1) Write a report describing your system

I am trying to automate trading environment using algorithmic trading. Algorithmic trading is where computer is trading instead of human. Algorithmic trading depends upon historical data with well defined rules and data.
My goal is to train my system enough to forecast trading events.Forecast is triggered by technical indicators. I train these technical indicators in training phase. My implementation is based on classification learner tree. In first part, I calculate X (as classification tree's features) values using these technical indicator. I have chosen Classification tree due to its binary nature. Classification tree creates tree based on the training data. I also calculates Y values as {-1/0/1} based on predefined rules and feed these feature X, Y values to classification tree create the model. In query, phase it outputs as the trading events in numbers -1/0/1 {Short, nothing, Long}

My chosen technical indicators are below:

a) Bollinger Bands : It is band with moving average stock price. It adds 2*standard deviation (volatility) above and below the moving average.
   Bollinger Band value for overbought condition:
   If (normalized_bbp value) <= -0.07, trigger buy
   For oversold condition
   If (normalized_bbp value) >= 0.07, trigger sell

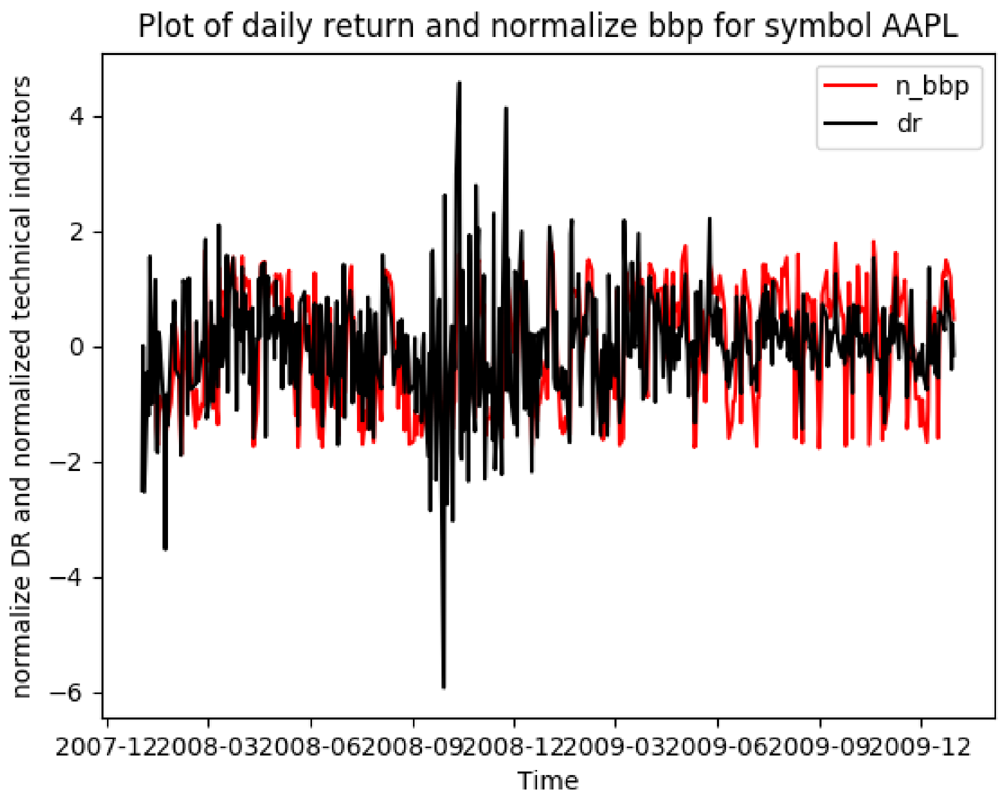
b) Price/SMA : It is the ration of daily stock price by moving average price. I take 7days window for calculating moving average. Price/SMA value for overbought condition
   (normalized_price/sma ratio) <= -0.05
   for oversold condition
   (normalized_price/sma) >= 0.05

c) Standard deviation of daily returns: It is the standard deviation of daily returns. I have chosen 7 days window for calculating rolling standard deviation.
   Stand dev of daily return (volatility) is restricted to 1.5

