CS7646 Machine Learning for Trading Assignment 8
Robert Bobkoskie Strategy Learner

INTRODUCTION

This project evaluated a regression or classification-based learner (SL) to develop a trading strategy. At the core of this strategy was a random forest learner that utilized three technical indicators[1] to classify trading actions: long, short, cash. For this assignment, these actions are equivalent to: buy, sell hold.

The indicators, which were derived from historical data, provide a signal for buying or selling a security. For this project the technical indicators that were utilized differed slightly for those used in Manual Strategy (MS). For Manual Strategy, I chose to implement: Simple Moving Average (SMA), Bollinger Bands® (BBI) and Chaikin Money Flow (CMF). However, during testing of the SL, I discovered that the security 'ML4T-220' was failing test cases because the CMF indicator was in error. Further investigation of the 'ML4T-220' data showed that the volume was constant throughout the in-sample-[period, thus causing the CMF indicator to fail. To fulfill the requirement to utilize three indicators, including one that was not discussed, I chose to develop the Relative Strength Index (RSI). To create a valid comparison between SL and MS, I modified Manual Strategy to account for the new indicator. The indicators provided the basis for developing a long-short strategy that was trained over an in-sample period and tested on an out of sample interval. The training strategy utilized a random forest of random trees with the following meta parameters:

```
self.learner = bgl.BagLearner(learner = rtl.RTLearner, kwargs={"leaf_size":10},
                              bags=30, boost=False, verbose=False)
```

For this report, the portfolio consisted of: a starting value of $100,000 in cash, and long/short positions in a single security, JPM. The in sample/development period is: January 1, 2008 to December 31, 2009. The out of sample/testing period is: January 1, 2010 to December 31, 2011.

To baseline the SL, a Benchmark (BM) portfolio was developed. The SL was also compared with the MS approach from the earlier project. As noted, the indicators were: RSI, SMA and BBI, where RSI is new to the MS. To account for this new indicator, the MS was modified accordingly. Both the BM and MS consisted of the same securities as the SL, however, the trading strategies were different. The BM portfolio employed a buy and hold strategy. That is, a long position (buy) was initiated on the first trading day of the interval and held until the end date of the interval. Recall the approach used the MS, the indicators were adjusted to optimize performance (Sharpe ratio and cumulative return) over an in-sample period and tested on an out of sample interval. Allowable positions for all strategies (SL, MS, BM) were: 1000 shares long, 1000 shares short, 0 shares.

I. Technical Indicators

This section discusses my implementation of ***indicators.py***. This file defined a function, ***indicators***, which applied three technical indicators over an interval of historical price data [sd, ed] for securities [syms] using a lookback widow of 14 days and returned a Pandas dataframe.

```
def indicators(syms, sd=dt.datetime, ed=dt.datetime, window=14)
```

The technical indicators I chose to utilize: Simple Moving Average (SMA), Bollinger Bands® (BBI) and Relative Strength Index (RSI) provided signals for a long/short trading policy.

Relative Strength Index (RSI): RSI is a momentum oscillator that charts the strength or weakness of a security based on the closing prices of a recent trading period. For this project, a 14-day trading period, or lookback window was used. RSI is typically measured on a scale from 0 to 100, with high and low levels marked at 70 and 30, respectively. However, for this project, I chose to normalize RS such that RSI oscillator between 0 and 1, thus the high and low levels range between 0.3 and 0.7. A security with stronger positive changes have a higher RSI than stocks which have had stronger negative changes. To implement RSI, I used a Pandas dataframe to calculate periods of gains and losses. I used these gains and losses to compute the Relative Strength (RS), and the RSI.

```
up_ret = daily_ret[daily_ret >= 0.].fillna(0.).cumsum()
up_gain = prices.copy()
up_gain.iloc[:,:] = 0.
up_gain.values[window:,:] =   up_ret.values[window:,:] -
                              up_ret.values[:-window,:]

dn_ret = -1*daily_ret[daily_ret < 0.].fillna(0.).cumsum()
dn_loss = prices.copy()
dn_loss.iloc[:,:] = 0.
dn_loss.values[window:,:] =   dn_ret.values[window:,:] –
                              dn_ret.values[:-window,:]

rs = (up_gain / window) / (dn_loss / window)
rsi = 1 - (1 / (1 + rs))                       # RSI range [0, 1]
```

