Name: Nidhi Nirmal Menon
GT username: nmenon34
GT ID: 903328735

Machine Learning for Trading

Strategy Learner

Steps taken to frame the trading problem as a learning problem for the learner:

The problem in this assignment is a trading problem that is concerned about whether or not to trade a stock on a particular day. We convert this trading problem into a learning problem for the learner. This is done by using the available data as training (in-sample) data for the learner and then using the new data as testing (out-sample) data. In this case, I have implemented an ensemble learner using BagLearner and RTLearner (Random Tree learner) which is a classification learner. So I convert the actions in the trading problem (BUY, SELL or DO NOTHING) to classes in the learning problem (+1, -1, 0). So when the learner is thus trained and then I run it on testing data, it returns the class label, each indicating an action in the trading problem.

Description:

Phase 1: Setting up the learning environment

- The Strategy Learner creates an instance of the BagLearner with leaf-size as 5 and bag-size as 20. The BagLearner in turn creates instances of RTLearner (20 random trees since bag-size is 20).

Phase 2: Training phase

- In the addEvidence() method, in-sample (training) data is read using util.py
- Technical indicators are calculated using indicators.py
- The data returned by indicators.py is concatenated to form the trainX dataframe which consists of values of all 3 indicators
- The N-day change in price is compared with the market variance considered (2% in my code) since impact=0, to generate the trainY data which could be +1, -1 or 0 to represent the 3 classes that the data is being classified into. The trainY data are in fact the labels of the classes.

Phase 3: Testing phase

- In testPolicy() method, out-sample (testing) data is read using util.py
- Technical indicators are calculated using indicators.py
- The data returned by indicators.py is concatenated to form the testX dataframe which consists of values of all 3 indicators
- We call the query() method of the learner to generate testY data

Name: Nidhi Nirmal Menon
GT username: nmenon34
GT ID: 903328735

- We then create a trades dataframe to show the trades that are carried out. Since the allowed stages are +1000, 0 and -1000, the trades could be anything between +2000 and -2000, depending on the testY value

Indicators:

I have used the same three indicators for both, manual strategy and my strategy learner. They are:

1. Simple Moving Average (SMA)
   A simple moving average (SMA) is an arithmetic moving average calculated by adding the closing price of the security for a number of time periods and then dividing this total by the number of time periods.

$$SMA_t^{(n)} = \frac{1}{n}\sum_{i=0}^{n-1} Price_{t-i}$$

2. Bollinger Bands
   Bollinger Bands® are a highly popular technical analysis technique. It is plotted two standard deviations away from a simple moving average.

$$Upper - band = rolling - mean + (2 * \sigma(prices))$$

$$Lower - band = rolling - mean - (2 * \sigma(prices))$$

3. Volatility
   Volatility is a statistical measure of the dispersion of returns for a given security or market index. Volatility is considered to be the standard deviation of prices over a window.

$$Volatility = \sigma = \sqrt{prices}$$

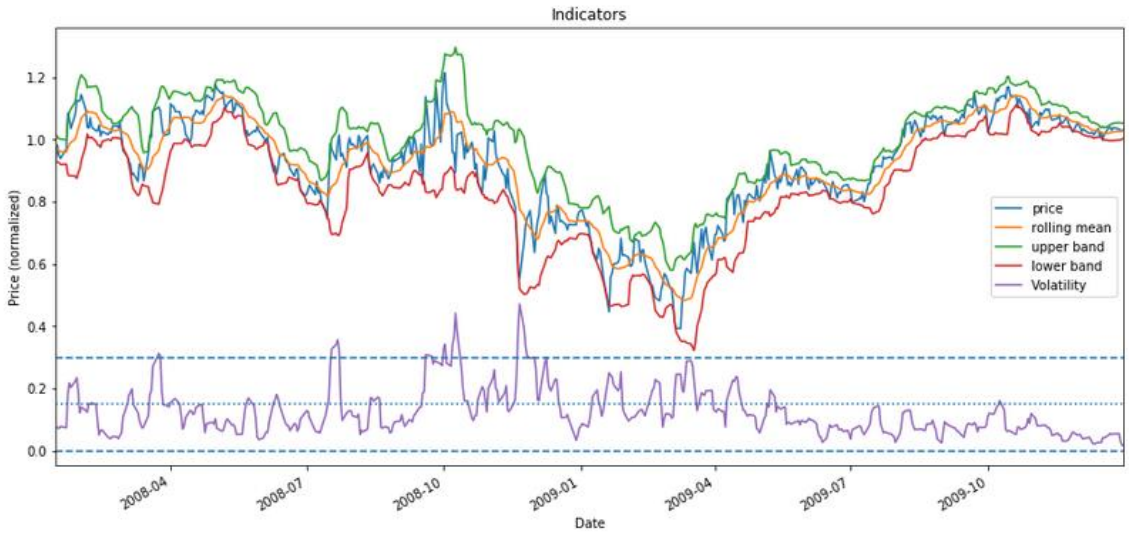

Fig.: Graphical representation of the three indicators that I have used in both, manual strategy and strategy learner

