implemented in this project. It was neither covered in lectures nor in instructor’s code.

1d. Please see the code in Indicators.py.

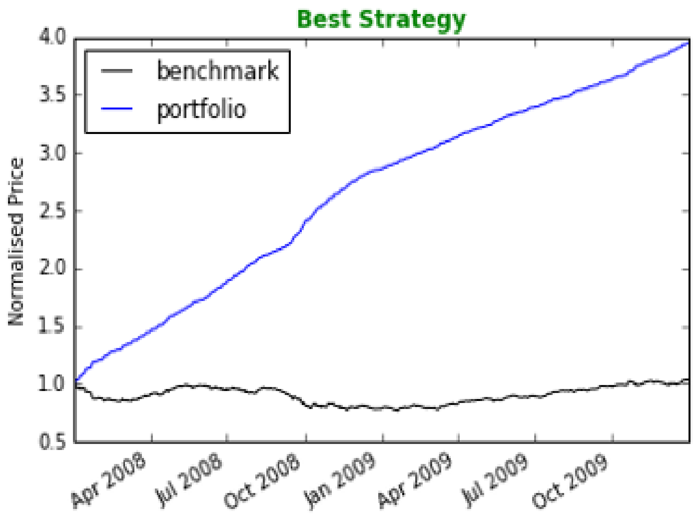
Set option=1 in main.py to plot charts for indicators.

```
instructions = "\
   select option \n \
       #1 to plot indicators \n \
       #2 to execute best strategy \n \
       #3 to run manual trader \n \
       #4 to run ML trader \n"

option = 1

if option==1 :
    plot_normalised_Indicators()

elif option==2:
    best_strategy()

elif option==3:
    run_manual_trader(version=1)
    run_manual_trader(version=2)

elif option==4:
    run_ML_trader()
else:
    print "Invalid option : " , option
    print  instructions
```

2.Best Possible Strategy

2 : Graph and performance-stats for Benchmark vs. portfolio value with best strategy is shown below.

Set option=2 in main.py to execute best strategy.



Portfolio

Cummulative_Return : 2.94948
Stdev of Daily Returns : 0.00321937085748
Mean of Daily Returns : 0.00273420127456

Benchmark

Cummulative_Return : 0.03164
Stdev of Daily Returns : 0.00874053752015
Mean of Daily Returns : 0.000100069116968

3.Manual Rule-Based Trader

a) Is the trading strategy described with clarity and in sufficient detail?
b) Is chart provided?
c) Does the submitted code rule_based.py properly reflect the strategy provided in the report?
d) Does the manual trading system provide higher cumulative return than the benchmark over the in-sample time period?

3a: Manual strategy makes use of four technical indicators described in section-1. Below is high level work flow of Manual Trader.

a) Read in Apple's stock data (Adjusted close, High, Low) for desired time period.
b) Define appropriate lookback period (n) for the indicators. In my case, it was set to 14 days.
c) Compute Technical Indicators as described in section-1 using information from step a and b. Sample output of step-c (features_dataframe) is shown below.

	BB%	Price/SMA	ROC	Slow Stochastic Oscillator
	bbp	psma	roc	d_slow
1/2/2009	0.590	1.017183	-7.654573	12.44564848
1/5/2009	0.879	1.0633226	-0.169599	23.99235975
1/6/2009	0.796	1.0471622	-2.525784	42.66610256
1/7/2009	0.704	1.0282037	2.0725389	58.69418859
1/8/2009	0.801	1.0442864	3.649635	64.3731534
1/9/2009	0.631	1.0194745	0.6360187	60.97914155
1/12/2009	0.493	0.9989654	3.4086916	54.59587956
1/13/2009	0.407	0.9867573	1.5463318	45.98520866

d) Run output of step-c through build_orders () module, which generates 'orders.csv' file.

e) Pass orders.csv through Market-Simulator (reuse code from mc2p1), which will compute the stock's performance and generates graph as shown in -3b below.

