CS7646 – Machine Learning for Trading

"Strategy Learner"

Creation and Interpretation of a Machine-Learning-based Trading Strategy

Patrick Klaus Baginski

GT User ID: pbaginski3

Brief/About this report:

This report summarizes and explains the approach used to build a machine learning based trading strategy applicable to any type of stock. Additionally, this report outlines a comparison between a traditional technical indicator-based strategy and the developed machine learning-based strategy. To do so, the reader will find detail on indicators used, parameters of experiments, outcomes and interpretations. Charts are provided to illustrate the conclusions.

Note: There was no submission of the Manual Strategy assignment other than indicators.py. Therefore, a complete new manual strategy was developed, with 2 out of 3 of the indicators used in the submission.

Content

1. Developing a machine learning-based trading strategy

Trading problems are ambiguous. Despite major improvements in performance based on recent research, time series analysis remains the holy grail of data science. There are several challenges including framing the problem of predicting prices in a manner machine learning models can digest. Feeding features (information on stocks for example) and prices to a model is insufficient, it would only yield a prediction of the price on the day of the provided information. Instead, it is necessary to provide the model with some help for the predictor variable. This help is to "shift" the time series, or, "shift" the data for the price ahead of features data. The reason for this is that a machine learning models are trying to predict or classify the Y-Variable; However, it doesn't help much to predict today's stock price. Instead, we need to predict the future stock price so that the machine learning model can "learn" based on the features what the future stock price will be. This is the crucial step to time series analysis and much philosophy and research has gone into the understanding of how to do this shift. For strategy learner, I have chosen this approach. I am using a data set of prices, cleaned for trading-days-only based on the SPY and filled (backward and forward) in case of missing price values. I then calculate the future return in "n"-days (the number of days to look into the future). I look 20 days into the future for this (more on this in chapter 3). Additionally, I am introducing technical indicators as the features. Those indicators are rate-of-change, relative-strength-index and the Bollinger value. Further information is provided in the next chapter; however, a 20-day period was used for calculation. Indicators are used as they provide a price-only based view of the stock performance in addition to signals for a trading strategy. I am not applying standardization or discretization to the data because of choosing a random forest regression learner (more info below). The reason is random forests are partitioning algorithms, which means the learner only utilizes the ranking of the features to create partition rules based on a threshold. This leads a final data set to train the machine learning model that uses the above three indicators to predict the return from 20 days in the future.

There are many types of algorithms that can be chosen to frame the trading problem: When should I buy, hold or sell the stock I am using my model on? In order to answer this question, different performance metrics come into play and I have used final portfolio value, cumulative return of the portfolio and number of trades made. When choosing the algorithm, I choose between three options to frame this trading problem:

- Classification: "Classify a BUY, HOLD, SELL based on three indicators on day X."
    - These types of algorithms promise good efficiency in terms of compute resources because of the classification that is being made in this problem (1 BUY, -1 SELL).
- Regression: "Predict the 20-day future return of stock X based on the three indicators on day Y"
    - While a bit slower in runtime, this type of algorithms produces a continuous output, making it easier to interpret as a "stock return" to the layman.
- Reinforcement learning: "Trade based on the three indicators that you see and get rewarded for a positively impacting trade. Pick the best strategy that was learnt"
    - The newest of the bunch promises great performance since it is based on a reward-based learning process, however its implementation is also the hardest one and that makes it hard to interpret/explain to the layman.

For the Strategy Learner in this report I chose to go with the regression route and using a random forest as the model. The reason for this being generally good performance of regression algorithms on time series analysis easier to understand model type. The present algorithm operates on 20 bags and utilizes a leaf size of 19 in order to avoid overfitting and provide a good predictive performance. More on finding these values in chapter 3. Lastly, in order to train and