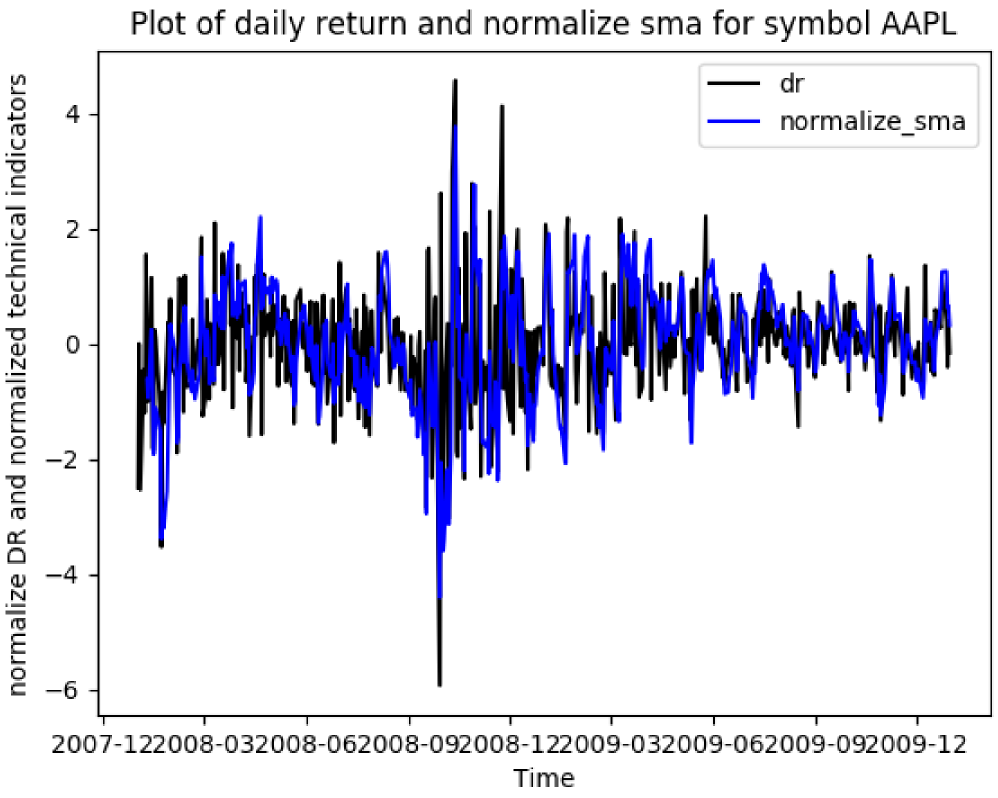
   Create Dataframe with triggered buy/sell signal. If we pass this dataframe through daily portfolio value, we can see the change in portfolio value based on triggered buy/sell events.

   Note : timeframe of AAPl is chose 2008/01/01 – 2009/12/28

Timeseries plot of normalized Bollinger band with daily return. It varies with daily return but create band around daily return.



Timeseries Plot of normalized price/s ma ratio. It varies with daily return .



The centerpiece of your report should be the description of how you have utilized your learner to determine trades

Calculating mod of leave's value. Minimum number of leaves is 5.
Steps to create training data with 3 indicators. Each indicators act as a feature or X value of Classification tree. My rules are described below as an pseudo code. Under this pseudo code, rules should be designed to trigger a "long" or "short" entry. At the terminal node of classification tree, I take mode of leave's values. Minimum number of leaves are allowed 5.

1) I normalize 3 technical indicators using x-mean/sd
2) Created numpy array using 3 standardized indicators
3) Discard lookback times' data. In this case it is 7 days, so remove first 6 days data. Or start training from 7th data
4) Pseudo code to do training:

```
Initialize CTLearner with 5 leaves
Set the buy and sell limit
Loop {steps through each day's data}:
        If (Buy condition for Technical indicators and holding share <= 0)
                If (future return > buy limit)
                        Store the +1 in Y as buy event

        Else If (sell condition for Technical indicators and holding share >= 0)
        If (future stock price ratio < sell limit)
            Store the -1 as Sell event
        Else
            Store the 0 as Do nothing
We have created X data and Y data as training data and it's output
Train the CTlearner with the training data
Testing the Insample/outsample data
    Loop(step through each day's data);
        Maintain position 200/0/-200 based on tree's output 1/0/-1
        Create order dataframe with triggered events
```

From Exact code perspective:

Created classification learner is trained using following condition

Feature1 = normalized Bollinger bands

Feature 2 = standard deviation of daily returns

Feature 3 = price/SMA ratio

a) Long:

    Tree feature for triggerin Long position

        (feature1 <= -0.7 and abs(feature 2)<1.5 and feature3 <= -0.5)