Pseudo code for build_orders () module

```
def build_orders(df_features) :
        Wait_period=21
        ALLOWED_SHARES =200
        Holdings=0
        Orders=[]
        For all rows in df_features :
                Wait_period -=1
                If (d_slow < 10) and ( psma <0.8) and (bbp<0.15) and (roc<-15) :
                        If (holdings < ALLOWED_SHARES) :
                                Orders .append([date ,'AAPL','BUY', ALLOWED_SHARES])
                                Wait_period = 21

                If (d_slow > 85) and ( psma >1) and (bbp>1)  :
                        if (holdings > - ALLOWED_SHARES) :
                                Orders .append ([date ,'AAPL',SELL, ALLOWED_SHARES])
                                Wait_period = 21

            #Must close positon after 21 days
            If(Wait_period == 0) :
                    If(holdings < 0) :
                    # previous postion was SHORT
                            Orders .append ([date ,'AAPL',BUY, ALLOWED_SHARES])

                    If(holdings > 0) :
                    # previous postion was LONG
                            Orders .append ([date ,'AAPL',SELL, ALLOWED_SHARES])

                    holdings=0
     return Orders
```

Sample Orders file produced by build_orders () module

Date	Symbol	Order	Shares
3/5/2010	AAPL	SELL	200
4/6/2010	AAPL	BUY	200
4/23/2010	AAPL	SELL	200
5/24/2010	AAPL	BUY	200
10/15/2010	AAPL	SELL	200
11/15/2010	AAPL	BUY	200
1/10/2011	AAPL	SELL	200

3b and 3d :

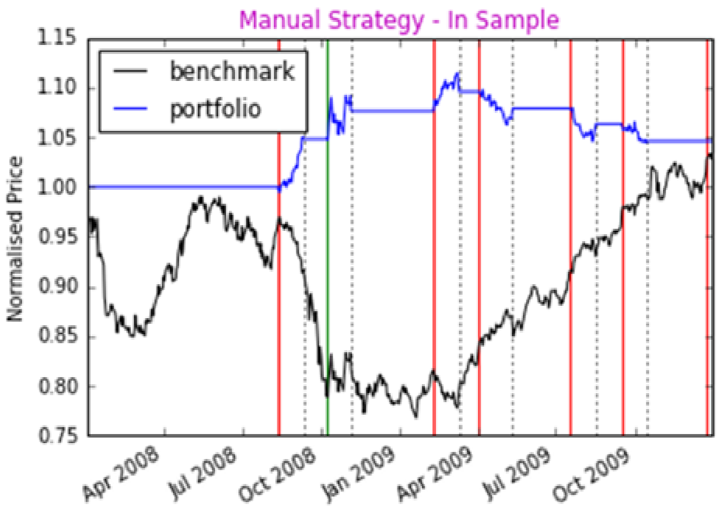
Manual trading strategy (version-1) is able to perform above benchmark and avoids the pitfall during 2008 recession.

I would like to mention about version-2 of manual trader I tried after reading on Piazza that 'it is allowed to enter a position on the <u>same</u> when a position is closed' .Version-2 is built upon version-1 (as described in section 3a) except that it re-enters the same position on 21st day after closing, unless there was a valid close signal generated by Indicators. But I found this approach risky (even though it gave good results) because it almost ignored >95% of the BUY/SELL signals from strategy. Hence I decided to stick with version-1.

> *Version-2 change: position on day-0 = LONG*
>
> *On day-21 :*
>
> *close LONG position.*
>
> *If (Close signal from Indicators == None)*
>
> *then re-open LONG position and start 21-day timer*

NOTE: Please note that in portfolio vs. Benchmark graphs in subsequent sections, I am showing black dotted lines for close, even though that requirement was dropped later by Professor. Reason I did that is because it gives quick visual confirmation that trade was closed exactly after 21 days, which is not possible if only BUY & SELL is plotted. Since I made black line quite pale, it doesn't hinder readability of the graph.

