Machine Learning Based Financial Stock Trading Engine

Introduction

The purpose of this project is to develop a Machine Learning based financial stock trader that learns trading strategies from historical stock performance and financial technical indicators data and makes trading recommendations for a given stock portfolio using classification algorithms. The project also develops a manual stock trader based on rule-based, hand-coded logic using the same technical indicators to highlight their power. The performance of both the manual and AI-learning algorithms are then compared to a naive benchmark trader that simply employs a "buy-and-hold" strategy for the duration of the evaluation period.

While both strategies utilize technical trading indicators (Technical Indicators) as a basis for predicting profitable trading opportunities. the manual strategy relies on interpreting trading signals from indicators, and the ML strategy uses Random Forest classification. In both cases, relevant "hyper parameters" are tuned to achieve optimal in-sample performance while minimizing "overfitting".

Experiments are conducted to a) compare their relative performance against a "benchmark" portfolio, and b) understand the effect of varying market impact.

It is hypothesized that the ML trader will perform better than the Manual trader in-sample. Relative performance out-of-sample remains to be seen.

Indicator Overview

Technical Indicators form the core of a technical analysis strategy to predict trading markets. They are mathematical calculations based on historical trading metrics such as prices, trade volumes, etc. ('Technical Indicator', Wikipedia)

The following three have been selected for use in this project:

- **Relative Strength Index** (RSI)

- **%Bollinger Band** (%B)
- **Stochastic Oscillator** (STO)

All three indicators are implemented in a single Python program and return continuous real-valued single-vectors organized by date. Project 6 describes the indicators in detail including their strengths, formulae, and implementation. The choice of indicators was influenced by simplicity, reliability, and diversity. For example, RSI/STO are both momentum indicators, and can confirm each other, as well as combine well with a moving-average based indicator (%B).

Table 1 summarizes the lookback period (common to both manual and strategy learners), parameters and thresholds for each of the three indicators.

	LOOKBACK PERIOD	SIGNAL GENERATION & THRESHOLDS	
	DAYS	**MANUAL TRADER INTERPRETATION**	**STRATEGY LEARNER**
RSI	14	< 30 **(Go Long)** > 70 **(Go Short)**	Learnt
%BB	20	Prior day < 0 <= Current Day <= 0.5 **(Go Long)** Prior day > 1 > Current Day >= 0.5 **(Go Short)** Prior day < 0.5 <= Current day **(Exit Long Position)** Prior day > 0.5 >= Current Day **(Exit Short Position)**	Learnt
STO	14	< 20 **(Go Long)** > 80 **(Go Short)**	Learnt

Table 1. Indicator Lookback and Optimization Parameters

Manual Strategy

The Manual Trader was implemented as a Python class, setting the above indicator thresholds. For the given symbol and date range, it retrieves result vectors from each indicator and infers independent long/short entry signals by comparing them

to the corresponding threshold settings (see Table 1). A majority voting system is used to combine the indicators to an overall signal as below:

- **Enter Long** Position when any two indicator values are inferred as "long" signals.
- **Enter Short** Position when any two indicator values are inferred as "short" signals.
- **Exit Long** Position when %B indicator breaks "over" the BB center line (%B=0.5), when measured between successive trading days
- **Exit Short** Position when %B indicator breaks "below" the BB center line (%B=0.5), when measured between successive trading days

Note - actual long or short entry trades are only made if project holding constraints are satisfied (Zero, 1,000 Long or 1,000 Short), and number of shares per trade restricted to +/- 1000 or +/- 2,000 (if needed).

The above strategy is effective since they are proven reliable indicators that work well together and representative of oversold/overbought markets. A majority voting system ensures that trades are only made based on at least two confirmations. Further, optimal indicator thresholds were determined by an offline tuning program running in iterative loops to test various combinations within the in-sample training range for JPM.

