Project 8: Strategy Evaluation

Christopher Hynes
chynes3@gatech.edu

***Abstract*—This paper develops a manual and random forest trading strategy using three financial indicators: %B, RSI, and %ATR. Their performance is measured against a benchmark strategy to determine whether they beat the market. Finally, the random forest learner is trained on three different levels of impact to illustrate how it adapts to transaction costs.**

1. INTRODUCTION

Technical analysis is an investing approach in which investors use historical price and volume information to predict future returns. Investors construct financial indicators from this information that can identify potential buy/sell opportunities, then combine these indicators in some way as a trading strategy.

The first step in building a trading strategy is to identify indicators that may be insightful. In this paper, I develop three: %B, RSI, and %ATR. We can combine and tune these indicators in several ways. I outline three: a manual strategy, a random forest strategy learner, and a buy and hold strategy as a benchmark.

The key question of this paper is how the performance of these strategies compares. I define the "performance" of these strategies as their risk-adjusted returns for the stock "JPM" over a training period. I use a later testing period to evaluate their performance on unseen data but perform all optimization on the *training set* as to not "peek" into the future.

I conduct three analyses to evaluate the three trading strategies I create. First, I compare the manual strategy to the benchmark in the in-sample and out-of-sample period to illustrate their behavior. Next, I compare the in-sample and out-of-sample performance of all three strategies. Finally, I evaluate the effect of transaction costs on the behavior of the machine learning strategy.

2. DATA

These experiments use historical asset data from Yahoo Finance. The in-sample data spans 505 trading days from January 1, 2008 to December 31, 2009, and the

out-of-sample spans 504 trading days from January 1, 2010 to December 31, 2011. I focus on the adjusted close price, high, and low of one stock, 'JPM'.

3. METHODS

After testing different subsets of indicators in the manual strategy, I chose the three that consistently had the best performance. Each indicator has parameters that can be tuned. I used industry standards to keep the focus on tuning the *strategy*, not the indicators (and because they consistently performed well).

3.1. Indicator Selection

3.1.1. *%B*

The %B indicator is a ratio of a security's price to its upper and lower Bollinger Bands, which I calculate as bands 2 rolling standard deviations above and below a simple moving average (SMA), both using a 20 day window (%B [ChartSchool], n.d.). Thus, I begin with their values:

BBs = Rolling Mean $\pm$ 2 $\times$ Rolling St. Dev.

Then %B = (Price - Lower Band)/(Upper Band - Lower Band)

We can use the %B indicator to find entry and exit points with the idea that large excursions in price from the MA indicate overbought/oversold conditions and will eventually return to it. It may signal a buy after %B crosses an upper/lower threshold (e.g., 0.15 and 0.85), anticipating a return to the MA.

3.1.2. *Relative Strength Index (RSI)*

The RSI is a "momentum oscillator [that] measures the speed and change of price movements" (RSI [ChartSchool], n.d.). First, I compute a SMA of gain ($) for days there were price increases, and SMA of loss for days there were decreases (both using a 14 day window). Then:

$RSI = 100/(1 + \text{Average Gain}/\text{Average Loss})$

RSI ranges from 0 to 100. A high value represents a strong uptrend in price, which can indicate overbought conditions, and vice versa. For extreme values, we might expect a return to the trend, so we could sell when RSI is above some threshold and sell when it is below another.

3.1.3. *%ATR*

I define a custom indicator, "%ATR", that considers volatility measured by the average true range (ATR). I wanted a volatility-based signal that might provide complementary information to %B. I start with ATR (ATR [ChartSchool], n.d.):

$$TR = \max\left(\text{High}_t - \text{Low}_t, |\text{High}_t - \text{Close}_{t-1}|, |\text{Low}_t - \text{Close}_{t-1}|\right) \text{ for days } t$$

The ATR is a SMA (I use 14 days) of TR. Next, I compute the upper and lower bands, which are centered around a SMA (I use 14 days) of price:

ATR Bands $= SMA \pm ATR$

Then $\%ATR =$ (Price - Lower Band)/(Upper Band - Lower Band)

Again, we may signal a buy/sell when %ATR is above/below some defined thresholds with the idea that a price deviation more than the ATR from the trend will eventually return to normal.

4. MODEL

4.1. Strategy Design

I chose to optimize on the Sharpe Ratio over returns, as I prefer high risk-adjusted returns to a potential stroke of luck that inflates returns. Both strategies make decisions by comparing a subset of each day's indicators to defined thresholds. However, the approach to finding the subset and thresholds differs.