Manual trader Version-1



portfolio

Cummulative_Return : 0.04594

Stdev of Daily Returns : 0.00310695242055

Mean of Daily Returns : 9.39253691001e-05

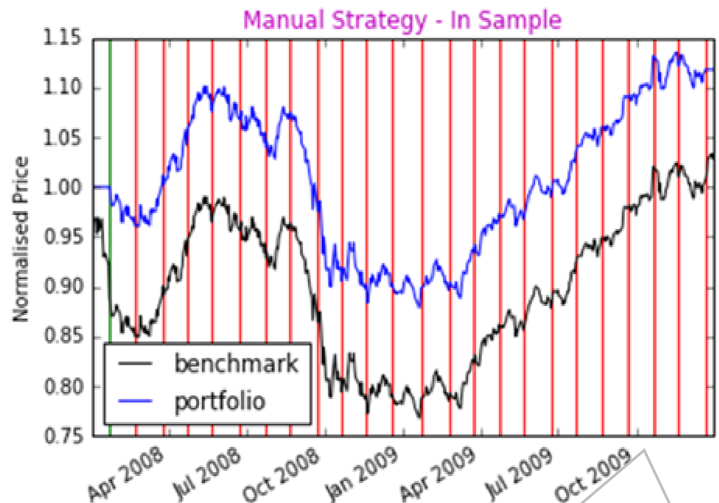
Benchmark

Cummulative_Return : 0.03164

Stdev of Daily Returns : 0.00874053752015

Mean of Daily Returns : 0.000100069116968

Manual trader Version-2



portfolio

Cummulative_Return : 0.1178

Stdev of Daily Returns : 0.00731709764611

Mean of Daily Returns : 0.000247760901326

Benchmark

Cummulative_Return : 0.03164

Stdev of Daily Returns : 0.00874053752015

Mean of Daily Returns : 0.000100069116968

3c: Please see the code in rule_based.py. Set option= 3 in main.py to run manual trader.

4. ML-based Trader

a) *Is the ML strategy described with clarity and in sufficient detail that someone else could reproduce it? (-10%)*
b) *Are modifications/tweaks to the basic decision tree learner fully described (-10%)*
c) *Does the methodology utilize a classification-based learner? (-30%)*
d) *Is chart provided?*
e) *Does the submitted code ML_based.py properly reflect the strategy provided in the report? (-30% if not)*
f) *Does the ML trading system provide 1.5x higher cumulative return or than the benchmark over the in-sample time period? (-5% if not)*

4a: ML trader utilizes technical indicators described in section-1 and classifies them under BUY , SELL or Do Nothing category. High level flow ML trader is described below.

1. Compute technical indicators BB%, Price/SMA, ROC ,Slow Stochastic Oscillator for in-sample period as described in section-1 and standardize it. This will server as feature vector (X) for ML learner.
   Standardization of feature X1: df['X1'] = (df['X1']-df['X1'].mean())/df['X1'].std()

2. Compute output labels (Y) :
   a. Compute 21 day forward return.
   b. Set all.Y == 0 (DO_NOTHING)
   c. If return in step-a > T1 : label == 1 (BUY)
   d. If return in step-a < T2 : label == -1 (SELL)
      Where threshold T1 and T2 can be set based on +/- n standard deviation of 21 day return or some absolute value. For me +/-0.5 gave acceptable performance.

   Sample output of step-1 and 2.