Comparison of Manual Strategy Trader with "Benchmark" In-Sample

- A "test project" Python program was written to run the Manual Strategy trader with JPM symbol for the in-sample period of 1/1/2008 to 12/31/2009.
- Benchmark results were obtained by buying 1,000 JPM shares on 01/01/2008 and holding through 12/31/2009.
- The manual trader beat benchmark in-sample returns as shown below:

	CUMULATIVE RETURNS	STANDARD DEVIATION	AVERAGE DAILY RETURNS	SHARPE RATIO
MANUAL STRATEGY	0.165408	0.005970	0.000321	0.854766
BENCHMARK	0.012325	0.017041	0.000169	0.157205

Table 2. Manual Trader v/s Benchmark In-Sample Results

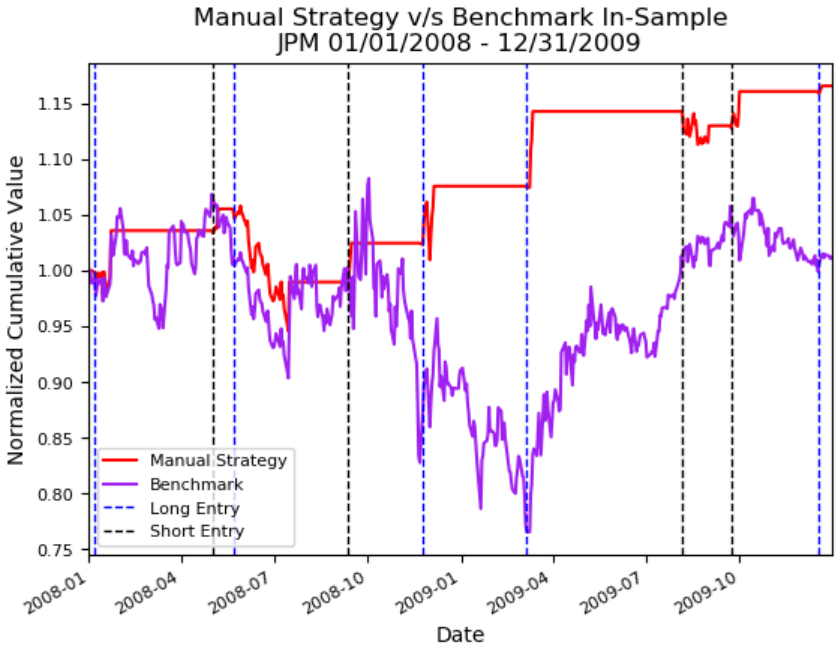

Figure 1. Manual Trader v/s Benchmark In-Sample Comparison

Comparison with "Benchmark" Out-Of-Sample

- By freezing all parameters after in-sample training, the manual trader was *tested* out-of-sample with JPM for the period of 1/1/2010 to 12/31/2011.
- Benchmark results were obtained by buying 1,000 JPM shares on 01/01/2010 and holding through 12/31/2011.
- Manual trader beat benchmark results as indicated in Table 3 and Figure 2
- It is observed that out-of-sample performance is worse than in-sample. This is primarily because optimal indicator thresholds were tuned only for in-sample data. Stock price trend for an out-of-sample time period doesn't necessarily follow in-sample, and therefore corresponding trades are not as profitable.

	CUMULATIVE RETURNS	STANDARD DEVIATION	MEAN OF DAILY RETURNS	SHARPE RATIO
MANUAL STRATEGY	0.028597	0.004091	0.000064	0.249838
BENCHMARK	-0.083579	0.008500	-0.000137	-0.256657