test this model, I have used the provided time ranges for in-sample (2008-2009) and out-of-sample data (2010-2011), as well as the starting value of cash (100,000 USD), the possible amounts of shares to be traded (negative/positive: 0/1000/2000) as well as the impact value (0.000 except for experiment 2). The process of applying this chosen model on the problem works by first generating the training data set. Here I use a look-back period of 60 days in order to have the indicators ready on the first possible day of trading. I then add the calculated future-day return to the trading days. The next step is to feed this data to the random forest learner, which then produces expected future returns as a predicted output. These can be used to create the ultimate trading data, which essentially tells us if we should buy, hold or sell a stock on a particular day. This is done by introducing the thresholds of YBUY and YSELL, which are set to 0.003 and -0.003 respectively. This means whenever the expected future return minus the impact (0.000 except for experiment 2) is higher/lower than this threshold, a buy/sell signal is returned. The final trading data is essentially a collection of daily trading actions in the shape of BUY/HOLD/SELL with stock volumes as described above. In order to account for position reversals, I am tracking cash (speak: holdings) and whenever a trade is induced by the thresholds and holdings allow, I am reversing the position. Lastly, the output of this can be introduced into a market simulator that calculates portfolio value based on the trades.

2. Researching and selecting indicators

To select the right indicators, I conducted research looking at the top indicators utilized by traders nowadays. This yielded a search for a specific goal: How many indicators does one need to get the most additional information out of the stock price? Therefore, I looked for indicators that would provide three types of information which are fundamentally different from each other, not visible on the stock price itself and yet encompassing additional information on the stock price. Those three types are: 1. Small scale trends in the data (rate of change) 2. History comparison of stock price (relative strength index) 3. Large scale trends in the data (Bollinger Value). With these three information types, the stock price data set the model uses to learn provides a good level of extra information while maintaining a low level of feature usage. This is important to strike a balance of Bias and Variance in the data set. Below the indicators are described and how they can yield trading signals.

Rate-of-Change is a technical indicator that is also a component of other technical indicators. It is a measurement of comparing prices at different points in time, most commonly today's price with a price that lies in the past. This type of comparison can help generate BUY/SELL signals, by measuring if the rate of change is accelerating or decelerating. (Figure 1)

Bollinger value is an indicator based on the principle of Bollinger Bands. Bollinger bands are a measure that combine the moving average of a price with a threshold of traditionally two standard deviations in both directions of the moving average. They are therefore an indicator related to the volatility of a stock, suggesting ultimately when prices move outside of the "bands" (speak, the standard deviation thresholds) a significant event has been triggered. This can either mean that prices are unusually high or low, therefore suggesting to make a BUY or SELL decision once an event occurs. The Bollinger value itself represents the value of the upper and lower band on a particular day. A trading signal is generated when this value is above (SELL) or below (BUY) two standard deviations away from the moving average. (Figure 3)

Relative Strength Index is a technical indicator also known as a momentum-oscillator. This means that it measures the strength/weakness of a stock based on the closing prices of a current trading window. It essentially measures the magnitude of price movements and then provides a value that tells if a stock is overbought or underbought, which translates into a SELL and BUY signal respectively. (Figure 2)

3. Tuning the machine learning-based trading strategy (describe testing the parameters here)

Once the indicators and model had been strategically chosen based on the trade-off of performance, compute time, level of information available to learn and threshold to make trades, the approach needed to be tuned for optimum performance. This is also called parameter and hyper-parameter tuning. The ultimately chosen tuning parameters have been described in chapter 1, however finding them required a strategic approach. In this case, all parameter values (YBUY, YSELL, bags, N, indicator-time-windows, leaf size) have been empirically developed. This means I developed a loop that ran the Strategy Learner multiple times across different ranges, one parameter at a time. In order to choose the right value, I looked at the cumulative return the model returned for each value and chose the one that produced the highest value before improvements to the value flattened off. The ranges I tested are below:

- N: 1-day to 45-days yielding final value of 20
- YBUY/YSELL: 0.0001 to 1.0 yielding final value of 0.003
- Bags: 10 to 250 yielding final value of 20
- Indicator-Time-Windows (same for all three although tested separately): 1-day to 45-days yielding final value of 20
- Leaf size: 5 to 150 yielding final value of 19

At this setting, the Strategy Learner model achieved averagely best performance and passed all test cases. The fact that most ended up being 20 was a coincidence as per the approach but could be explained by the fact that future returns can be best predicted on a similar time frame for indicators.