	X (feature vector)				Y
	bbp	psma	roc	d_slow	label
1/23/2008	-0.06133	0.818633	-28.6553	12.98329	-1
1/24/2008	-0.01699	0.818601	-24.6876	12.98329	-1
1/25/2008	0.020163	0.80214	-26.8148	12.98329	-1
1/28/2008	0.103787	0.819341	-24.0852	12.98329	-1
1/29/2008	0.181209	0.844114	-26.6753	10.02631	0
1/30/2008	0.229301	0.866939	-25.7531	8.769986	-1
1/31/2008	0.301465	0.905927	-21.6167	9.925802	-1
2/1/2008	0.308278	0.912152	-25.1882	12.05334	-1
2/4/2008	0.298527	0.918935	-22.1186	13.636	-1
2/5/2008	0.283108	0.921176	-18.967	12.97119	-1
2/6/2008	0.181575	0.885713	-24.1698	10.28419	0
2/7/2008	0.192864	0.898649	-24.8646	7.423585	0
2/8/2008	0.294414	0.948075	-19.3779	6.952463	0

3. Convert the Regression Random Tree to Classification Tree (details in section 4b below) . Feed X and Y to the learner to train the model.

4. Build orders (psuedo code below) and run it through market simulator to compute stock performance.

```
Wait_period=21
For all rows in data :
        Wait_period -=1
```

```
If (predicted_Y == BUY) and (wait_period <=0 ) and (holdings < ALLOWED_SHARES) :
        Add_order ([date ,'AAPL','BUY', ALLOWED_SHARES])
        Wait_period = 21

If (predicted_Y == SELL) and (wait_period <=0 ) and (holdings > - ALLOWED_SHARES) :
        Add_order ([date ,'AAPL',SELL, ALLOWED_SHARES])
        Wait_period = 21
```

```
#Must close positon at 21st day
If(Wait_period == 0) :
        If(holdings < 0) :
        # previous postion was SHORT
                Add_order ([date ,'AAPL',BUY, ALLOWED_SHARES])

        If(holdings > 0) :
        # previous postion was LONG
                Add_order ([date ,'AAPL',SELL, ALLOWED_SHARES])

        holdings=0
```

5. I experimented with different leaf sizes and bag counts in range 5 to 50 and achieved reasonable performance with leaf size=5 and bag_count=20. Leaf size < 5 was not used to avoid overfitting the model to training data.
    a. Acurracy of label prediction ranged from 0.82 to .94, with mean of 0.92

4b:

<u>Creation a classification learner</u>

My ML trader uses Random Tree Classification learner, which is based on Adele Cutler's Random tree implemenation.

a) *It randomly picks a feature to split on (ie it doesn't use information gain or correlation for feature selection.)*
b) *Split-value is determined by taking mean of two randomly selected feature value.*

Following changes were made to convert a RegressionTree (implemented in mc3P1) to a Classification Tree

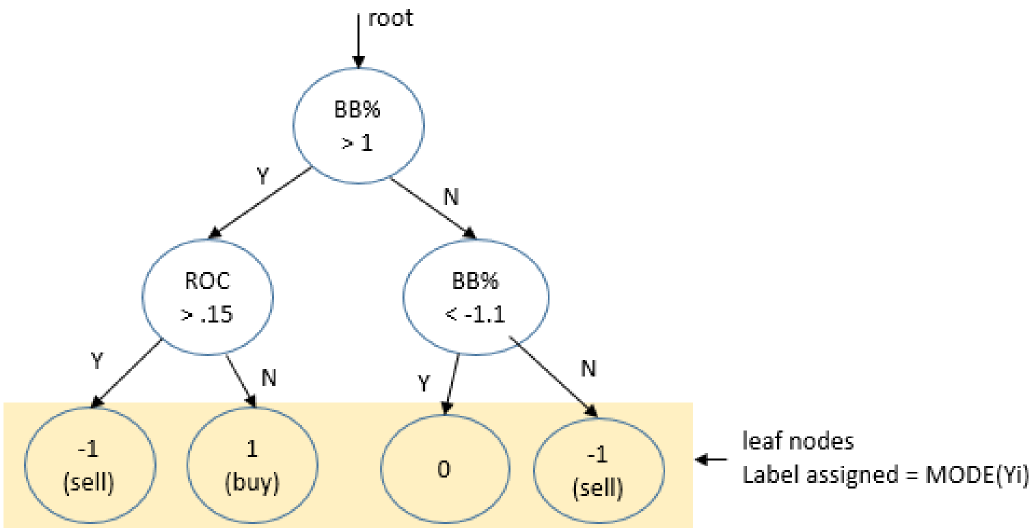
- Use Mode (instead of Mean) of selected feature values (Xi) to create label for a leaf node. This gives a disrete value for label.
  Psuedo Code
  ```
  Build_randome_tree(data) :
          If (data.shape[0] ==1) or (all data.y same) :
                  return [ leaf , Mode(data.y) , NA , NA)
  ```

