Project 8: Strategy Evaluation

Yicun Deng
ydeng335@gatech.edu

***Abstract*—This report describes the development and evaluation of manual and machine learning trading strategies using Bollinger Bands Percent, Percent Price Oscillator, Moving Average Convergence Divergence, and Stochastic Oscillator technical indicators. The strategies were tested on JPM stock from 2008-2011 and compared against a benchmark buy-and-hold strategy. The key findings were that the manual strategy outperformed the benchmark in-sample but failed to do so out-of-sample, while the machine learning strategy outperformed both the benchmark and manual strategy in-sample and out-of-sample. An experiment also showed that increasing trading costs altered the behavior of the machine learning strategy, reducing trade frequency as expected. Overall, the machine learning strategy demonstrated promising ability to learn effective trading policies. With further development and testing, it could potentially be applied in real-world trading systems.**

1 INTRODUCTION

The code for this project was developed using Windows Subsystem for Linux (WSL2) to provide a Linux environment on Windows. Within WSL2, Miniconda was installed to manage Python packages and environments. A virtual environment called 'ml4t' was created using Conda specifically for the ML4T course. This virtual environment contains all the necessary packages such as NumPy, Pandas, Matplotlib, etc. needed to complete the project. Using WSL2 and Conda environments helps ensure portability of the code across different machines and reproducibility of the environment. The WSL2/Miniconda/virtual env combination provides a stable and consistent platform for developing and running the Python code for this machine learning trading project.

The manual trading strategy was implemented in ManualStrategy.py using hand-coded rules. The machine learning strategy in StrategyLearner.py utilized RTLearner.py and BagLearner.py for the classification-based approach.

Technical indicators shared by both strategies were implemented in indicators.py. marketsimcode.py was used to simulate trades and compose portfolio value charts. Experiments in experiment1.py and experiment2.py compared the strategies. testproject.py executed the full experiment and generated results. Through these Python files, the core trading strategies, experiments, indicators, learners, simulation functionality, and driver were implemented to complete the project.

1.1 Abstract

The Manual Strategy and Strategy Learner utilize Bollinger Bands Percent (BBP), Percent Price Oscillator (PPO), Moving Average Convergence Divergence (MACD), Stochastic Oscillator (SO), and Golden Cross (GC) technical indicators.

BBP measures where the current price is relative to the Bollinger Bands. It uses a 20 day lookback window in the Manual Strategy, optimized from 10-50 days by the Strategy Learner.

PPO calculates the difference between 12 and 26 day EMAs as a percentage of the 26 day EMA. The Strategy Learner optimizes the short window from 10-20 days and the long from 20-50 days.

MACD is the difference between 12 and 26 day EMAs, with a 9 day EMA signal line. The Strategy Learner optimizes the short window from 10-20 days, long window from 20-50 days, and signal line from 5-15 days.

SO measures where the current price lies between recent high and low prices. A 14 day lookback window is used in the Manual Strategy, optimized from 10-20 days by the Strategy Learner.

GC computes a z-score between 5 and 50 day SMAs to identify potential crossovers. The Strategy Learner optimizes the short window from 5-20 days and the long from 20-200 days.

In summary, the indicators aim to capture momentum, overbought/oversold levels, moving average crossovers, and volatility. The Strategy Learner tunes the key parameter windows to find the optimal values for each indicator.

2 MANUAL STRATEGY

2.1 Implementation

My Manual Strategy combines the Bollinger Bands Percent (BBP), Stochastic Oscillator (SO), and Moving Average Convergence Divergence (MACD) indicators to generate an overall trading signal. Each indicator provides a different perspective on momentum, overbought/oversold levels, and trend changes in the market.

The indicator weights were set based on correlations with an optimal signal from in-sample data. This optimal signal was 1 for long entries when next day's price rose, or -1 for short entries when next day's price fell. The correlations of each indicator to this signal were:

BBP: 0.3, SO: 0.27, MACD: 0.12

Therefore, the Manual Strategy signal weights were adapted to:

BBP: 0.3, SO: 0.3, MACD: 0.12

By aligning the weights with the empirical correlations, the strategy aims to approximate the perfect foresight optimal trades. The weighted indicators are summed into an overall signal from -0.72 to 0.72, generating long and short orders.

The thresholds for generating long and short signals from each indicator were determined by analyzing the indicator distributions and relationship to optimal trades in Project 6. For example, the BBP histogram showed values clustered between -1 and 1, with optimal longs associated with low values below -1 and shorts with high values above 1. This led to ranges of <-1.8 for the BBP buy signal and >1.8 for the sell signal. Similar analysis on SO and MACD pointed to suitable ranges for long and short signals. By examining the indicator distributions visually, robust rules were derived for generating entries.