Table 3. Manual Trader v/s Benchmark Out-Of-Sample Results

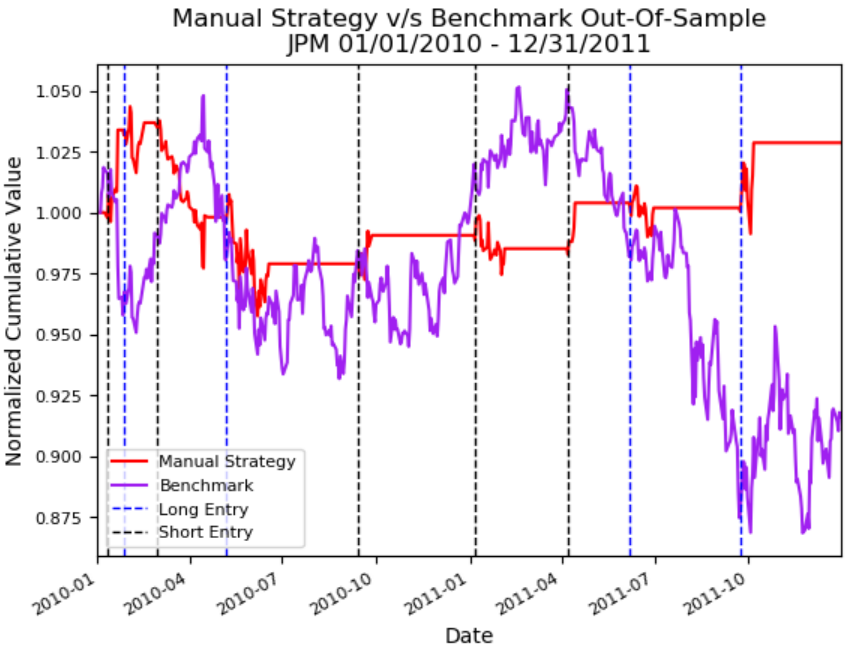

Figure 2. Manual Trader v/s Benchmark Out-Of-Sample Comparison

Strategy Learner

The Strategy Learner was implemented as a Random Forest Classification learner. The trading problem was framed as an ML Classification problem by:

- Use indicator values for a known past "training" period as "features".
- Compute cumulative returns for each training day for an initially chosen "N" day period, including transaction costs.
- Choose initial thresholds to compare the N-day returns and classify label to indicate long/short/hold (close) trading signals.
- Build training model using the above feature-label dataset.
- Test model in-sample using cross-validation techniques, measure overall cumulative returns, and maximize by tuning various hyperparameters.

The following hyperparameters were used to tune the model:

1. Number of days to predict a rate of return – "**N**".

2. Minimum acceptable thresholds for N-day return rates for making entry/exit trades – referred to as "**YBUY**" and "**YSELL**".
3. Number of individual random tree learners used for the forest - "**Bags**".
4. Maximum number of "leaves" that can be aggregated– "**Leaf Size**".
5. Strategy for interpreting a "no recommendation" situation when predicted N-day returns do not meet criteria for long or short market entries; referred to as - "**Exit Strategy**". Two possible strategies are:
    - "Close" current positions (buy/sell as needed to liquidate) OR
    - "Hold" current positions and wait for long/short signals.

Optimal hyperparameter values were obtained by writing an offline tuning program that iteratively varied the hyperparameters and maximized for cumulative returns for the 5 symbols in this project (JPM, ML4T-220, AAPL, SINE_FAST_NOISE, UNH). Initial "guesses" were based on prior experience or observing price movement trends (e.g., wavy ML4T-220 prices).

```
for symbol in ['JPM', 'AAPL', 'ML4T-220', 'SINE_FAST_NOISE', 'UNH']:
    for ybuy in np.arange(0.005, 0.055, 0.005):
        for ysell in np.arange(-0.005, -0.055, -0.005):
            for exit_strat in ['C', 'H']:
                for N in range(5, 15, 1):
                    for bags in range(10, 41):
                        for leaf_size in range(5, 11):
                            # instantiate strategy learner
                            # set hyperparameter current values
                            # add evidence using in-sample crossvalidation training date range
                            # test policy using in-sample crossvalidation testing date range
                            # compute and write portfolio returns for analysis
```

Figure 3. Random Forest Strategy Learner Hyperparameter Tuning Program

With an observed run time of ~1 second/iteration, the estimated run time was about 14 days, which was impractical. Introspection of price movement patterns provided suggestions that were confirmed running "narrower" program versions. Smaller cross-validation time series datasets were used to further reduce program runtime. The following hyperparameter values were determined as optimal:

Exit Strategy ("Hold"); N (5 days); YBUY (0.030); YSELL (-0.030); Bags (40); Leaf Size (10)

Data adjustment was required to convert the N-day cumulative returns to discrete Buy/Sell/Hold (or Exit) values as labels. These were translated to +1/-1/0 respectively, and the classifier trained against these discrete values.