Name: Nidhi Nirmal Menon
GT username: nmenon34
GT ID: 903328735

Adjustment of data:

I did not have to adjust the data in any way because of my choice of learner. I have chosen an ensemble technique of BagLearner with RTLearner (Random Tree Learner) in this case which works on the data available in this assignment (i.e. prices). Had I chosen a learner like QLearner, I would have had to discretize the data before the learner could be trained or tested.

Random Tree Learner works by just classifying the data based on the labels for which it only needs to calculate the mode of the input data. Hence no sort of adjustment was carried out on the data for the functioning of the strategy learner.

Name: Nidhi Nirmal Menon
GT username: nmenon34
GT ID: 903328735

Experiment 1:

The code for this experiment is included in experiment1.py

In this experiment, I have used the above mentioned 3 indicators to compare my manual strategy with my learning strategy in sample. For this experiment manual strategy works as in the previous assignment, based on defined if-else rules for individual indicator values to arrive at the final decision of whether to BUY, SELL or DO NOTHING with the shares. As for strategy learner, it uses an ensemble of machine learning techniques to make the same decision. The benchmark is to buy 1000 shares on the first trading day and sell 1000 shares on the last day. The trades are considered only for the symbol 'JPM' in this case.

Experiment description:

Steps:

1. We read in the prices using code in util.py
2. We initialize **strategy learner** and train it using addEvidence() method. Once the learner is trained, it is tested using testPolicy(). The resulting dataframe is then passed to a function where decision is taken on whether to BUY or SELL shares considering that the allowable positions are 1000 shares long, 1000 shares short, 0 shares. We then compute portfolio values for the strategy learner. This is displayed in the plot in **green color**.
3. Next, we use the testPolicy() of the **manual strategy** to compute the dataframe by that method. The portfolio values are calculated for the result and is displayed in the plot in **black color**.
4. We construct a dataframe for the **benchmark conditions** and compute its portfolio values and display in the plot in **blue color**.

Assumption:

Our assumption is that strategy learner will work better than benchmark and manual strategy.

Parameters:

Symbol: JPM
In-sample period: January 1, 2008 to December 31, 2009
Starting cash: $100,000
Allowable positions: 1000 shares long, 1000 shares short, 0 shares
Commission: $0.00
Impact: 0.00
Leaf-size: 5
Bag-size: 20

Outcome:

A comparison of both, manual and learning strategies show that while both beat the benchmark, the Strategy Learner gives a better overall performance because in manual strategy we are making decisions solely on the basis of 3 indicator values whereas strategy

Name: Nidhi Nirmal Menon
GT username: nmenon34
GT ID: 903328735
learner uses ensemble techniques along with indicators which reduces overfitting and gives better performance on data. This can be observed from the following charts obtained on running experiment1.py.

Graphical results:

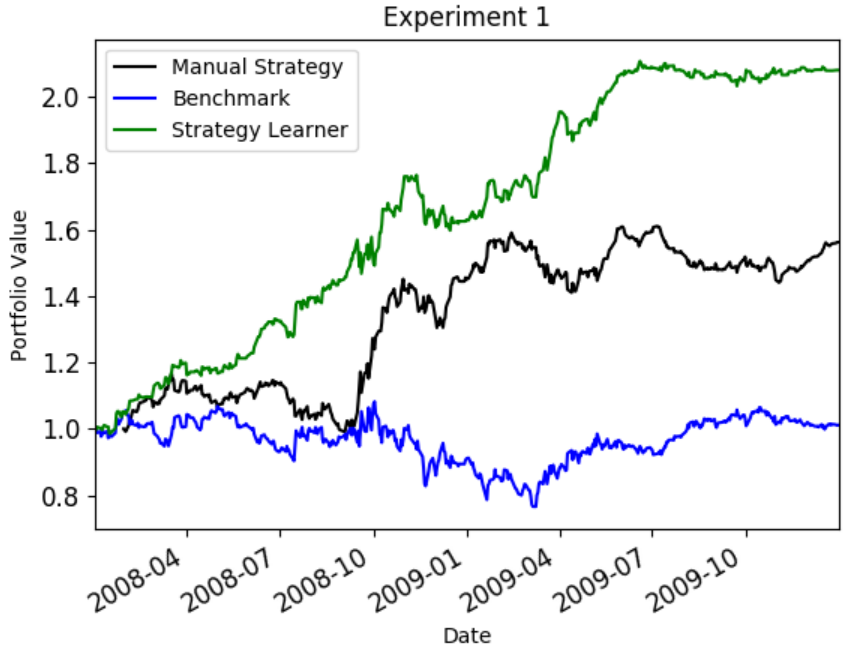

Fig.: Comparison of manual strategy and learning strategy with benchmark

At times it was observed that for a short time period, Strategy Learner was a bit inefficient and performed worse than both, benchmark and manual strategy. But for the most part it was the best. An example of such an observation is given below:

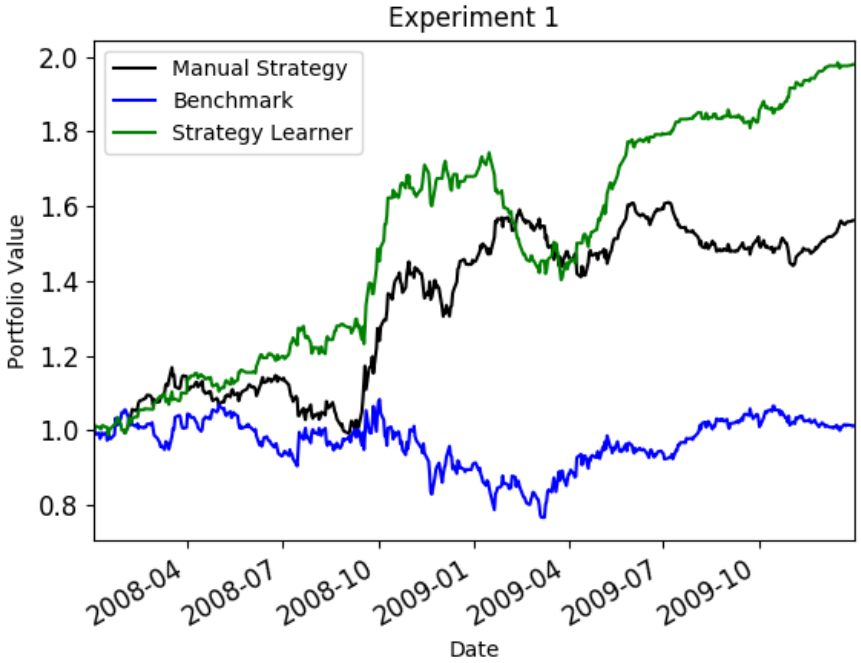

Fig.: Comparison of manual strategy and learning strategy with benchmark

Name: Nidhi Nirmal Menon
GT username: nmenon34
GT ID: 903328735

Statistical results:

Impact Ratio	Benchmark	Manual Strategy	Strategy Learner
Average of daily returns	0.000168086978191	0.000995095526282	0.00126637380587
Standard deviation of daily returns	0.0170043662712	0.0122870682009	0.0113597508166
Cumulative returns	0.0123	0.562	0.8322
Sharpe Ratio	0.156918406424	1.28563230068	1.76967446448

Fig.: Comparison of benchmark with manual strategy and strategy learner

Would you expect this relative result every time with in-sample data? Explain why or why not?

For in-sample data, I would expect the strategy learner to certainly cross the benchmark and in most cases even out-perform the manual strategy which is almost like a brute-force method with human interpretation. There is a slight possibility of strategy learner underperforming since it uses Random Tree learner in the ensemble technique which introduces randomness due to which the decisions could be suboptimal rather than optimal.

Name: Nidhi Nirmal Menon
GT username: nmenon34
GT ID: 903328735

Experiment 2:

The code for this experiment is included in experiment2.py

Hypothesis:

Impact is the extent to which the buying or selling moves the price against the buyer or seller, i.e., upward when buying and downward when selling. In other words, it is the cost that a buyer or seller of stocks incurs while executing a transaction due to the prevailing liquidity condition on the counter. Hence, it will work in the favor of the trader if the impact is low. So my hypothesis is that the strategy learner should work better for smaller values of impact.

As for the metrics, I hypothesize that as the impact value decreases, 'cumulative returns' and 'sharpe ratio' should increase.

Description:

The value of the ratio of N-day change in price (i.e. (N day price difference/current price) for an N-day lookup window) has an influence on the training labels 'trainY' assigned in strategy learner. Along with this ratio, I take into consideration the impact factor and a threshold for market variance (set to 2% in my code) when deciding the labels. The labels are assigned as 1, 0 or -1 depending on whether the ratio is greater than, equal to or less than the sum of market variance and impact factor. These labels denote classes for the RTLearner which signify actions in the trading world:

+1 : LONG

0 : CASH

-1 : SHORT

Parameters:

Symbol: JPM
In-sample period: January 1, 2008 to December 31, 2009
Starting cash: $100,000
Allowable positions: 1000 shares long, 1000 shares short, 0 shares
Commission: $0.00
Impact: 0.05, 0.005, 0.0005
Leaf-size: 5
Bag-size: 20

Testing the hypothesis:

1. We read in the prices using code in util.py
2. For every value of impact factor, do the following steps:
   a) We initialize strategy learner and train it using addEvidence() method.