4.1.1. *Manual Strategy*

In the manual strategy, I acted as the learner and tested dozens of combinations of indicator thresholds, recording the SR of each in a spreadsheet. I also tested different combinations of indicators: first, by changing the indicators themselves, then by combining them in different ways (votes, etc.). I both tested random numbers and looked into the data to see where I might tweak an indicator to be more profitable.

I decided on a system where all three indicators must agree on a decision. The strategy sells when %B, RSI, and %ATR are above 0.85, 61, and 1, respectively. It buys when %B, RSI, and %ATR are below 0.15, 23, and 0, respectively.

I believe that such a strategy may *generally* be effective because, as mentioned, high values of these indicators suggest overbought conditions and vice versa. At such extreme price levels, we might expect a reversion to the trend. Then, we might sell a stock when we believe it is overpriced and buy when underpriced. Combining indicators generates more informed trading decisions.

Important is the nature of these three indicators: each has unique information. %B is a longer-term measure of volatility, with a lookback period of 20 days compared to %ATR's 14 days. %ATR includes high and low prices in addition to adjusted close, providing a more comprehensive view of volatility. Finally, RSI measures the market's *trend*, which can indicate bull and bear markets.

While we may have some luck with technical analysis, we cannot ignore the effects of fundamentals, macroeconomic conditions, and events. For example, consider JPM: in a recession (or, as an extreme, bankruptcy), the price may fall and remain below oversold. On the other hand, an innovation like the best AI trading strategy in the market would do the opposite.

Such external events can be dangerous if a trading strategy is caught long in a bear market or short in a bull market. If I used standard RSI values, the manual strategy held positions where the market moved against it.

4.1.2. *Strategy Learner*

I chose to use random forests, so I had to translate the goal, predicting future returns, into something interpretable by a computer. The goal is to buy stocks whose price will increase and short those that will decrease. Then, I label:

$$Y_t = \begin{cases} 1 & \text{if Price}_{(t+3)} \geq 0.015 + \text{impact} \\ -1 & \text{if Price}_{(t+3)} \leq -0.025 - \text{impact} \\ 0 & \text{otherwise} \end{cases}$$

In other words, Y is 1 for days where a stock's price three days ahead is at least 1.5% higher, -1 when it is at least 2.5% lower, and 0 otherwise after considering impact: the percentage that price moves *against* any transaction. We only want to make a trade that is profitable *after* transaction costs. The parameters involved in the calculation of Y were the most difficult to optimize.

A random tree randomly chooses an indicator X, splits the data into an upper and lower branch where $X_i > \text{median}_X$ and $X_i \leq \text{median}_X$, respectively, and recursively repeats these steps until each branch has some specified number of

observations. This branch is then a leaf, and the leaf's most frequent Y value is the prediction for that set of feature values. A random forest queries many random trees and returns the mode of the predictions.

There is no need to standardize or otherwise adjust the data because random trees use the median, which works with continuous numbers and depends only on the order of values. Standardizing would not change this order.

I started with a low leaf size of 5 and high bag size of 200 because I wanted to first focus *purely* on in-sample performance and *then* make it more generalizable. Next, I tried hundreds of combinations of Y (and later, learner) parameters. I ran 30 trials of each combination for more consistent results and recorded the corresponding average SR and return in a spreadsheet.

I started with a lookahead of 1 day and incremented the lower and upper thresholds from 0.01 to 0.05, and then by increments of 0.05 up to 0.25: first individually, and then together. I did this for lookahead values up to 8 days, as performance consistently diminished with a longer lookahead. I noticed that lower thresholds performed better, so I tested halfway between the lower values (up to 0.03) I had already tested.

Lower lookahead periods and smaller Y thresholds tended to maximize in-sample JPM returns. However, they performed badly with long-term trends and other symbols, specifically AAPL, ML4T-220, SINE_FAST_NOISE, and UNH, so I tested different combinations of leaf sizes and lookahead periods. After finding that a leaf size of 8 and lookahead of 3 days consistently worked well for these symbols, I again tuned threshold values, settling on an upper of (0.015 + impact) and lower of (-0.025 - impact).

Finally I tested this strategy and changed bag size by 20 each time. I chose 100 bags because increasing it beyond this number improved performance by an amount that I thought did not justify the increase in complexity and run time.

I set the seed to my GT ID (903969044) for all experiments for replicability.

4.1.3. *A Note on Both Strategies*

Both strategies are restricted to a position of 1,000 or -1,000 shares. I programmed my strategies to avoid staying in cash: they always hold 1,000 shares short/long until the opposite position is signaled, never 0. Also, recall that both

these strategies use adjusted closing prices for all decisions. Since this price is only available at the *end* of a trading day, any orders that the strategies signal will be traded on the *next trading day*.