Simple Moving Average (SMA): This indicator smoothes out price action by filtering out the "noise" from daily (random) price fluctuations. I used SMA to identify a trend in a stock's price movement. I developed the SMA indicator by obtaining the [quotient of the portfolio price over the mean price of the portfolio over the lookback window] minus one. To utilize the indicator over the in and out-of-sample trading intervals, I used the Pandas rolling mean function. SMA is an oscillator that fluctuates between -0.5 and +0.5.

```
sma = (prices.loc[:,'PORTFOLIO'] /
      prices.loc[:,'PORTFOLIO'].rolling(window=window,
      min_periods=window).mean()) - 1
```

Bollinger Bands® (BBI): BBI is an indicator of volatility and ranges between +-2 standard deviations of the SMA. I used BBI to identify overbought, or oversold conditions in a security. I developed BBI by obtaining the quotient of SMA over two standard deviations of the portfolio over the lookback window. To utilize the indicator over the in and out-of-sample window, I used the Pandas rolling standard deviation function. BBI is an oscillator that fluctuates between -1 and +1.

```
bbi = (prices.loc[:,'PORTFOLIO'] - indicators_df.loc[:,'RM']) /
      (prices.loc[:,'PORTFOLIO'].rolling(window=window, center=False).std() *2)
```

II. Strategy Learner

This section describes my approach for creating an agent to learn a trading strategy by classifying actions [buy, sell, hold]. Learning was addressed through a Bag Learner API, which in turn utilized a random tree learner for classification.

The indicators described in ***Section 1*** comprised the set of attributes (features), **Xi** that were used for training. Since the learning involved a decision tree, normalization of the features was not necessary. However, the Xi were scrubbed of NaN values by slicing the dataframe to exclude data in the window (recall, `window=14`).

```
df = idr.indicators(syms, sd, ed, window)
df = df.iloc[window-1:,:]
df.fillna(0,inplace=True)
```

The actions **Yi** (for training) were determined by calculating an N-day reward (R) based on the historical adjusted closing price data. The learner performed well with N=2, thus, all experiments in this project used that value. Actions were decided based on a conditional that evaluated if R was greater than ***IMPACT*** (buy) or if R was less than negative ***IMPACT*** (sell), else hold. Experiments 2 will discuss how modifying ***IMPACT*** influences the strategy.

```
for idx, day in enumerate(prices.index[:-N]):
      R = (prices.values[idx+N] - prices.values[idx]) / prices.values[idx]
      if R>IMPACT:
            Y[idx] = self.buy
      elif R<-IMPACT:
            Y[idx] = self.sell
      else:
            Y[idx] = self.hold
```

III. Strategy Learner: Experiment 1

The objective for this experiment is to consider the 'consistency' of learning for SL. To gauge consistency, I ran experiments that evaluated performance over different in-sample periods and considered different securities. For this experiment, the learner did not consider