Name: Nidhi Nirmal Menon
GT username: nmenon34
GT ID: 903328735

b) Once the learner is trained, it is tested using testPolicy().
c) The resulting dataframe is then passed to a function where decision is taken on whether to BUY or SELL shares considering that the allowable positions are 1000 shares long, 1000 shares short, 0 shares. So the trade can be anything between buying a 2000 shares to selling 2000 shares. We then compute portfolio values for the strategy learner.

3. Plot graphs of Strategy Learner for all three values of impact.

Results:

The results observed support the hypothesis.

Graphical results:

I have plotted the chart (the code for which is included in experiment2.py) for impact values 0.0005, 0.005 and 0.05. The general trend in portfolio values for the three cases was observed to be following the order:

p_val (impact=0.0005) > p_val(impact=0.005) > p_val(impact=0.05)

The chart looks as shown below:

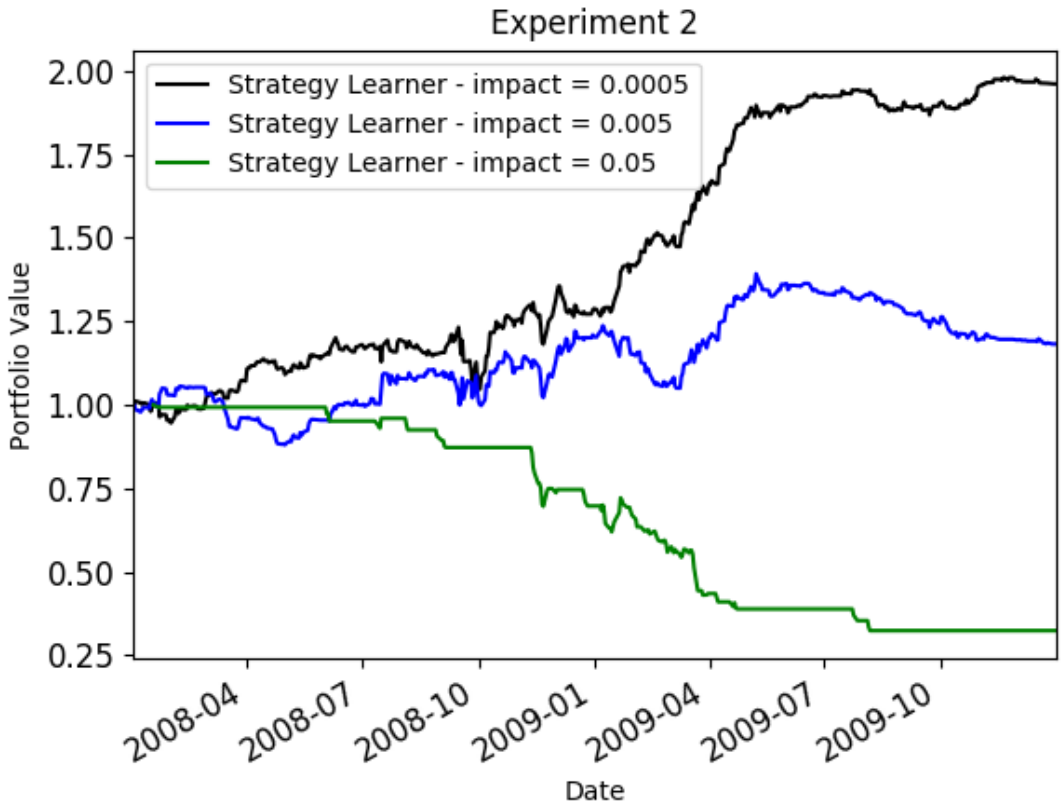

Fig.: Comparison of portfolio values for strategy learner
for three different values of impact

Name: Nidhi Nirmal Menon
GT username: nmenon34
GT ID: 903328735

Statistical results:

Impact Ratio	0.05	0.005	0.0005	Trend as impact value decreases
Average of daily returns	-0.00538095936793	0.000777693652855	0.00160178192606	Increases
Standard deviation of daily returns	0.0243020401942	0.0130371839143	0.0104172790588	No particular trend was observed
Cumulative returns	-0.943786318687	0.94694560505	2.44089647995	Increases
Sharpe Ratio	-3.5149345953	0.418309618551	1.18041310246	Increases

These tabular results prove the hypotheses that as impact value decreases, ‘cumulative returns’ and ‘sharpe ratio’ increase.