Project 8 Strategy Learner: Random Forest Approach

CS7646 – Machine Learning for Trading
Michael Tong (mtong31)

Introduction:

This project began with a fairly simple choice, selection of a machine learning method for defeating my almighty manual learner. However, with a -10.5% return on out of sample data, this is not a monumental task. The selection criteria thus was more subjective and ultimately I decided upon using the decision tree method since it was done earlier on and I want to *reinforce* my understanding of it. Use of the decision tree also allows me to revisit the project and optimize its performance, since it did not go too swell the first time around (also gives me some time to mess around with boosting).

For use of the decision tree, the project criteria specifies the random decision tree formulation method, and to reduce variance and overfitting, bootstrap aggregating (bagging) is employed. With the algorithm selected, the next selection criteria is the technical indicators. Since this project is performed in comparison with our manual learner, the four indicators from the previous project is used as a baseline, namely, Bollinger Bands (BB), Stochastic Oscillator (STOSC) crossings, Stochastic Oscillator movements, and Moving Average Convergence-Divergence (MACD) crossings. The additional indicators tested were the Commodity Channel Index (CCI), Simple Moving Average (SMA), and On-Balance Volume (OBV). The indicators needed some slight adjustments from their manual strategy application, which will be discussed in the next section.

Finally, with the indicators employed, all the associated files are called into a single StrategyLearner script. The script accepts an impact level, which dictates how much the trade will affect the market price, and contains two primary functions to generate a trading file containing buy, hold, and sell signals. The first primary function is the training stage which is called with `addEvidence()`, and it accepts a symbol, start, and end date. This function abstracts the training process of creating a random forest (bagged random decision tree) based on indicator values and price and/or volume movements. The following function, `testPolicy()`, then tests the random forest against sample indicator data, by abstracting the process of running the sample indicator values through the random forest to generate buy, hold, and sell signals. Secondary functions employed are `get_tree()`, which can alter the decision tree type (random or correlation-based), its leaf size, and the number of bags for the bagger; `df_yTrain()` which generates an optimal trading strategy for training the indicators against; `feature_balance()`, which checks the number of features used across all trees in the forest; and `author()`, to engrave my name in software.

Indicator Adjustments:

Beginning with the indicators, the first challenge is to convert crossover signals (BB, MACD, and STOSC) from the previous project into a continuous value for use in the random forest. Since the crossover's were previously identified as a zero or one to indicate no crossover and a crossover respectively, this approach is not be applicable for use in a regressive random forest. Thus, the crossover's are changed to a fractional approach, dividing the signal by its indicator value. This converts the original binary approach to a percentage of crossover, indicating the degree to which it is about to crossover, or has passed the crossover. For BB's, a crossover event is identified as the price movement crossing the upper or lower bands, which is converted to price/upper_band, and lower_band/price (inverted to maintain a constant 1.00 crssing). The MACD, which measures momentum by comparing two price averages, is converted to MACD_Signal/MACD, where the signal is a moving average of the MACD. The STOSC (K) is also a momentum indicator, which compares the current price to a windows high's and low's, where it's signal (D) is a similar three-day moving average of the indicator, which is converted to D/K.

The other adjustment that needs to be made is normalization of the SMA to make it independent of the magnitude of the price, and strictly on the size of the daily differential. This is employed by

dividing the SMA by the adjusted close price. The CCI and OBV did not need any additional modifications.

Trading Indicators:

With the indicators adjusted to suit the random forest, the action signals buy/hold/sell need to be generated to train the indicators against. This is done in a similar fashion to the BestPossibleStrategy approach in the manual strategy project, however, the market impact is implemented as an effective threshold factor. With the impact at zero, trades are made in response to all price shifts from day to day, which is determined with the `DataFrame.diff()` method. If a price falls, indicated by a negative value from `DataFrame.diff()`, a sell signal is generated, and vice-versa for a positive value. If there is no price movement, a hold signal (0) is generated. The market impact comes into play by creating a threshold that the price difference must cross in order to trigger the buy or sell signal, which is a percentage of the current price. If the threshold is not surpassed, the result is a hold signal.