4. Experiment 1: Comparison of two strategies

In Experiment 1, the above described machine learning model is run on the in-sample time frame and test symbol JPM. The results are then compared to the same data used for the manual strategy model (a purely technical-indicator-based trading model). The assumptions going in were that the Machine Learning model would outperform the technical indicator model, given its ability to learn generalizations of the information seen to predict future values and make trading decisions based on those. However, as seen in figure 4 and based on final portfolio value, this does not hold true. While the ML model outperformed the manual strategy in a small timeframe early on, it loses out in the end. This relative result is expected every time with in-sample data at a high probability, though not 100% given the random effect introduced as part of the random forest learner architecture. The reason why the manual strategy outperforms the ML model is largely due to the trades made and the simple indicator-based logic to determine those trades. The ML model clearly learns how the stock price behaves and largely follows it, while the manual strategy acts purely based on N-minus-1-day data to make a trading decision today. However, when utilizing out of sample data this outcome is not expected, as the ML model generalizes better whereas the manual strategy's indicators will be outdated by the time an out-of-sample decision needs to be made based on them. (Figure 4)

5. Experiment 2: Comparison of effect of impact on machine learning-based strategy

For the second experiment, all parameters were kept the same as for experiment 1 with the exception of impact. Following an empirical approach to determine the effect of impact on trading decisions of the ML model, an initial hypothesis was formed. This hypothesis was based

on two metrics: The final portfolio value and the amount of trades being made by the model. The hypothesis was:

"With impact being considered as an input to the model, the performance of the ML model in terms of final portfolio value should decrease. Conversely, the number of trades being made should decrease as impact increases."

This hypothesis has only partially proved itself as true. In fact, figure 5 and figure 6 depict the outcomes of this experiment. The experiment was to test the two chosen performance metrics across different levels of impact which are largely different from each other: 0 (experiment 1), 0.5 (extraordinarily large impact), 0.05 (large impact), 0.005 (small impact). This experiment was repeated multiple times to compare the outcomes across different seeds that were set. The outcomes in portfolio values have varied highly, likely due to the randomness of the RTLearner. However, when setting the seed and comparing a particular outcome it shows that large impacts cause the strategy to sometimes perform better than the 0-impact model, likely due to the placing of the trades. However, small impact values increase the trading activity of the machine learning model and result in a better performance than the 0-impact model. This outcome was surprising at first but makes sense: At large impacts (0.5, 0.05) the model hardly trades, in fact it returns the benchmark as a 1-trade-only buy-and-hold strategy. The cost of trading is simply too high at that point. At 0 impact, the model trades purely as it learnt within the in-sample period. However, at a small impact, the model issues more trade orders than at 0-impact because the YSELL threshold is met more often due to the logic of "(expected return – impact) <> YBUY/YSELL". This trading frequency leads to a better portfolio value, suggesting there is room for improvement in setting YBUY/YSELL on the machine learning model with 0.000 impact. The conclusion is on the amount of trades made, the above logic applies, however on portfolio value (e.g. on what dates trades are executed) there is randomness due to the learner.