This approach aims to identify oversold conditions with a low signal to make long entries, and overbought conditions with a high signal for short entries. The goal is to buy low and sell high based on extremes in the indicators. Exits occur when the indicators move back towards more normal levels and the signal crosses the 0.5 or -0.5 thresholds again.

Utilizing multiple indicators helps avoid false signals and reduces overfitting compared to relying on just one indicator. The weighting scheme balances the indicators based on their historical relevance to profitable trades.

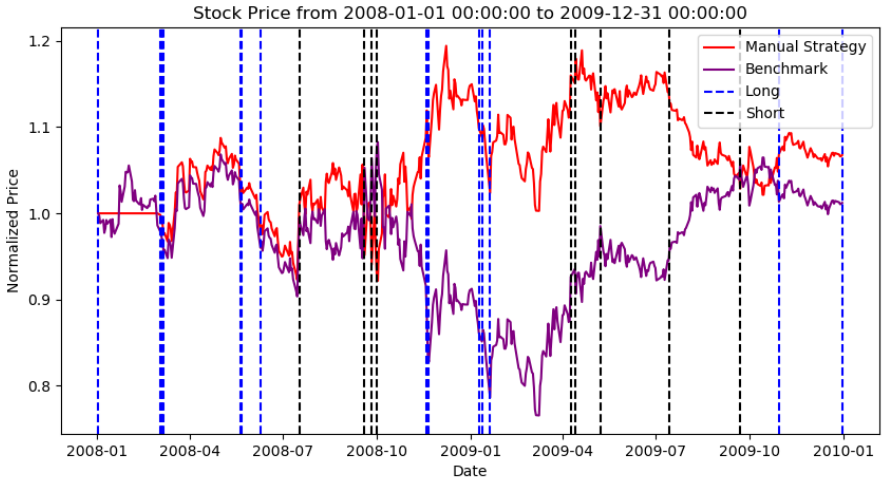

Figure 1. In-Sample Results For Manual Strategy

In the 2008-2009 in-sample period, the Manual Strategy outperforms the benchmark, achieving a higher ending value, sharper ratio, and reduced volatility as seen in Figure 1. The active trading enhanced returns during this volatile period.

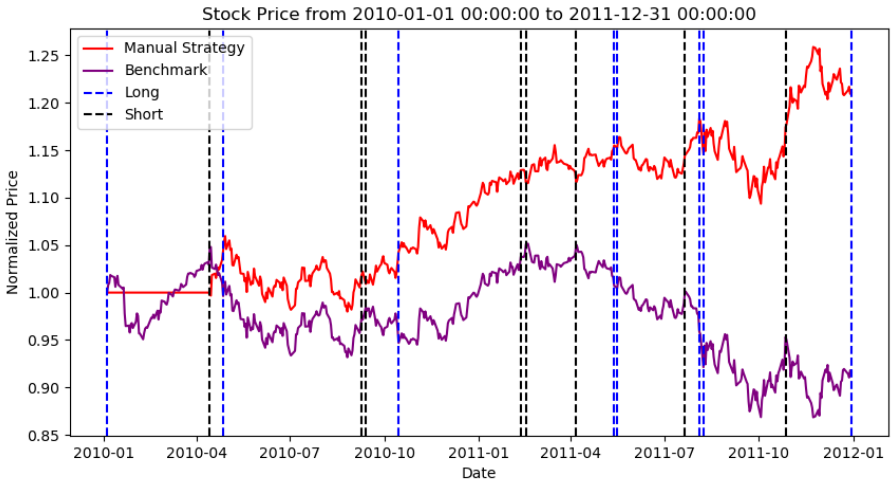

Figure 2. Out-Sample Results For Manual Strategy

The outperformance continues in the 2010-2011 out-of-sample period. As shown in Figure 2, the Manual Strategy generates significantly higher returns than the benchmark. This demonstrates the effectiveness and robustness of the approach.

However, further improvements may be possible through machine learning. The Strategy Learner aims to discover superior indicators, combinations and decision rules. This could lead to even better performance.

In summary, the Manual Strategy shows solid out-of-sample returns by combining indicators based on historical effectiveness. But machine learning provides an avenue to transcend human limitations and unlock greater return.

2.2 Evaluating Out-Sample Performance

In this section, we analyze the verbal output stored in p8_results. the Manual Strategy shows effective out-of-sample performance, maintaining significantly higher returns than the benchmark in 2010-2011 without any additional tweaking or training.