- I also modified Bag Learner where it computes Mode of Y values (labels) returned by each RT_Learner instance.
  Psuedo Code
  ```
  Query_bagLeaner(data)
          For i in number of bags :
  ```

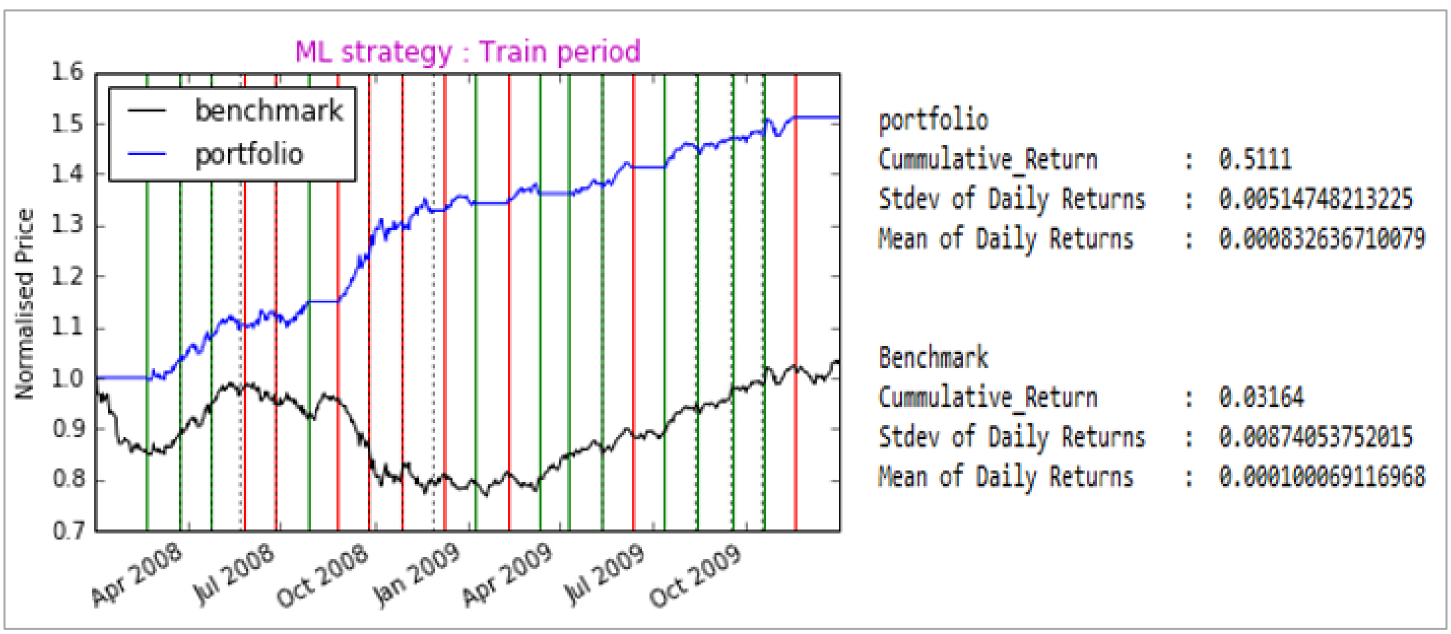
```
            PredY[i] =  RT_learner[i].query
        return Mode(PredY)
```

4c : Please see previous section 4a and 4b for details on classification learner implementation.