6. Figures

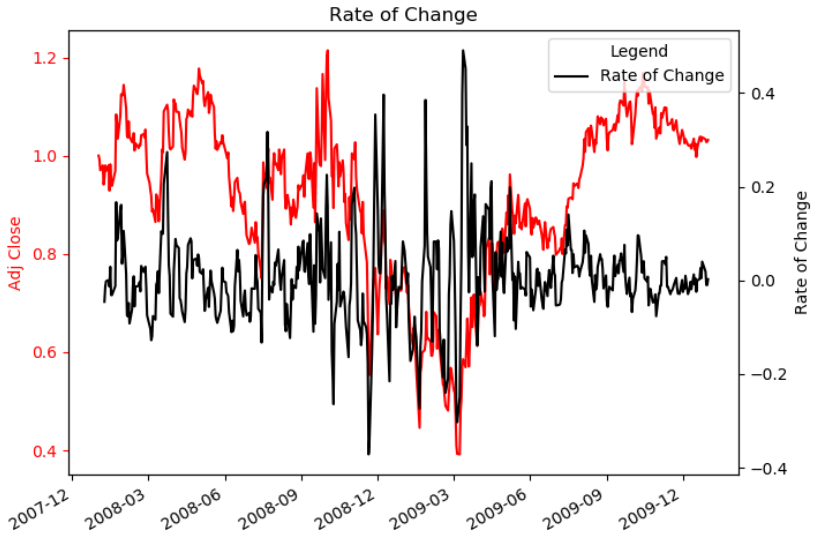

Figure 1: Rate of Change

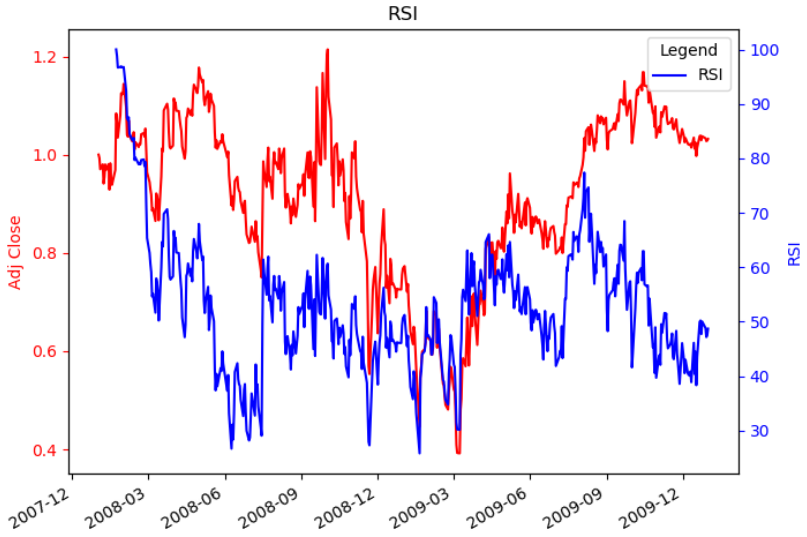

Figure 2: Relative Strength Index

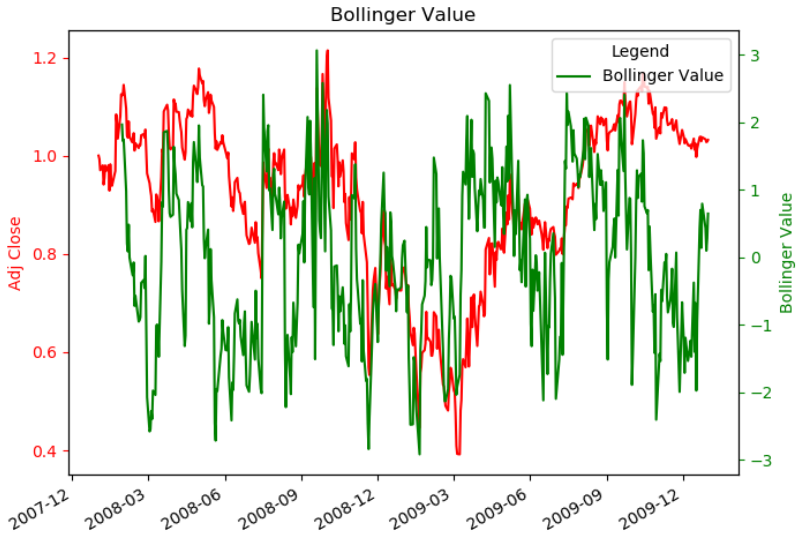

Figure 3: Bollinger Value

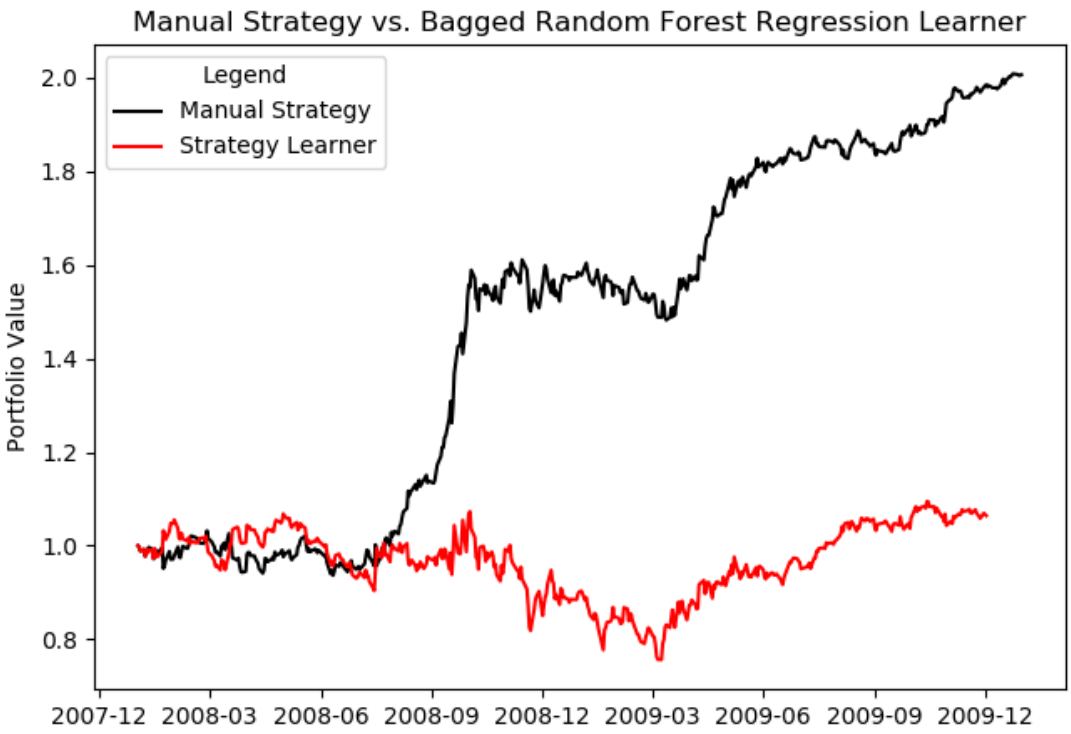