4.2. Experimental Design

4.2.1. *Experiment 1*

The foremost concern about any trading strategy is its performance. In experiment 1, I compare the returns of the benchmark, manual, and learner strategies on the in-sample and out-of-sample data. We might expect the learner to perform the best because it *learns* to predict returns, followed by the manual strategy in second and benchmark in third place.

4.2.2. *Experiment 2*

As mentioned, I include impact in the Y label calculation to account for transaction costs. This characteristic is important as it allows the learner to alter its behavior for different values of impact. I calculate two metrics, cumulative returns and the number of trades, across three values of impact: 0.00, 0.015, and 0.03, to illustrate these changes in the in-sample period.

Since impact represents a transaction cost, we might expect that higher impacts will diminish cumulative returns. Moreover, since the learner only signals a trade if it thinks that the 3-day return will be greater than some percentage *and* transaction cost, it will likely make fewer total trades.

5. DISCUSSION

5.1. Manual Strategy Performance

Figure 1 shows the performance of the manual and benchmark strategies on the in-sample data. Table 1 provides selected statistics of each strategy. The manual strategy has a much higher cumulative return: 53.68% compared to the benchmark's 1.23%. It also has a lower standard deviation of daily returns: 1.16% vs. 1.70%, resulting in a SR of 1.258 to the benchmark's 0.157. Thus, the manual strategy has much higher returns with less risk.

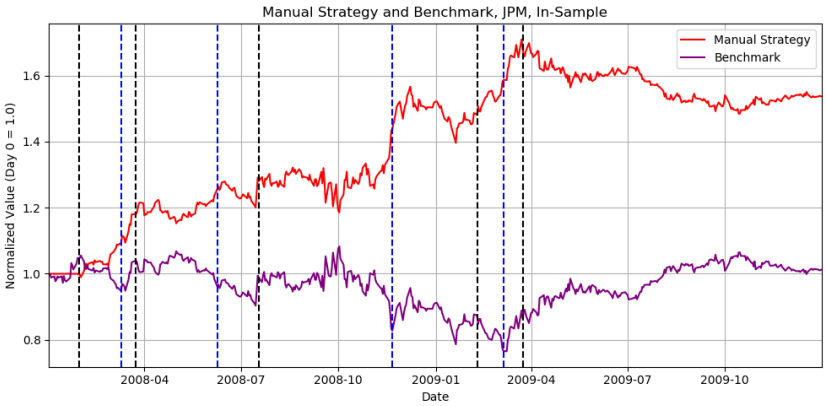

Figure 1—Normalized cumulative portfolio returns of the manual and benchmark strategy, in-sample data. Blue lines indicate long entry points, black lines indicate short entry points.

	Cumulative Return	**Mean Daily Return**	**Std. Dev. Daily Return**	**Sharpe Ratio**
Manual Strategy In-Sample	0.536769	0.000920	0.011610	1.257919
Benchmark In-Sample	0.012325	0.000169	0.017041	0.157205
Manual Strategy Out-of-Sample	0.105342	0.000229	0.007724	0.470445
Benchmark Out-of Sample	-0.083579	-0.000137	0.008500	-0.256657

Table 1—Cumulative return, mean daily return, std. dev. of daily return, and Sharpe Ratio of the manual and benchmark strategies for the in-sample and out-of-sample data.

The manual strategy also outperforms the benchmark on the out-of-sample data, though less consistently. Figure 2 plots their portfolio values. The manual strategy has a much higher portfolio value across practically the entire in-sample period. In the out-of-sample period, however, this gap is smaller (and nonexistent for three periods after the start).

Still, the manual strategy has a final return of 10.53%, while the benchmark is down 8.36%. Again, it has a lower standard deviation and higher SR (0.470 vs -0.257). The manual strategy manages to profit despite the price of JPM *decreasing* over this time period, supporting the ability of technical analysis.

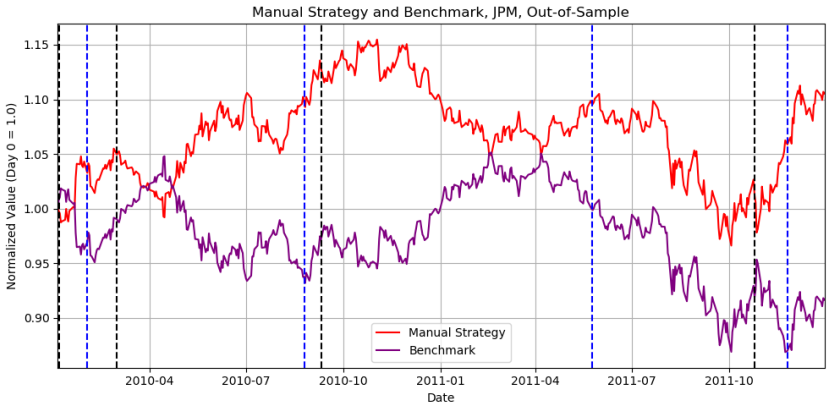

Figure 2—Normalized cumulative portfolio returns of the manual and benchmark strategy, out-of-sample data. Blue lines indicate long entry points, black lines indicate short entry points.