To summarize, ML strategy computes a feature vector (X) using technical Indicators and discrete labels Y - 1,-1 or 0 which represets BUY, SELL ,DO_NOTHING based on 21-day forward return.
From this (X,Y) data, learner builds a tree where leaf nodes are assigned discrete labels 1,-1 or 0. This is done by taking **Mode** of Y_i . This ensures that predicted_Y is discrete /non-continous and signals conditon for BUY, SELL ,DO_NOTHING.
Simple visualization of classification learner is given below.



4d :



4e : Please see code in ML_based.py. Set option=4 in main.py to run ML trader.

4f : ML trader gives 16x times higher cumulative return than the benchmark. (0.5111 vs. 0.03164). Please see section 4d.

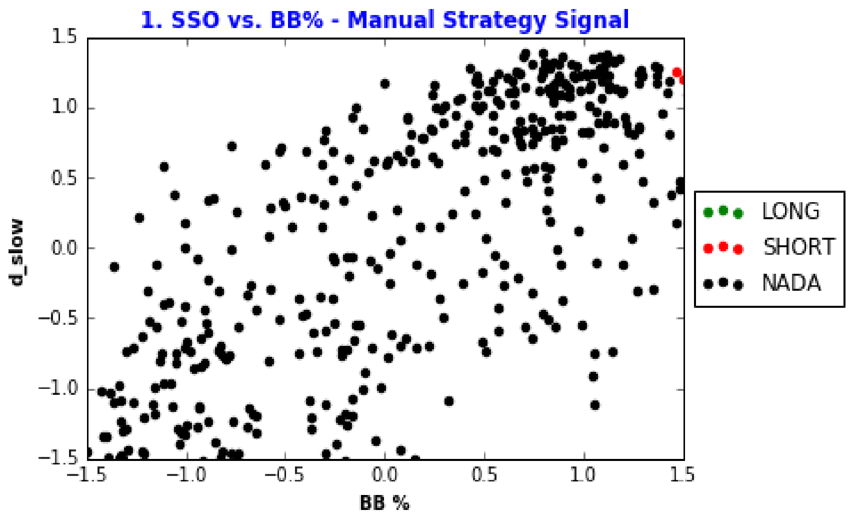
5. Visualization of data

Charts below show scatterplot of two features namely Bollinger Band% (x-axis) vs. D% of Slow Stochastic Oscillator (y axis).

Note: Each point in scatterplot represents the signal generated by rule_based and ML trader. Please note that this will include BUY/SELL signals that were generated by trader but not entertained due to 21-day holding requirement. I used this approach because it gives more points to compare between actual vs. predicted labels in ML-Strategy's case.

5

1. Rule based Trader

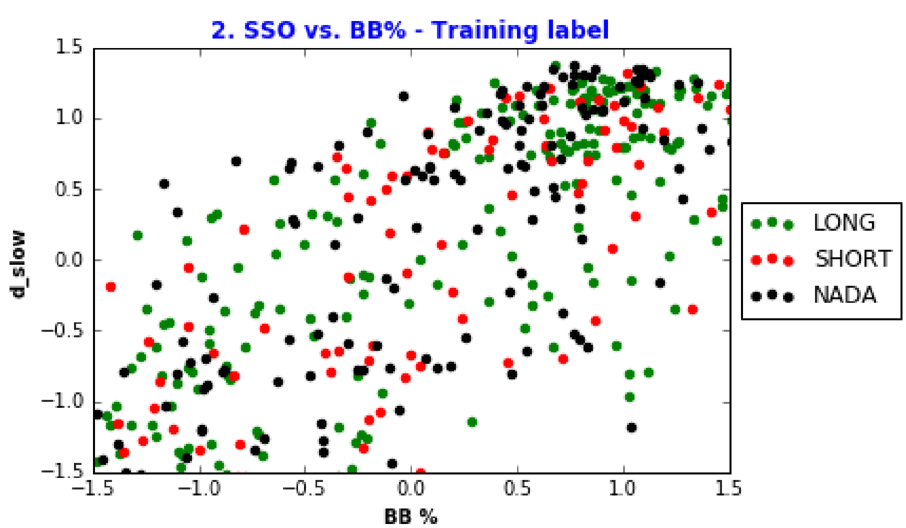


Please note there were much fewer BUY/SELL signal generated by manual trader when compared to ML trader. Hence there is dominance on black points.