Training and Testing:

With the trading indicator data, we can now generate our random forest! Our random forest generates multiple random decision trees based on mappings of the indicator values to the trading indicators discussed in the two paragraphs above. As an additional note, the training phase occurs after a time-frame characterized by the window attribute, which is dictated as the largest training window needed for all the indicators. This is employed to prevent the decision trees from training on indicator data that is np.nan as a result of the window size not being met.

Once training is complete, the result is a forest which can accept indicator values and respond with trading actions. Since the resulting values are averaged together, as a result of running the values against multiple trees, the resulting values are bound continuously between -1 and 1. To standardize the values to -1, 0 and 1 (sell, hold, buy), the values (y) are rounded to the domains of $y <= -0.5 \equiv -1$, $y >= 0.5 \equiv 1$, and $-0.5 < y < 0.5 \equiv 0$.

The selection criteria for the final indicators is rather crude, where the foundational four from the previous project are employed and tested for an out of sample error (to surprisingly good success), then the new indicators are added one by one to test their impact on the out of sample error and comparative returns against various random seed values. Through multiple trials, it appears that added indicators, CCI, normalized SMA, and OBV, increases the volatility of the portfolio performance with minimal ceiling gains. As a result, only BB’s, MACD, and STOCH are incorporated.

Experiment 1:

The first of two experiments is a comparison between our new random forest strategy trader, our fabled manual strategy trader, and the benchmark, which is 1000 shares held from day one. The criteria for this comparison is that the exact same indicators and window/threshold values is used, which isolates the comparison purely to the trading criteria method. The comparison is performed on the JP Morgan (JPM) ticker, with in sample training dates of 01 January 2008 to 31 December 2009. After ten trials, the average return of the strategy learner is 1.70, whereas the manual and benchmark is 1.45 and 1.01 respectively! The returns from each trial for the strategy learner is shown below, along with their associates plots. Both the benchmark and manual learner do not vary since their performance is not randomized.

Strategy Learner
1.801255
1.837016
1.739086
1.556408
1.603555
1.765667
1.651359
1.569182
1.549384
1.893279