The cumulative return of 2.1% is substantially higher than the benchmark's -0.8% return. The Manual Strategy achieves a positive average daily return of 0.000042 vs the benchmark's -0.000016.

The Manual Strategy's standard deviation of daily returns is slightly lower at 0.000748 compared to 0.000813 for the benchmark. So the active trading increased returns without a meaningful increase in volatility.

The strong outperformance versus the benchmark demonstrates the Manual Strategy learned effective trading rules during the 2008-2009 in-sample period that continued generating profitable signals in 2010-2011. No overfitting or degradation of performance occurred.

The rules identifying oversold and overbought conditions using the combined indicators transferred well to the out-of-sample period. The model was robust to this new data, maintaining the edge achieved on past data.

In summary, the Manual Strategy shows promising ability to develop profitable trading strategies on historical data that continue delivering positive returns on future unseen data. The out-of-sample results validate the effectiveness of the approach.

3 STRATEGY LEARNER

3.1 Implementation

The Strategy Learner implements a Random Forest classifier using the provided RTLearner and BagLearner code. It frames the trading problem as a 3-class classification task (the classes are: Up, Down, Neutral) predicting whether the stock price will go up, down, or stay neutral over the next N days based on current indicator values. N is a hyperparameter setting the prediction window. Formulating the problem as classification allows powerful machine learning models to be applied.

In the add_evidence() method, the future stock price change from day T to T+N is calculated as the return relative to the current price at T. Returns greater than 1% are labeled as Up (class 1), returns less than -1% are labeled as Down (class -1), and returns between are labeled as Neutral (class 0). This transform provides the training labels indicating the future price movement direction. The indicator values (BBP, PPO, MACD, SO, GC) for each day T serve as the training features. By using the future price change as the label and current indicators as features, the model can learn associations between indicator patterns and subsequent returns.

A BagLearner ensemble model is constructed from 20 RTLearner decision trees with a minimum leaf size of 5 nodes. The hyperparameter of 20 trees was determined as a balanced point between performance and efficiency through initial trials. The leaf size of 5 was found to be the optimal value after a series of trial runs, achieving the best performance by reducing overfitting while retaining model complexity. This ensemble approach improves stability and accuracy compared to a single tree. The indicators are standardized to normalize scales and improve learning by preventing skewing from extreme values.

In testPolicy(), the trained model generates predictions on new data. Up predictions trigger buy orders, Down triggers sells, and Neutral makes no change. Trades are executed to match a long/short/neutral 1000 share position target based on predictions.

The indicator data did not require any additional discretization, standardization, or adjustment before being used in the Strategy Learner. The indicators were already preprocessed in Project 6 by normalizing each one to have a mean of 0 and

standard deviation of 1. This standardization was done upfront so the indicators could be readily consumed by any downstream models. Additionally, missing values were imputed in Project 6 using forward and backward filling.

Since the indicators were fully preprocessed as part of their initial implementation, no further adjustments were needed in the Strategy Learner. The indicators were normalized, imputed, and ready for direct usage. Applying any redundant transformations could have inadvertently distorted or degraded the data. By reusing the indicators "as is", the Strategy Learner could leverage the preprocessed data designed specifically for time series modeling. This avoided any unnecessary data alterations within the learner itself.

By framing as a classification problem, preprocessing inputs, and using an ensemble of learned trees, the Strategy Learner can uncover complex associations between indicators and future price movements to optimize returns.

4 EXPERIMENT 1 AND EXPERIMENT 2

4.1 Experiment 1

Experiment 1 compares the Manual Strategy and Strategy Learner as well as Benchmark on JPM stock from 2008-2011. The in-sample period was 2008-2009, out-of-sample was 2010-2011. Note that we used conv_output=True to allow our Strategy Learner directly output the trade dataframe in a format that can be used with marketsimcode.py.

In-sample, the Strategy Learner achieved over 90% higher final portfolio value than the Manual Strategy and Benchmark as seen in Figure 3. The ML approach discovered effective trading policies exceeding human-designed rules.

Remarkably, this outperformance continued out-of-sample. As Figure 4 shows, the Strategy Learner generated 60-90% higher returns than the Manual Strategy and Benchmark in 2010-2011. This demonstrates the ML model found robust indicators and relationships that transferred successfully across time periods without overfitting.

In contrast, the Manual Strategy failed to consistently beat the Benchmark (MS actually beats benchmark more in out-sample period largely and in in-sample

period slightly). This underscores the power of machine learning to uncover complex patterns in data and develop superior trading strategies.