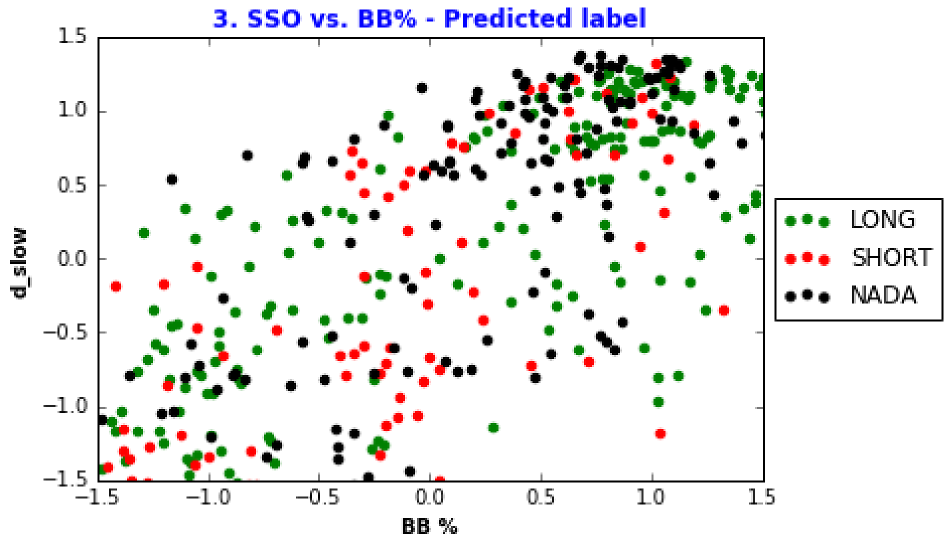
Many of these signals are at +/- 1.5-1.7 boundary and hence don't show up in this chart.

I had to restrict the axis to +/- 1.5 captured to meet the project requirement.

2. The training data for ML Trader.



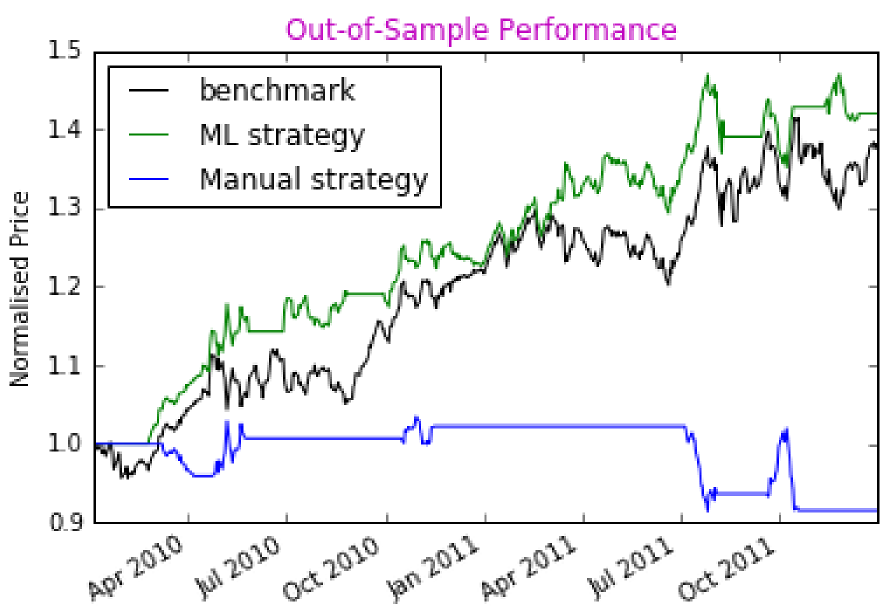
3. Response of ML learner when queried with the same data (after training).