Table 1: Experiment 1 Strategy Performance

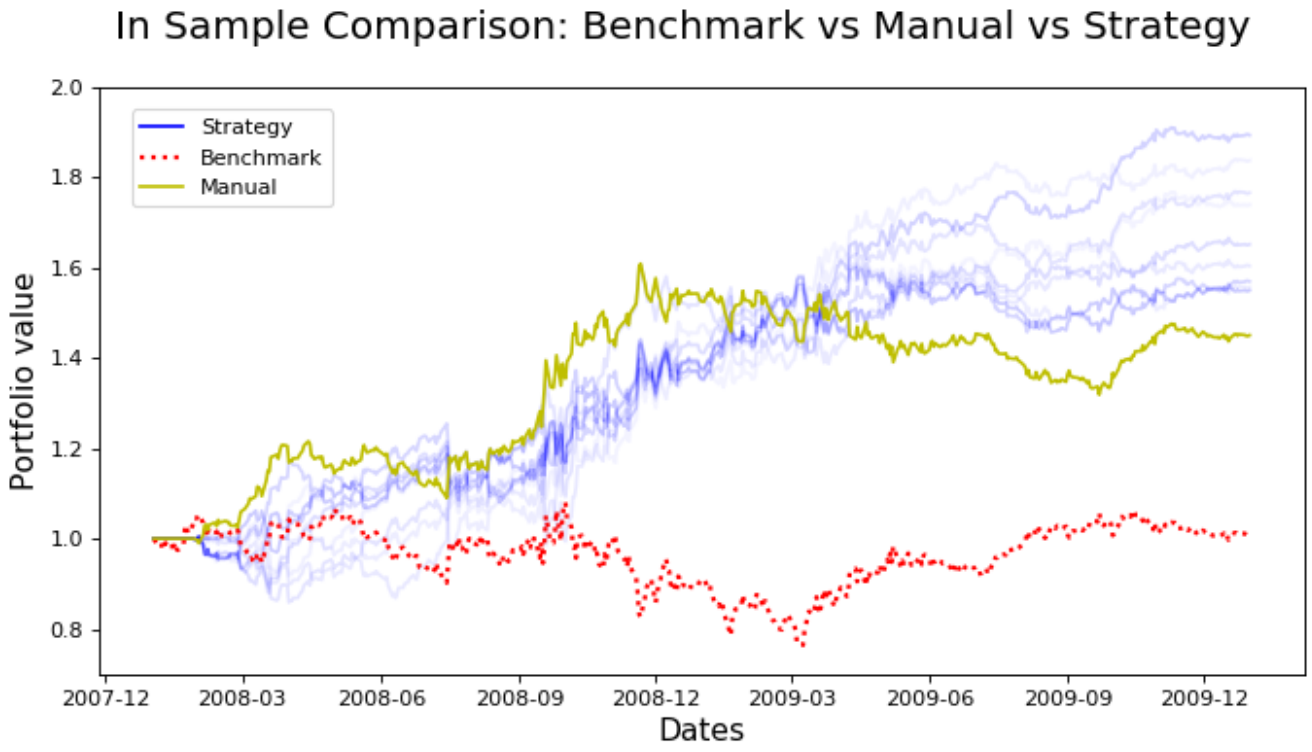

Figure 1: Experiment 1 Comparison

From the data above, we see that the strategy learner performs consistently better than the manual strategy on all ten trials. The boundaries of the performance metrics for the strategy learner is 1.54 and 1.84, which is pretty fantastic. With the superimposed graph it is interesting to see that nearly all of the trials have a consistent shape near the middle of the plot at around December 2008, and then separates towards the final date. What is also interesting is that during the consistent pinch at the middle, the manual strategy actually out performs the strategy learner, but loses momentum and plateaus while the strategy learning continues to increase. The benchmark is pretty consistent at not making money. From this plot we can conclude that the strategy learner is consistently better at training the data. While for these ten trials, it appears that the strategy learner is always better, it is very possible for it to perform worse than the manual learner, but is very unlikely. This is due to the randomized processing of best factors to split on for the random decision tree, and since randomness is inherent to the process, it could theoretically randomly generate a bad tree, multiple times, and create an effectively dead forest. However, that situation is highly unlikely, and the use of a forest is designed to lower those variances. While ten points is not a significant amount of trials, the consistency of the results is compelling.

Experiment 2:

The last of the two experiments is strictly with the strategy learner, and it involves manipulating the impact value discussed earlier. As mentioned for the decision tree, it acts as an effective threshold for trading, which conceptually restricts trades to the degree of price differentials. While at first thought, adding a threshold to trades would seem detrimental since it prevents trading at every possible opportunity, it does potentially have the effect of reducing overfitting. As such, the threshold would prevent trading on insignificant price changes, which can be synonymously characterized as noise. If that is to occur, I imagine that a small amount of impact would be beneficial as it prevents the random forest from training on insignificant price changes. This will possibly lead to better performance and less variance. However, since impact is a percentage of the price, there is definitely a happy medium for where it operates best. The higher the impact value, the larger the price difference needs to be in order to execute an order, which means that less and less orders are executed. To test these hypotheses ten trials are once again taken for the same JPM ticker and dates as Experiment 1. Impact values of 0, 0.005, 0.02, 0.05, and 1.0 are used. The results are shown in the chart and figures below.

Trial	0	0.005	0.02	0.05	1
1	2.165913	1.492531	1.74132	1.972994	1
2	1.623083	2.05918	1.274573	2.268542	1
3	2.179969	1.423682	2.140425	2.01268	1
4	1.523954	2.196738	2.002498	1.883754	1
5	2.240259	1.765744	1.619222	1.630576	1
6	2.181031	1.838972	1.55399	1.88326	1
7	1.967675	2.085587	1.919348	2.02324	1
8	1.515469	1.276191	1.83663	2.141846	1
9	2.071965	1.742098	1.892878	1.935537	1
10	1.664782	1.746866	2.510143	2.337423	1
Average:	1.91341	1.7627589	1.8491027	2.0089852	1
Std Dev:	0.2823979067	0.2848969459	0.3215614179	0.1935156765008	0