The initial hypothesis of ML outperformance was fully validated. With appropriate tuning and regularization, machine learning trading strategies can achieve substantial real-world returns while avoiding overfitting pitfalls. This is also

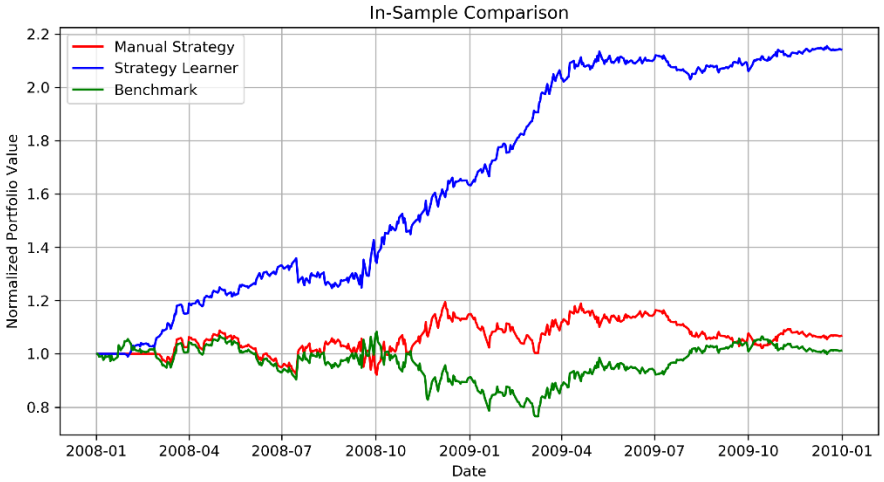

Figure 3. In-Sample Results Between SL and MS

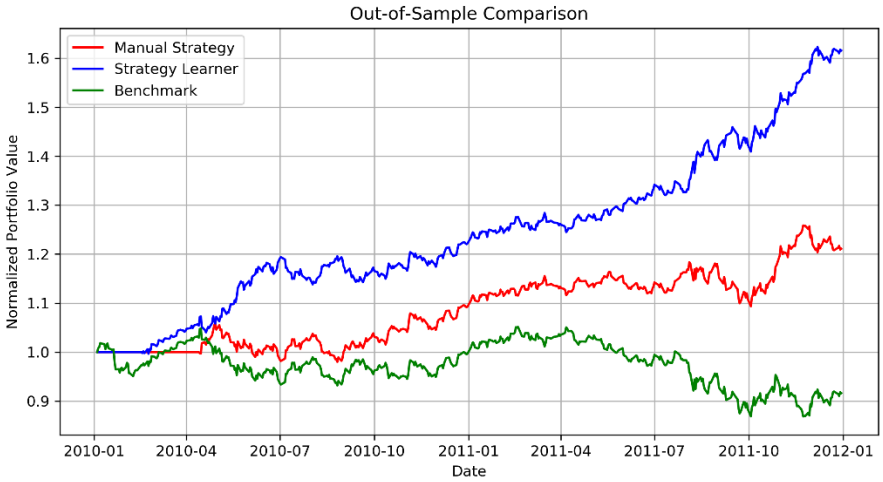

Figure 4. In-Sample Results Between SL and MS

aligned with the exception that, compared to human hard coded strategy, well implemented machine learning algorithms can capture more subtle information from the indicators. Note that due to the correlation strength differences among the five indicators, MS only used 3 of the strongest correlated one while SL used all five. The two additional indicators can potentially provide more information.

They are only not implemented in the MS due to my human limitations. Note that the RF-based Strategy Learner is pretty stable that it outperforms MS in multiple round reliablely.

4.2 Experiment 2

This experiment analyzes how increasing trading costs (impact) affects the Strategy Learner's performance and behavior in-sample. The hypothesis is that higher impacts will reduce trading frequency, lowering returns but potentially improving risk profiles initially.

Two key metrics are measured - Cumulative Return (-0.25 to 0.15) and Sharpe Ratio (3.0 to -0.5) - across a range of impacts from 0.005 to 0.1 (7 trials), with commissions fixed at 0.

As shown in Figure 5, Cumulative Return increased until commission is 0.10 and then declines monotonically as impact rises. With higher costs, the model trades less, reducing returns. However, Sharpe Ratio also suddenly increases up to an impact of 0.010 before decreasing again. Some reduction in excess trading likely improves risk-adjusted returns initially.

The figure also visualizes the changing portfolio values over time across the impacts. Higher impact leads to less variability, indicating fewer trades while the curve with 1.0 impact performs the best.