Experiment 1 (Manual Strategy / Strategy Learner)

The purpose of this experiment was to compare the relative performances of the Manual Strategy and Strategy Leaner traders. The experiment was coordinated through a standalone Python program that trained the Strategy Learner with in-sample JPM data (1/1/2008 to 12/31/2009), and then separately tested against both in-sample and out-of-sample dates (1/1/2010 to 12/31/2011). It also queried the Manual Strategy Trader for in and out-of-sample dates and additionally retrieved benchmark results for both periods through a convenience method in Manual Strategy. Trading costs (impact/commission) were included. Previously tuned hyperparameters were retained, with common indicator code base and settings.

The initial hypothesis is that the strategy learner will significantly outperform both manual strategy and benchmark for in-sample. Out-of-sample testing is difficult to predict without future peeking, but lower returns expected with both traders.

Normalized cumulative portfolio returns were computed for each set of trades obtained for Manual Strategy, Strategy Learner, and Benchmark through MarketSimulator code from prior projects. Figures 4 & 5 plot their performance.

In-Sample Testing

- As expected, Strategy Learner significantly outperformed both manual strategy and benchmark, confirming the strengths of machine learning based trading techniques when queried against previously trained data.
- Manual Strategy outperformed benchmark moderately, confirming the ability of the technical indicators to reasonably predict market conditions.
- For a given set of prices, parameters, and configurations, both Manual Strategy and Benchmark are deterministic. The strategy learner has some amount of randomness due to use of Random Tree, so its results are not

always exactly same even in in-sample testing. However, with in-sample testing, the indicator-to-returns correlations previously learnt during training are significantly stronger than the amount of randomness. Therefore, we can expect similar, but not same *relative* results for in-sample.

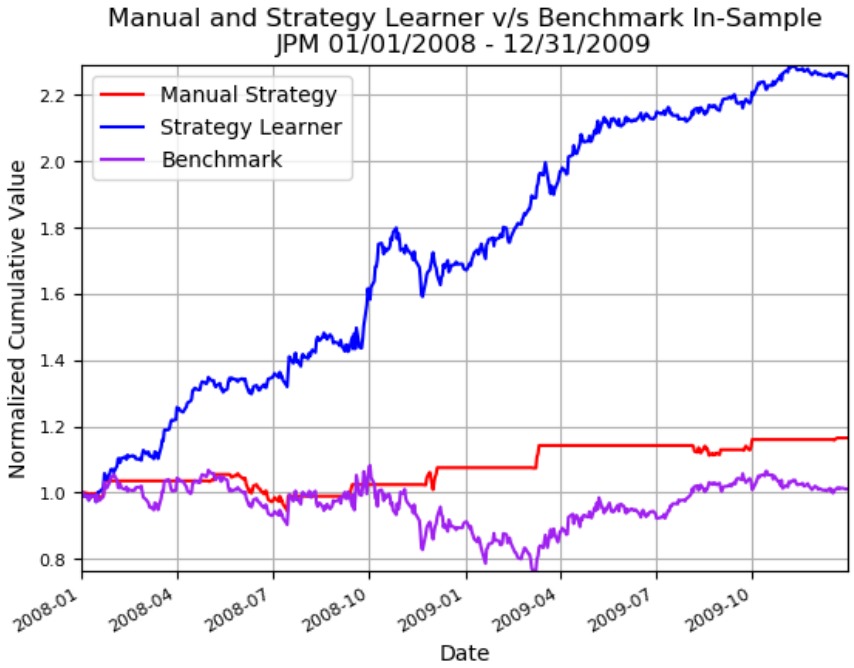

Figure 4. Experiment 1. In-Sample Comparison

	CUMULATIVE RETURNS	STANDARD DEVIATION	MEAN OF DAILY RETURNS	SHARPE RATIO
MANUAL	0.165408	0.005970	0.000321	0.854766
STRATEGY	1.255923	0.010090	0.001666	2.621226
BENCHMARK	0.012325	0.017041	0.000169	0.157205