These differences occur because I *chose* those thresholds that maximized in-sample returns: *I* designed it this way. Models generally do well with the data they were trained on and worse on unseen data, even if it is a profitable strategy with certain features that outperform a buy-and-hold approach.

5.2. Experiment Results

5.2.1. *Experiment 1*

Figures 3 and 4 compare the learner's in-sample and out-of-sample performance, respectively, to the manual and benchmark strategies. Their returns are 132.09%, 53.68%, and 1.23%. The learner performs best, likely because it *learns* patterns from the data. We cannot expect this relative result *every* time, as the random forest will be different every run. I set a seed, so they are here.

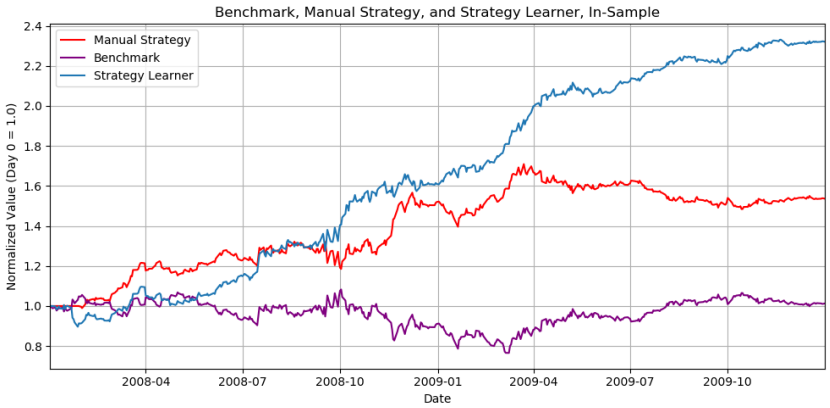

Figure 3—Normalized cumulative portfolio returns of the manual, learner, and benchmark strategy, in-sample data.

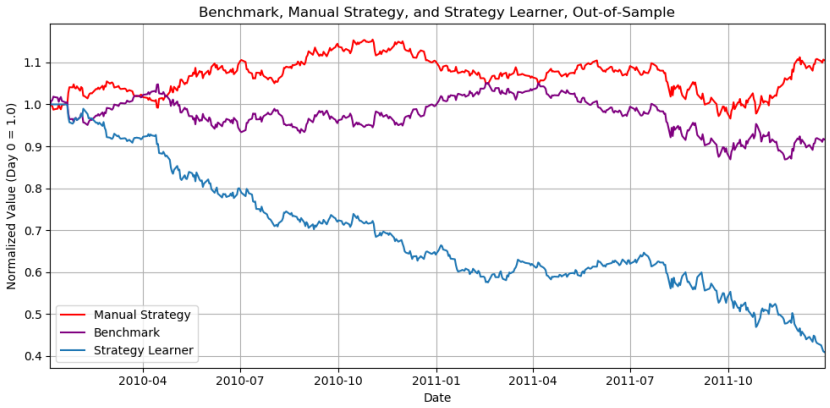

Figure 4—Normalized cumulative portfolio returns of the manual, learner, and benchmark strategy, out-of-sample data.

The learner's returns are much lower for the out-of-sample data, a disappointing result. Its return is -59.11% compared to the benchmark's -8.36% and manual strategy's 10.53%. One possible consequence of simply maximizing in-sample performance is overfitting. Here, the learner likely picks up exact patterns from the training data and so struggles to generalize to new data.

5.2.1. *Experiment 2*

Figures 5 and 6 show the cumulative returns and number of transactions for all three strategies on the in-sample period. The results support both hypotheses: impact has a negative relationship with both returns and the number of trades. The first is obvious in figure 5: returns are 185.53%, 64.97%, and -9.12% when impact is 0%, 1.5%, and 3%, respectively.