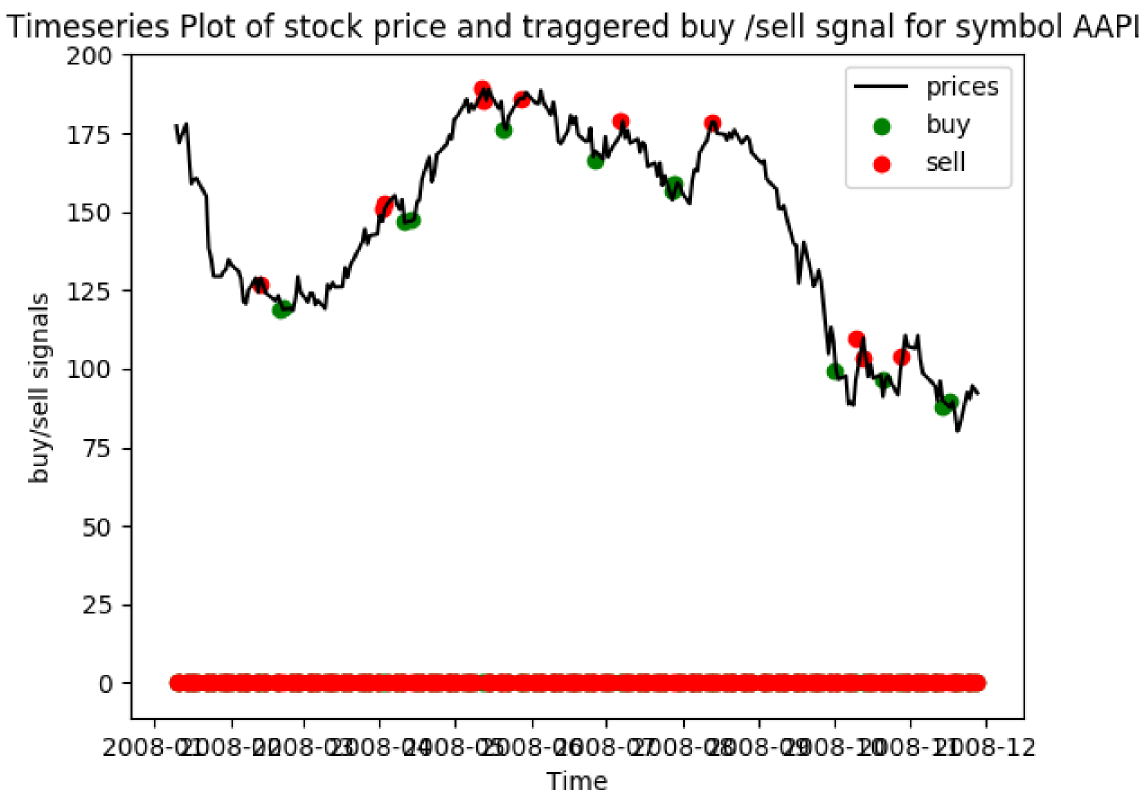
b) Short:

    Tree feature for triggerin Short position

        (feature1 >= 0.7 and abs(feature 2)<1.5 and feature3 >= 0.5)

Timeseries plot of triggered buy/sell events. From plot, it is clear, buy is triggered when stock is oversold, sell is triggered with stock is overbought

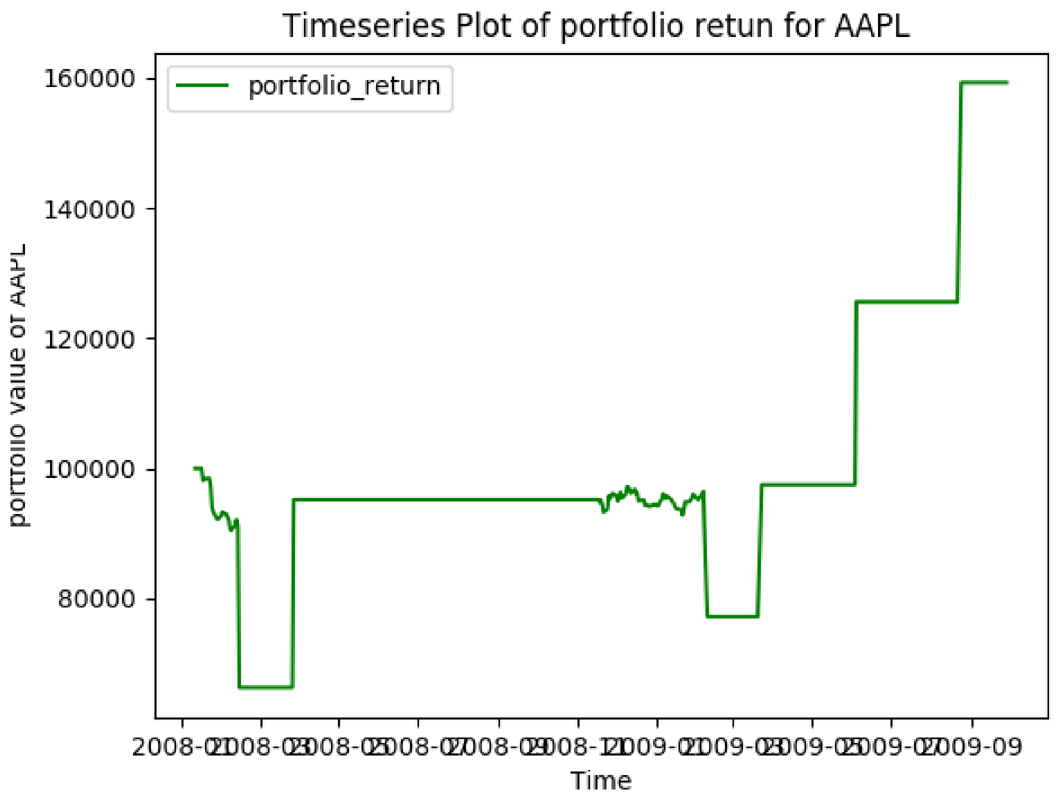
Note : timeframe of AAPl is chose 2008/01/01 – 2008/12/28