Table 2: Experiment 2 Impact Comparison

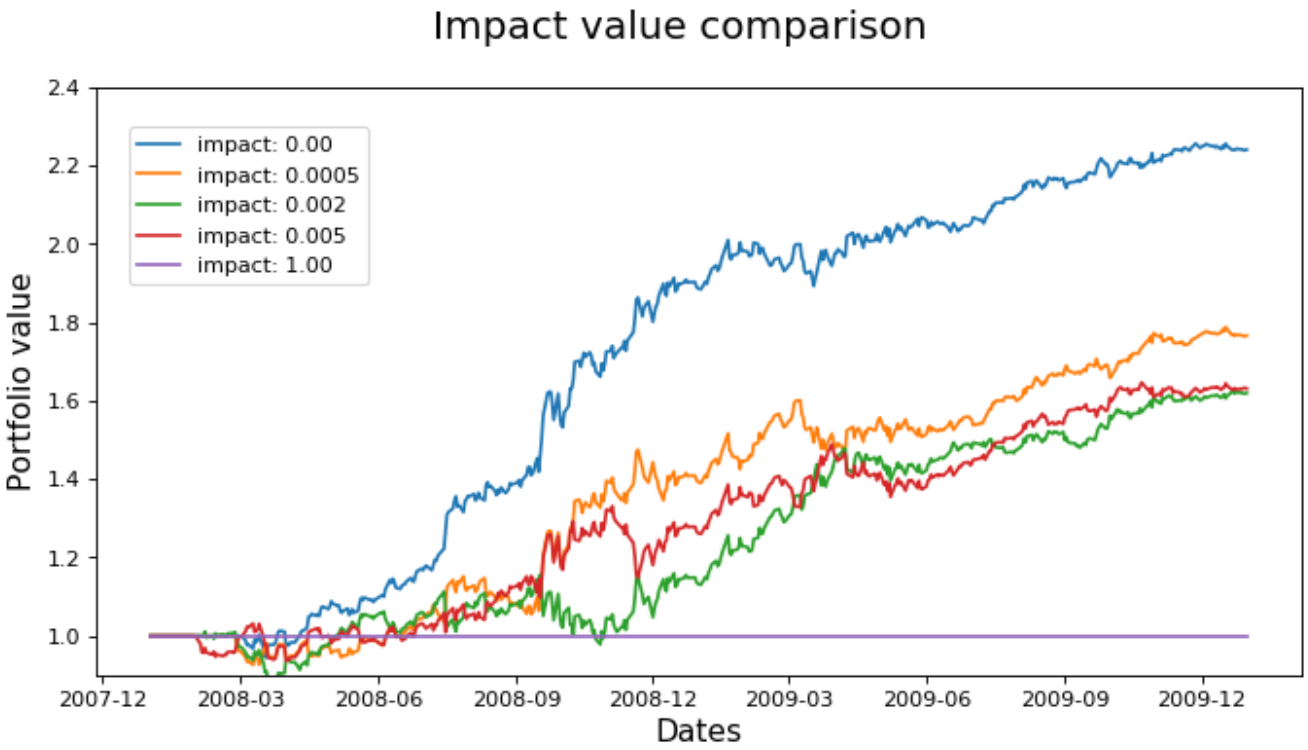

Figure 2: Experiment 2 Trial 5

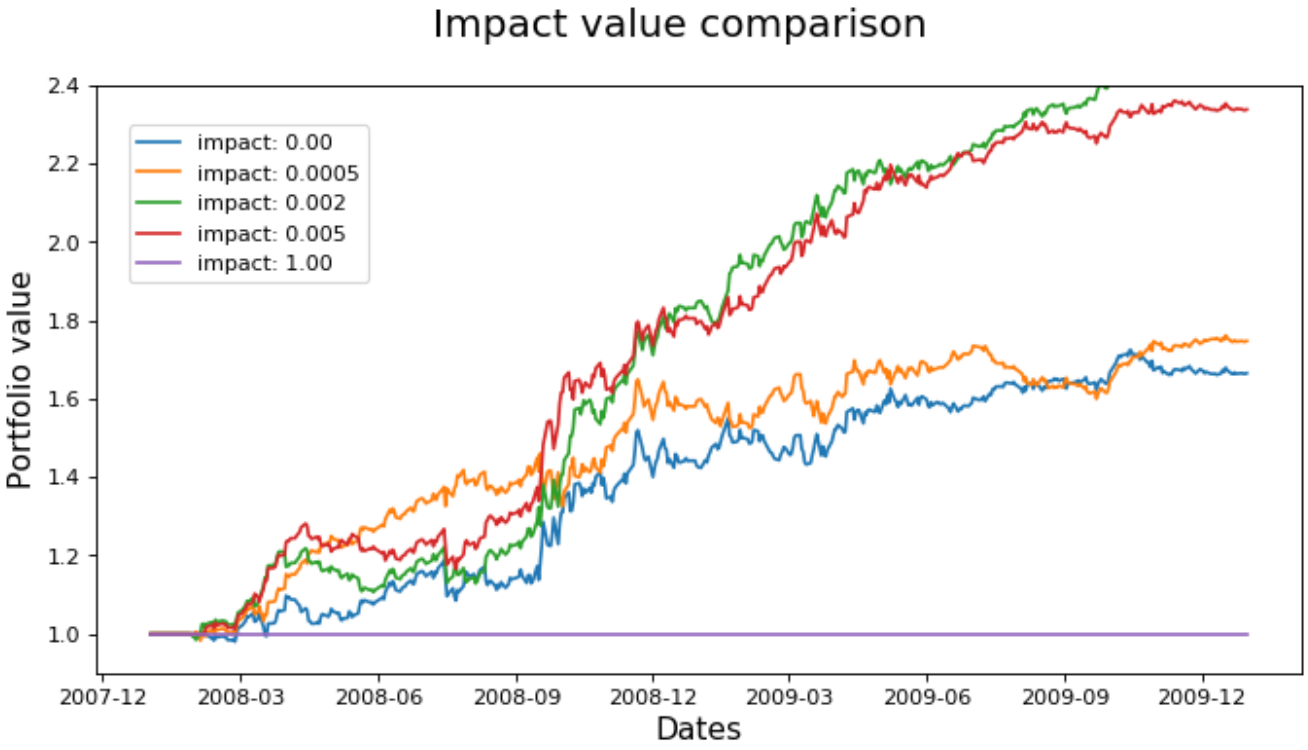

Figure 3: Experiment 2 Trial 10

From the chart, what immediately became interesting is the fact that the strategy learner for 0.00 impact, which is exactly the same as the one used Experiment 1, performed significantly better than before (average went from 1.54 to 1.913). The average is better than the best trial in Experiment 1! I'm not sure what caused this anomaly, but it is interesting to note.

Next, it does appear that my hypotheses on the affect of impact was correct for this small sample size. At an impact value of 0.05, we find that the average is highest, and the standard deviation is the lowest. Secondly, at 1.00, no trades are made as the threshold is too high to execute any orders. This result supports the concept that the impact value does affect both the variance and stability of the learner. The most likely cause is the reduction in training on small price changes. Secondly, is it important to identify that there appears to be a range of impact values that are beneficial. At values of 0.005 and 0.02, we find the worst average and standard deviation values of the bunch (excluding 1.00). From 0.02 to 0.05, we find a significant change in performance for a small change in the impact value. From the trials, we can support or hypotheses that the impact value can have an appreciable affect on the performance of our strategy learner, potentially increasing the robustness by reducing variance and overfitting, and the overall performance by implementing an effective noise filter on small price changes. We also discover however that this performance increase is extremely sensitive, as a change from 0.02 to 0.05 altered the performance of the trainer from one of the worst performers to the best for our experiment.