Learner was able to correctly predict labels for ~89% of the in-samples.

6. Comparative Analysis

6a: Out of sample performance comparison for both ML and Manual strategy



6b: Comparison of cumulative return for both ML and Manual strategy and for In and out of sample dataset is given below.

Cummulative Return

	In-sample	Out-of-Sample
Benchmark	0.03164	0.38034
ML	0.54304	0.41998
Manual	0.04594	-0.0848

Portfolio performance better the benchmark for

- ➢ In-sample ML strategy (16x)
- ➢ In-sample Manual Trader (1.45x)
- ➢ Out-of-Sample ML strategy (1.1x)

However Manual strategy performed poorly for out-of-sample.

Below is comparison of other performance indicators such as volatility, Sharpe ratio and average daily return.

- ➢ For in-sample dataset, ML strategy gives best Sharpe Ratio even though it has slightly higher volatility than manual strategy.
- ➢ For out-of-sample, ML strategy gives slightly better Sharpe Ratio and lower volatility than benchmark.

In-Sample

Benchmark
Cummulative_Return : 0.03164
Stdev of Daily Returns : 0.00874053752015
Mean of Daily Returns : 0.000100069116968
Sharpe Ratio : 0.181744884778

In-sample : ML Strategy
Cummulative_Return : 0.54304
Stdev of Daily Returns : 0.00520796321639
Mean of Daily Returns : 0.000874485026422
Sharpe Ratio : 2.66553715036

In-sample : Manual Strategy
Cummulative_Return : 0.04594
Stdev of Daily Returns : 0.00310695242055
Mean of Daily Returns : 9.39253691001e-05
Sharpe Ratio : 0.479897600224

Out-of-Sample

Benchmark
Cummulative_Return : 0.38034
Stdev of Daily Returns : 0.00855351975727
Mean of Daily Returns : 0.00067750810562
Sharpe Ratio : 1.25738971294

Out-of-Sample : ML Strategy
Cummulative_Return : 0.41998
Stdev of Daily Returns : 0.00675593258234
Mean of Daily Returns : 0.000720096768351
Sharpe Ratio : 1.69202129749

Out-of-Sample : Manual Strategy
Cummulative_Return : -0.0848
Stdev of Daily Returns : 0.00511423096223
Mean of Daily Returns : -0.000163047979857
Sharpe Ratio : -0.75508244666

ML strategy: in vs. out of sample

- Though out-of-sample performance is better than benchmark for ML strategy, it is much lower when compared to in-sample.
- This is no surprise as model is expected to not perform that well on unseen data.
- I took precautions to not over fit the model to training data by keeping leaf_size >=5 and utilizing bagging.
- Since classifier uses random tree, there was quite some variance in the results. I managed to reduce this variance to a great extent by increasing bag count but it's not completely eliminated. Results I presented above are best of 15 runs.

Manual vs. ML strategy

As evident from table above, even though in-sample performance of manual strategy is better than benchmark, it is much lower when compared to ML strategy. Few probable causes that might explain this difference are

- I had mainly relied on trial and error method to tweak thresholds for technical indicators in rule-based strategy, after visualizing it against the price. Hence my manual strategy is as good as my pattern recognition skills, which obviously is not that good.

- Besides Classifier being smarter (ie blessed with computational power) at recognizing relevant pattern in the data, ML strategy also benefited from the fact that it made decision based on 21-day forward return, whereas my manual strategy is based on historical trend.

- Manual strategy abided by one set of hard thresholds, whereas ML strategy took the majority vote (via bagging) to generate a BUY/SELL signal and hence less biased.