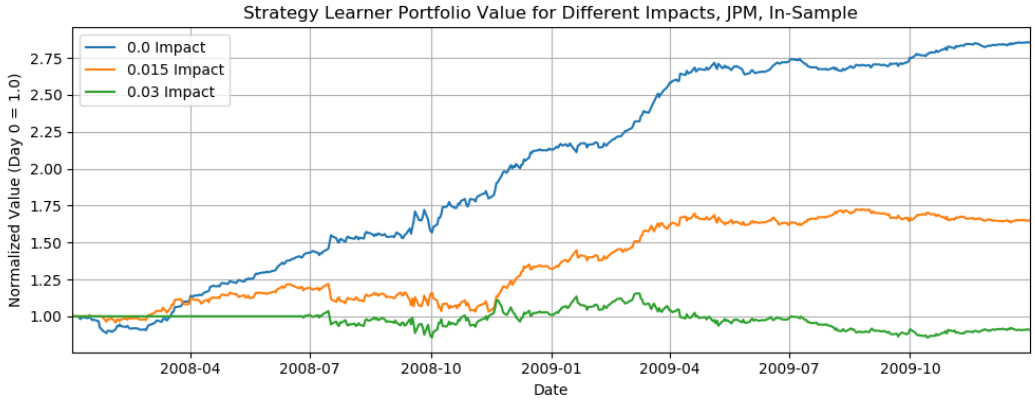

Figure 5—Normalized cumulative portfolio returns of the learner strategy for impacts of 0.0, 0.015, and 0.03, in-sample data.

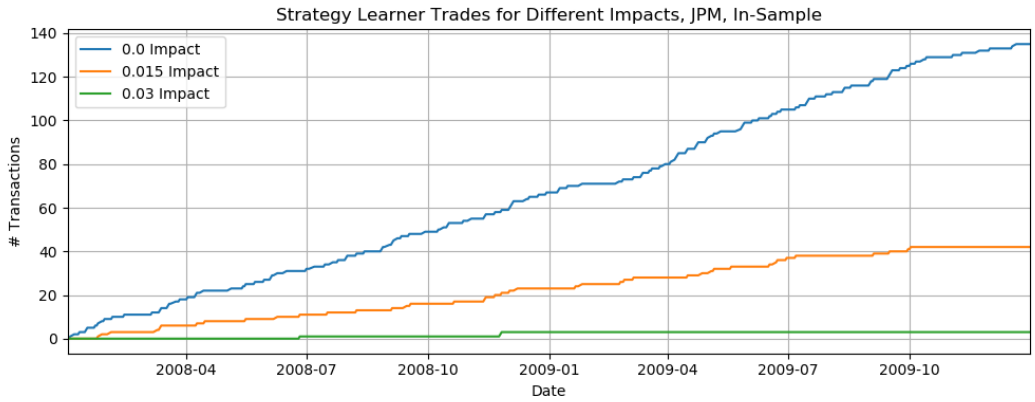

Figure 6—Number of trades of the learner strategy for impacts of 0.0, 0.015, and 0.03, in-sample data.

In the end, the number of trades was 135, 42, and 3 for impacts of 0%, 1.5%, and 3%, respectively**.** Recall that the learner only trades when it predicts that the 3-day return is sufficiently higher than the transaction costs. With higher transaction costs, these thresholds are more difficult to trigger (and so, trigger less). For example, with 3% impact, the learner does not predict any sufficient returns until around July, 2008, when it makes its first trade.

6. CONCLUSION

Investors can use financial indicators to build trading strategies. In this paper, I outlined two, a manual and machine learning strategy, and compared them to buying and holding. Despite outperforming the other strategies in-sample, the learner struggled with new data: it performed worst on the out-of-sample data. Still, the manual strategy performed best, providing evidence that technical analysis can be profitable.

However, one should be cautious of this learner's approach. Tuning a trading strategy to maximize the in-sample returns of a single stock can have unintended consequences. It may perform well with the data on which it was trained but not be generalizable to other assets or time periods: symptoms of overfitting. The primary challenge in creating a trading strategy is tuning its parameters, not choosing the indicators or programming it.

Future papers may work to make the learner more generalizable, perhaps by tuning it on some *universe* of stocks or using cross-validation. Moreover, a different learner altogether, such as a Q-Learner, may perform better.

7. REFERENCES

1. ATR [ChartSchool]. (n.d.). Retrieved March 11, 2024, from https://school.stockcharts.com/doku.php?id=technical_indicators:average_true_range_atr
2. %B [ChartSchool]. (n.d.). Retrieved March 9, 2024, from https://school.stockcharts.com/doku.php?id=technical_indicators:bollinger_band_perce
3. RSI [ChartSchool]. (n.d.). Retrieved March 9, 2024, from https://school.stockcharts.com/doku.php?id=technical_indicators:relative_strength_index_rsi