Conclusion:

From this project and associated experiments, we've found that the strategy learner is pretty much better overall in comparison with our manual learner, +1 to machine learning against humans. The conversion from the manual learner to a random forest was fairly trivial, and did not require many modifications. The results were pretty astonishing as the strategy learner preformed significantly better on our in sample data, and is fairly modular in that indicators can be added and removed easily. We've also found that the impact value can have an appreciable affect on the performance of the learner, but a narrow calibration of that value is required as it is very sensitive to small changes. What was not discussed however is the computational demands between the two trainers, the strategy learner takes significantly longer to process compared to the manual strategy, but luckily in the modern world the discrepancy for this small data set is insignificant. Lastly, I couldn't help it but to also perform a comparison of the learners on the out of sample data, and after ten trials it performed surprisingly well! The plot below displays fairly consistent out-performance of the manual strategy, but does occasionally do worse.

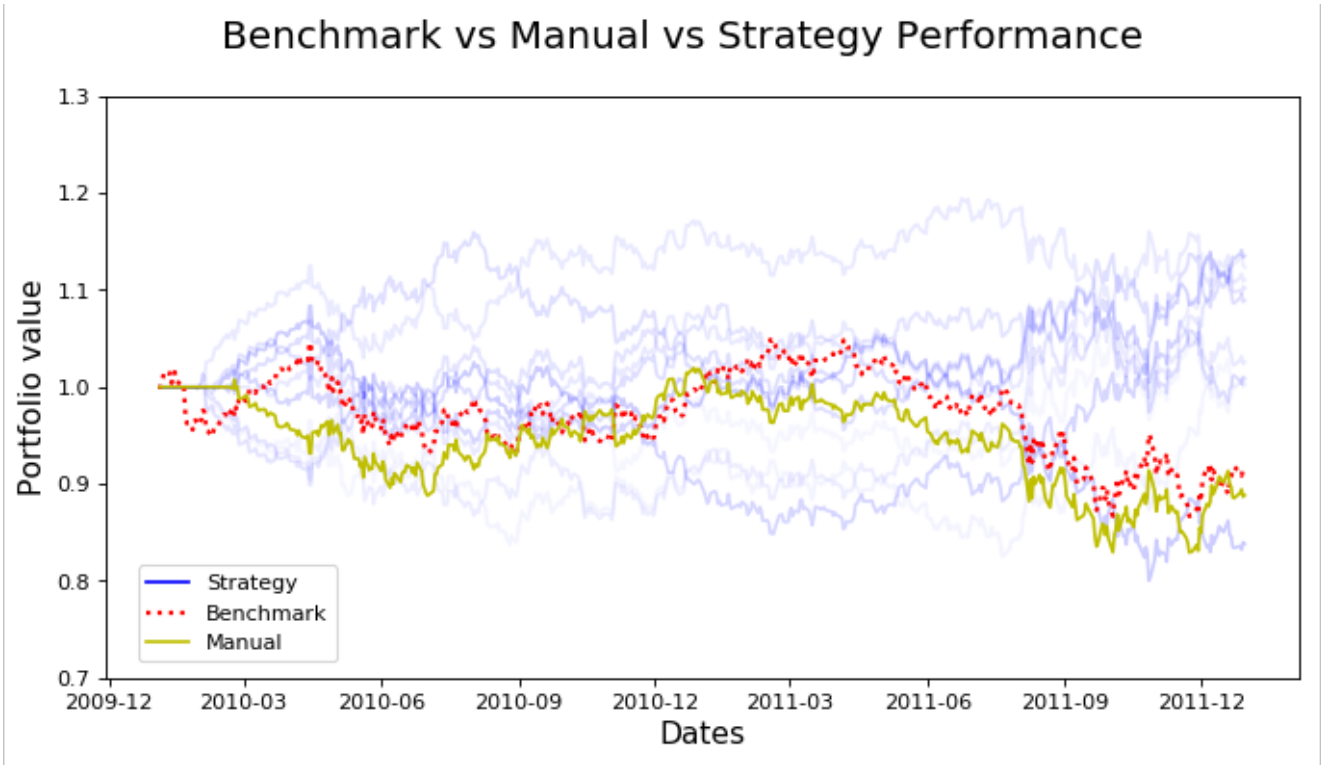

Figure 4: Out of sample comparison