Figure 4: Comparing strategies

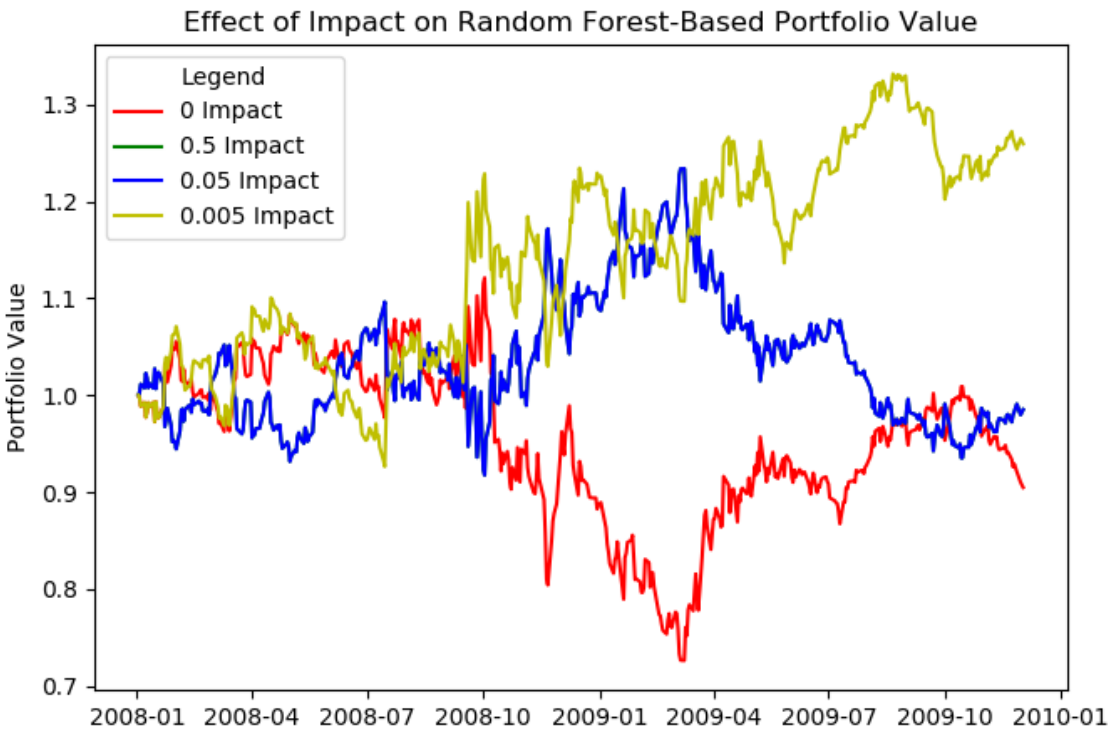

Figure 5: Comparison of impacts

	Portfolio Value	# of trades
Imp. 0.0	90480	65
Imp. 0.5	98520	1
Imp. 0.05	98520	1
Imp. 0.005	125940	95

Figure 6: Trading activity across impacts