Table 4. Manual Trader v/s Strategy Learner v/s Benchmark In-Sample Results

Out-Of-Sample Testing

- Strategy Learner beat benchmark and provided positive returns but underperformed relative to both manual strategy and compared to in-sample, even though best practices to minimize overfitting were followed (leaf size, bags, no future peeking, etc.). This is likely due to hyperparameter settings being optimized in-sample collectively for a

group of 5 symbols, suggesting that machine learning models for trading should be tuned for each symbol separately.

- Manual strategy outperformed benchmark again, although by a lesser extent, once again confirming the quality of technical indicators.

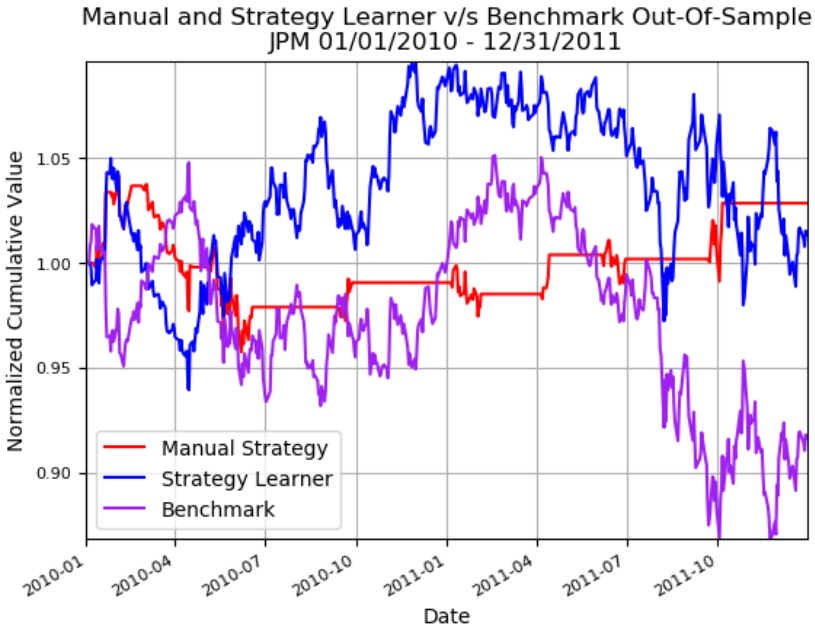

Figure 5. Experiment 1. Out-Of-Sample-Sample Comparison

	CUMULATIVE RETURNS	STANDARD DEVIATION	MEAN OF DAILY RETURNS	SHARPE RATIO
MANUAL	0.028597	0.004091	0.000064	0.249838
STRATEGY	0.013788	0.007779	0.000057	0.117186
BENCHMARK	-0.083579	0.008500	-0.000137	-0.256657

Table 5. Manual Trader v/s Strategy Learner v/s Benchmark Out-Of-Sample Results

Experiment 2 (Strategy Learner)

The purpose of this experiment was to assess the effect of market impact on the Strategy Learner, particularly on in-sample trading. The experiment was coordinated by a stand-alone Python program that trained the Strategy Learner with JPM data between 1/1/2008 and 12/31/2009, in three distinct trials with varying impact (0, 0.005, 0.010), $0 commission, and tested against in-sample data. Two metrics of interest (Cumulative Return, Number of Trades) were studied.

Hypothesis: Since Strategy Learner accounts for impact as part of its learning, it is expected that as impact increases, N-day returns decreases resulting in a) fewer trades satisfying threshold settings, and b) reduced profitability of trades and cumulative returns. The summary of metrics and charts (below), confirms the hypothesis of higher impact leading to lower returns and number of trades.