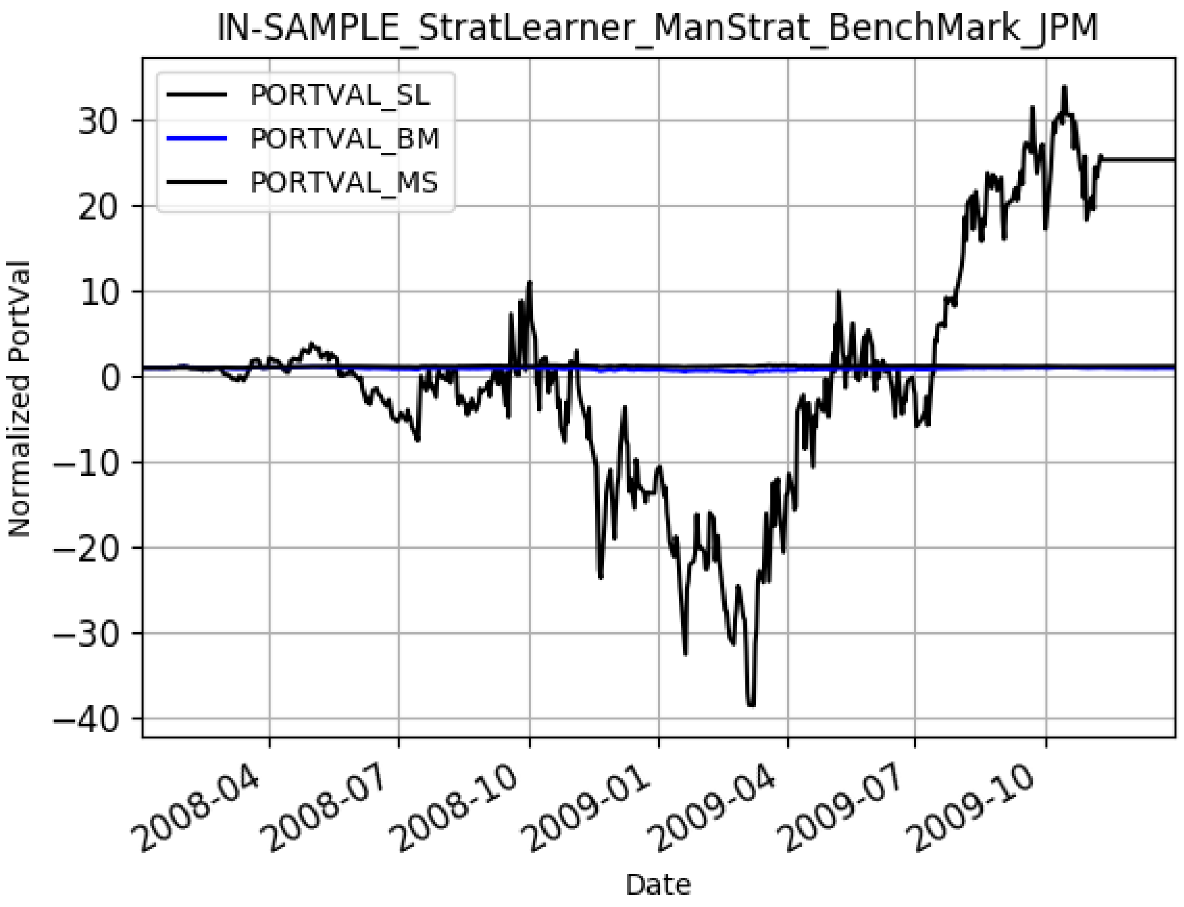
IMPACT, or commission when developing a strategy. However, the statistics from the market simulator did account for transaction costs: impact=0.005, commission=9.95 for all portfolios [SL, BM, MS]. Performance was captured in ***Table 1*** for these three scenarios:

1. SL, MS (and BM), utilized the same parameters: security 'JPM', train = test sample periods. Repeat three times.
2. SL, MS (and BM), utilized the same parameters: security 'JPM', train' = test' sample periods. Repeat three times.
3. SL, MS (and BM), utilized the same parameters: security 'AAPL, train' = test' sample periods. Repeat three times.

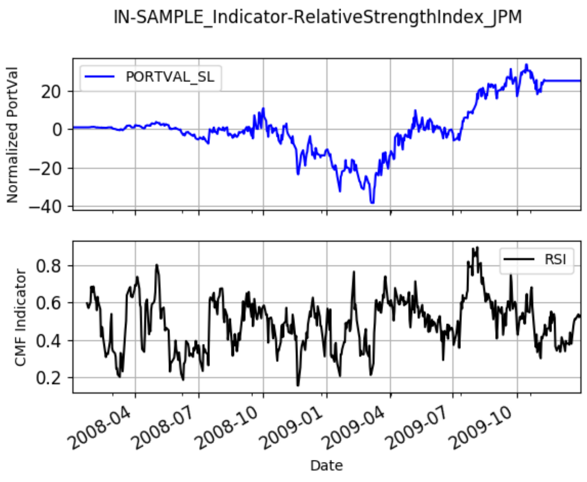
Table 1: Data collected for three runs of ***experiment1.py***. The data is concatenated in the cells corresponding to the experiment.