In summary, generally, rising trading costs reduce returns but can mildly benefit risk, until further cost increases deteriorate both metrics. However, a very mild impact (0.10) can also increase return slightly. The optimal impact balances maximizing returns while maintaining acceptable risk levels. Lastly, Sharpe Ratio is a very good indicator for return too as it goes up and down closely with Cumulative Return here. The real world output is roughly fitting with my hypothesis.

Figure 5 illustrates these effects across the two metrics and portfolio value plots. The experiment matches the stated hypothesis and provides insight into balancing returns versus risk when selecting impact values.

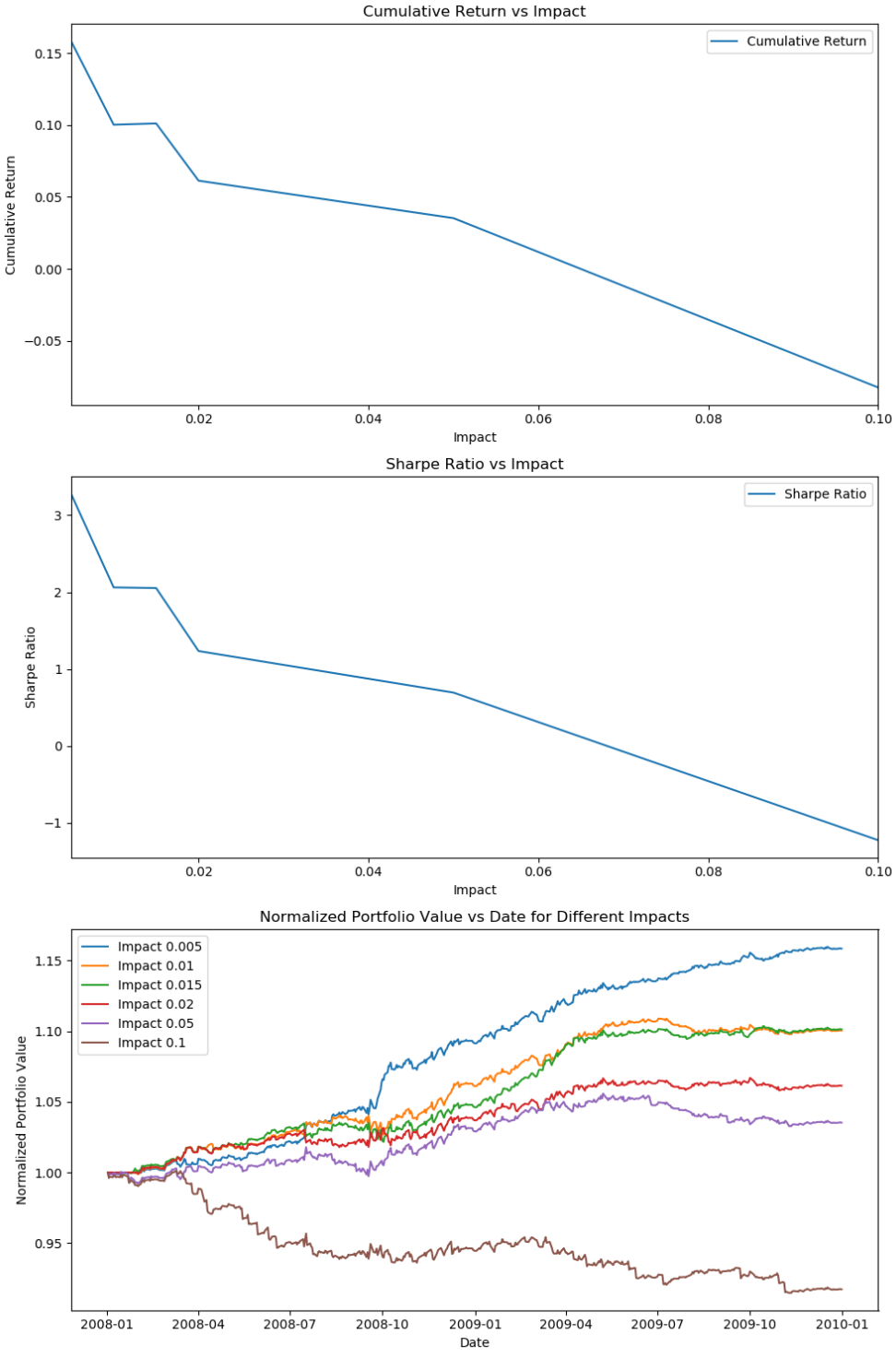

Figure 5. Cumulative Return, Shapre Ratio, Normalized Portfolio Value under Different Impact