	IMPACT = 0	IMPACT = 0.005	IMPACT = 0.010
CUMULATIVE RETURNS	1.575700 (157.57%)	1.026599 (102.66%)	0.851171 (85.12%)
NUMBER OF TRADES	94	53	43

Table 6. Experiment 2. Effect of Market Impact

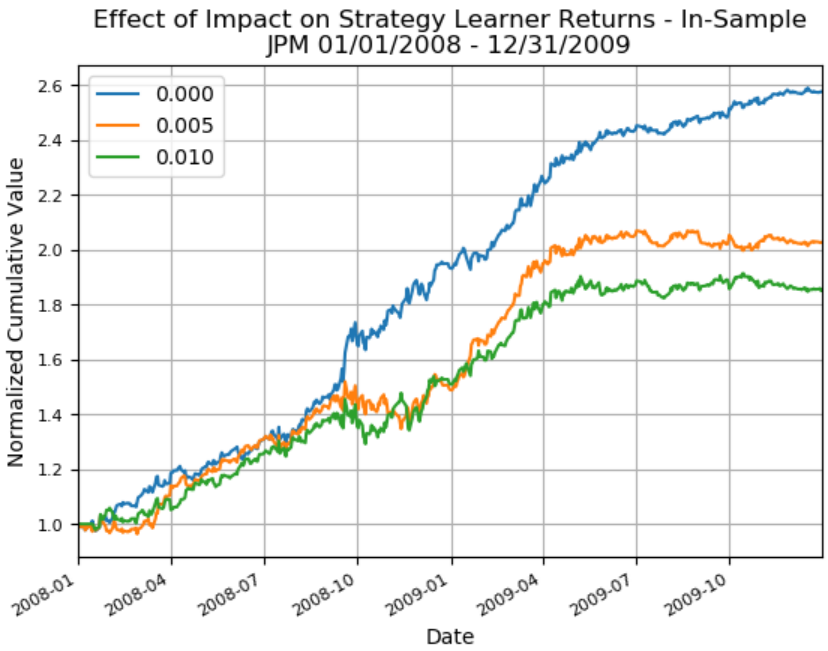

Figure 6. Experiment 2. Effect of Market Impact on Cumulative Return

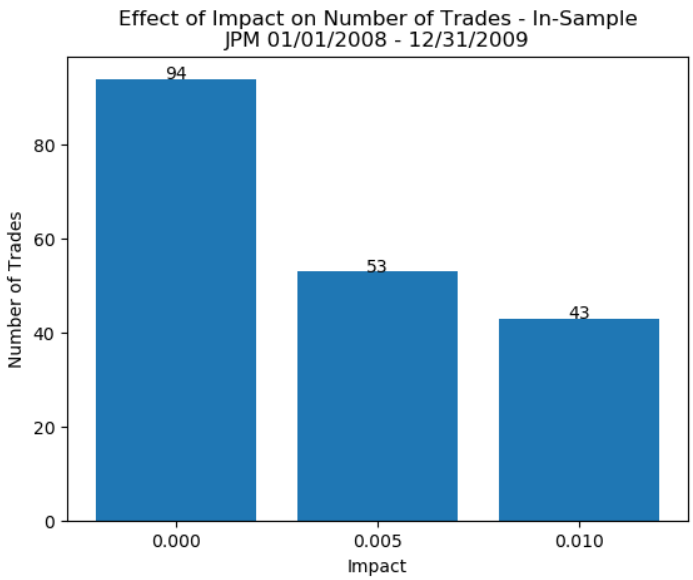

Figure 7. Experiment 2. Effect of Market Impact on Number of Trades

REFERENCES

1. Technical Indicators, 2023 Stock Charts. Available at: https://school.stockcharts.com/doku.php?id=technical_indicators (Accessed 14-Apr-2023)
2. 'Technical indicator' 2023 Wikipedia. Available at: https://en.wikipedia.org/wiki/Technical_indicator. (Accessed: 14-Apr-2023)