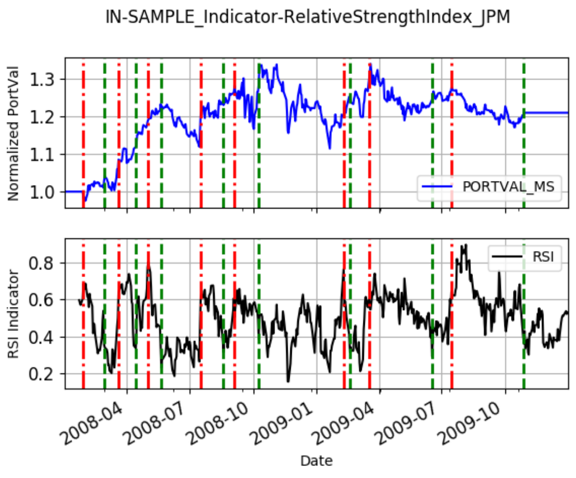
STRATEGY (Security) [Train] [Test]	**CR**	**ADR**	**SDDR**	**SR**	**# Trades**
SL (JPM) [2008-01-01, 2009-12-31] [2008-01-01, 2009-12-31]	24.31761 26.33193 15.75002	0.46063 0.36202 0.08932	6.37830 5.94874 5.27157	1.14644 0.96607 0.26896	159 185 193
BM (JPM) [2008-01-01, 2009-12-31] [2008-01-01, 2009-12-31]	-0.02565 -0.02565 -0.02565	0.00067 - -	0.038418 - -	0.27998 - -	1
MS (JPM) [2008-01-01, 2009-12-31] [2008-01-01, 2009-12-31]	0.21003 0.21003 0.21003	0.00052 - -	0.013703 - -	0.607056 - -	16
SL (JPM) [2010-01-01, 2011-12-31] [2010-01-01, 2011-12-31]	-21.74692 -23.62684 -24.73717	-6.86051 -2.39991 -2.28657	146.51306 54.80323 38.90517	-0.74334 -0.69517 -0.93299	179 171 175
BM (JPM) [2010-01-01, 2011-12-31] [2010-01-01, 2009-12-31]	-0.21776 -0.21776 -0.21776	-0.00031 - -	0.018715 - -	-0.26576 - -	NA
MS (JPM) [2010-01-01, 2011-12-31] [2010-01-01, 2011-12-31]	-0.05814 -0.05814 -0.05814	-0.00091 - -	0.00842 - -	-0.17269 - -	NA
SL (AAPL) [2010-01-01, 2011-12-31] [2010-01-01, 2011-12-31]	178.78344 188.95870 174.66063	0.02930 0.02155 0.01155	0.54016 0.43476 0.43446	0.86114 0.78706 0.42206	113 123 127
BM (AAPL) [2010-01-01, 2011-12-31] [2010-01-01, 2009-12-31]	3.83546 3.83546 3.83546	0.00427 - -	0.04751 - -	1.42678 - -	NA
MS (AAPL [2010-01-01, 2011-12-31] [2010-01-01, 2011-12-31]	2.04114 2.04114 2.04114	0.00355 - -	0.04735 - -	1.19273 - -	NA

Figure 1: Corresponds to the first run, the orange shaded data in ***Table 1*** where: SL-CR=24.31761, BM-CR=-0.02565, MS-CR=0.210035 and TRADES=[SL:159, BM:1, MS:16].



Figures 2 and 3: Depict RSI, the new indicator used for SL and MS respectively.





From the results in ***Table 1***, it is evident that the SL does not produce the similar performance when run multiple times over the same in-sample period with the same security. This is verified when evaluating performance for SL over different in-sample periods and/or with a different security. Contrast the inconsistent results of SL with the consistent performance for MS, which had identical performance for sequential run over all three experiments. This agrees with the hypothesis that decision trees are unstable, where a small change in the data can lead to a large change in the structure of the decision tree and thus repeated runs may not produce the same decisions, hence different outcomes.

IV. Strategy Learner: Experiment 2

The objective for this experiment is to consider how ***IMPACT*** influences the SL. To gauge ***IMPACT***, I set up the experiment to measure the number of trades and performance over a range of values for ***IMPACT***: [0.00, 0.15, 0.20, 0.25, 0.26, 0.27]. Recall from Section 2, that the SL classified (learned) decisions based on ***IMPACT***:

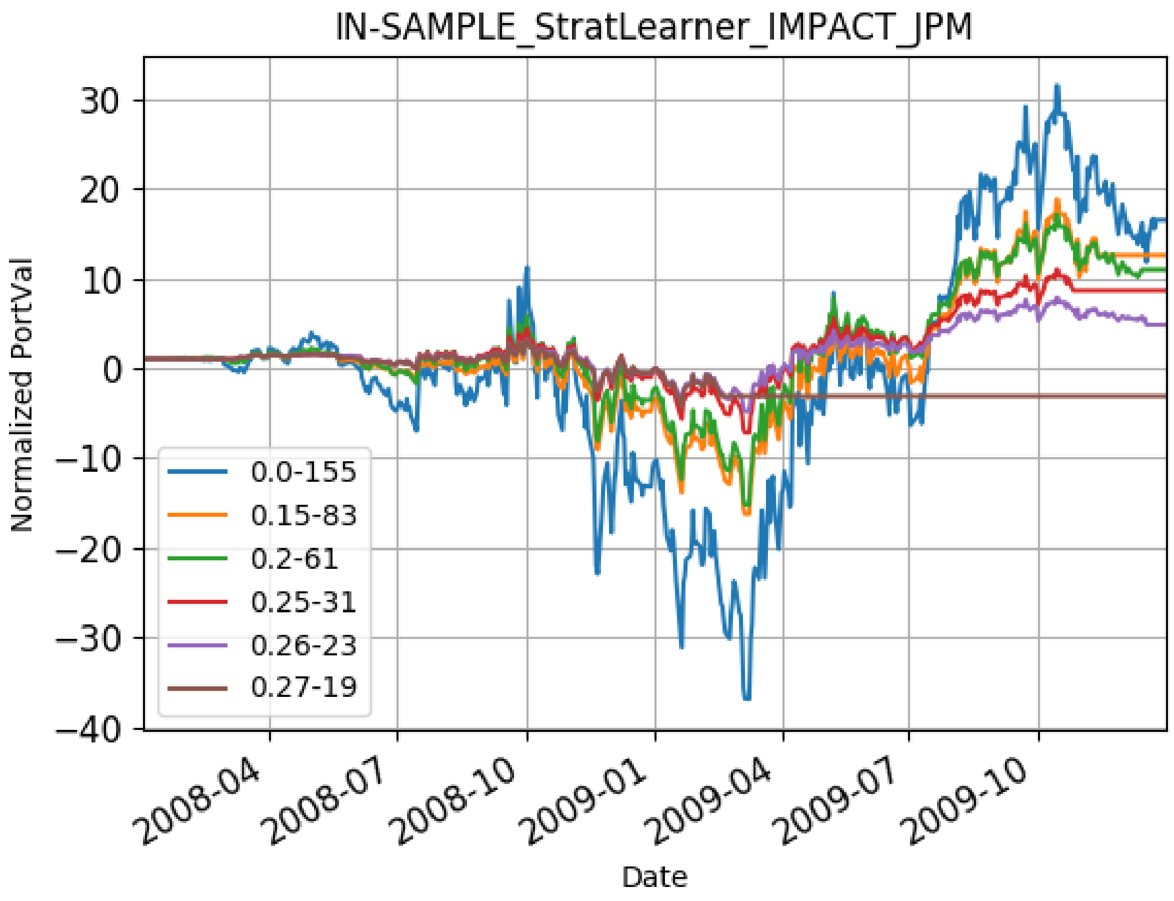
- If R was greater than ***IMPACT*** (buy)
- Or if R was less than negative ***IMPACT*** (sell)
- Else hold

To study ***IMPACT*** directly, and simplify the experiment, I did not create the learner to make decisions on the traditional value/range for ***IMPACT***, which is defined by basis points (1 basis point=0.01%=0.0001). This approach eliminated the need for an additional ***THRESHOLD*** parameter when the learner considers the reward for predicting actions, i.e., ***`if REWARD - IMPACT < THRESHOLD, then buy`***.

Table 2: Data collected for ***`experiment2.py`***, where the ***IMPACT*** takes the values [0.00, 0.15, 0.20, 0.25, 0.26, 0.27]. The data is concatenated in the cells corresponding to the strategy (SL, BM, MS) and was collected using **(JPM)** over in-sample **[2008-01-01, 2009-12-31].**

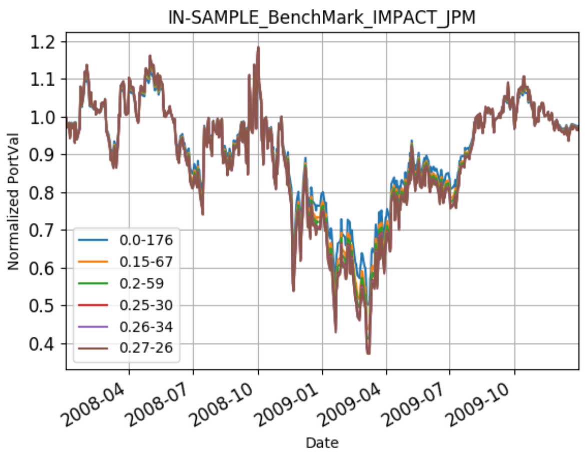
STRATEGY	IMPACT	CR	ADR	SDDR	SR	# Trades
SL	**0.00**	15.50942	-0.38305	5.0387	-1.20682	155
SL	**0.15**	11.58765	0.63934	17.40507	0.58311	83
SL	**0.20**	9.96287	-0.09377	1.39437	-1.06764	61
SL	**0.25**	7.65516	0.20970	4.36094	0.76334	31
SL	**0.26**	3.82206	0.15896	3.55997	0.70887	23
SL	**0.27**	-4.11815	-0.85005	10.02883	-1.34553	19
BM	**0.00**	-0.02555	NA	NA	NA	NA
BM	**0.15**	-0.02888				
BM	**0.20**	-0.03019				
BM	**0.25**	-0.03163				
BM	**0.26**	-0.03194				
BM	**0.27**	-0.03225				
MS	**0.00**	0.26203	NA	NA	NA	NA
MS	**0.15**	-1.39821				
MS	**0.20**	-2.00182				
MS	**0.25**	-2.63343				
MS	**0.26**	-2.76328				
MS	**0.27**	-2.89436				