Describe the steps you took to frame the trading problem as a learning problem for your learner.

a) From given prices of symbol, I developed the features as the three technical indicators (Bollinger band, price/sma ratio, volatility of daily returns)
b) Normalized the values of these features. This is X data and it goes through trading rule mentioned above.
c) I have developed random classification tree, which randomly picked features values. It splits randomly the given column of feature. It distributes the data in halves.
d) Take the mode of terminal node's values, when feature's number of data is <= 5
e) Captured outputs {mode of terminal node) as an Y as -1/0/1 {Short/Nothing/Long}
f) Feed through classification tree's training function. It create classification tree based on given X and Y value
g) We get the new set of price range for given symbol. We use these new X as an query.
h) Classifciation tree's ouput -1/0/1 is collected and corresponding even is triggered as {short, nothing, Long}
i) Create Dataframe with triggered buy/sell signal. If we pass this dataframe through daily portfolio value, we can see the change in portfolio value based on triggered buy/sell events.

Timeseries plot of Portfolio value due to triggered buy/sell signal. We can see the increase in portfoo value over the time period. Daily portfolio values is calculated based on holding of shares * current price of shares + cash
Note : timeframe of AAPl is chose 2008/01/01 – 2009/9/28



In the course of creating your learning trading strategy you will probably evaluate a number of different (hyper-) parameters for your learner and your trading strategy

Hyperparameters are parameters whose values are set prior to the commencement of the learning process. By contrast, the value of other parameters is derived via training. Hyperparameter optimization or model selection is the problem of choosing a set of optimal hyperparameters[for a learning algorithm, usually with the goal of optimizing a measure of the algorithm's performance on an independent data set.

As my learner is random classification tree, my hyper parametrs are

a) Maximum depth of random tree: Maximum number of terminal nodes
b) Number of features : Number of features to consider while searching for best split.
c) Minimum sample Split: Minimum number of samples which are required in a mode to be considered for splitting. Control overfitting. Too high value can lead to underfitting.

d) Minimum sample leaf: Define the minimum samples required in a terminal node (leaf)

Conduct and report on two experiments that illustrate the methods by which you refined your learner or strategy to excel at the assigned task.

My test cases were failing . I take an instance where I see the improvement and made chart out of it. For symbol SINE_FAST_NOISE, it was returning with
In sample return .34

I chose 2 hyper parameters to experiment with

a) max_depth

code: I made a change in random tree. Maximum depth is also considered one of the parameter to create the terminal node (leaf).

It improved the performance. Excerpt from below chart:

Max_depth	In sample return
Not consdered	.34
6	1.105
7	1.35
8	1.48
9	1.2
10	.87
14	1.2
18	1.67

When max_depth is 6, tree is going through underfitting. When we increase further max_depth, it optimizes the tree, further increasing to 14-18, tre goes through overfitting.

b) minimum samples

Code change: Minimum samples are also considered one of the parameter to create the terminal node. If node contains data less than the minimum samples, we create terminal node.
It improves performance. Excerpt from below chart:
Minimum samples = 0.5 -1% of total data

Minimum samples	In sample return
Not consdered	.34
.5% of data	.78
.6% of data	1.23
.8% of data	1.45
1% of data	1.76
3% of data	1.76

It improves the performance till 1% of data, after increasing its value performance doesn’t change further.