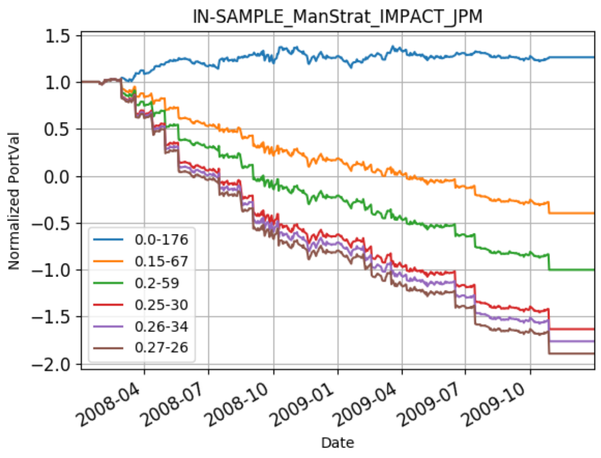
Figure 4: Corresponds to the SL statistics where ***IMPACT*** takes the values [0.00, 0.15, 0.20, 0.25, 0.26, 0.27], the orange shaded data in ***Table 2***.



Figures 5 and 6: Corresponds to the statistics where ***IMPACT*** takes the values [0.00, 0.15, 0.20, 0.25, 0.26, 0.27].

- BM is the light-green shaded data in ***Table 2***.
- MS is the light-blue shaded data in ***Table 2***.





Observing the data in both ***Table 2*** and plot in ***Figure 4***, one can conclude that ***IMPACT*** has a strong influence on learning actions and a high correlation with the number of trades and performance. Note the inverse relationship between ***IMPACT*** with both the number of trades and performance. As ***IMPACT*** is increased, both the number of trades and performance decrease. The plot in ***Figure 4*** also clearly shows this relationship. The labels in ***Figure 4*** correspond to the: *<IMPACT Value>-<#Trades>*. It appears that as ***IMPACT*** is increased, the portfolio returns are dampened. In fact, the line corresponding to the highest ***IMPACT***, ***0.27-19*** is nearly straight, while the line associated with zero ***IMPACT***, ***0.0-155*** has the most variance, i.e., the lowest lows and highest highs. The others fall mostly in an orderly fashion between [***0.0-155***, ***0.27-19***].

The results from ***experiment1.py*** agree with the hypothesis that considering an ***IMPACT*** value when making decisions will affect the number of trades. That the performance would also be correlated with ***IMPACT*** was unexpected.

Through rote learning, ***IMPACT*** also influences the performance of MS. This is different than the ***IMPACT*** on SL. For all portfolios, ***IMAPCT*** will decrease returns, as shown in ***Table 2*** and ***Figures [4, 5, 6]***.

V. Conclusion

For experiment 1, the hypothesis was that creating a trading strategy with classification learner would not produce consistent results over repeated runs. In fact, to achieve consistency, a seed needs to be planted to tame the randomness of the classification. The results from ***experiment1.py*** clearly point out the inconsistent nature when applying a classification learner to make decisions.

From ***experiment2.py***, the ***IMPACT*** on the learner's classification of actions was evident. As the ***IMPACT*** was increased, the number of trades and performance decreased. Increasing the ***IMPACT*** appeared to influence the learner's classification of actions, i.e., learning appeared dampened, along with performance.

Interestingly, one can tie ***experiment1.py*** and ***experiment2.py*** together by observing the performance of SL to MS (***Table 2***) relative to ***IMPACT***. For **(JPM)** over in-sample **[2008-01-01, 2009-12-31]**, the performance of SL was superior to MS for all values of ***IMPACT***, except for the highest ***IMPACT*** value of 0.27.

[1] https://en.wikipedia.org/wiki/Technical_indicator