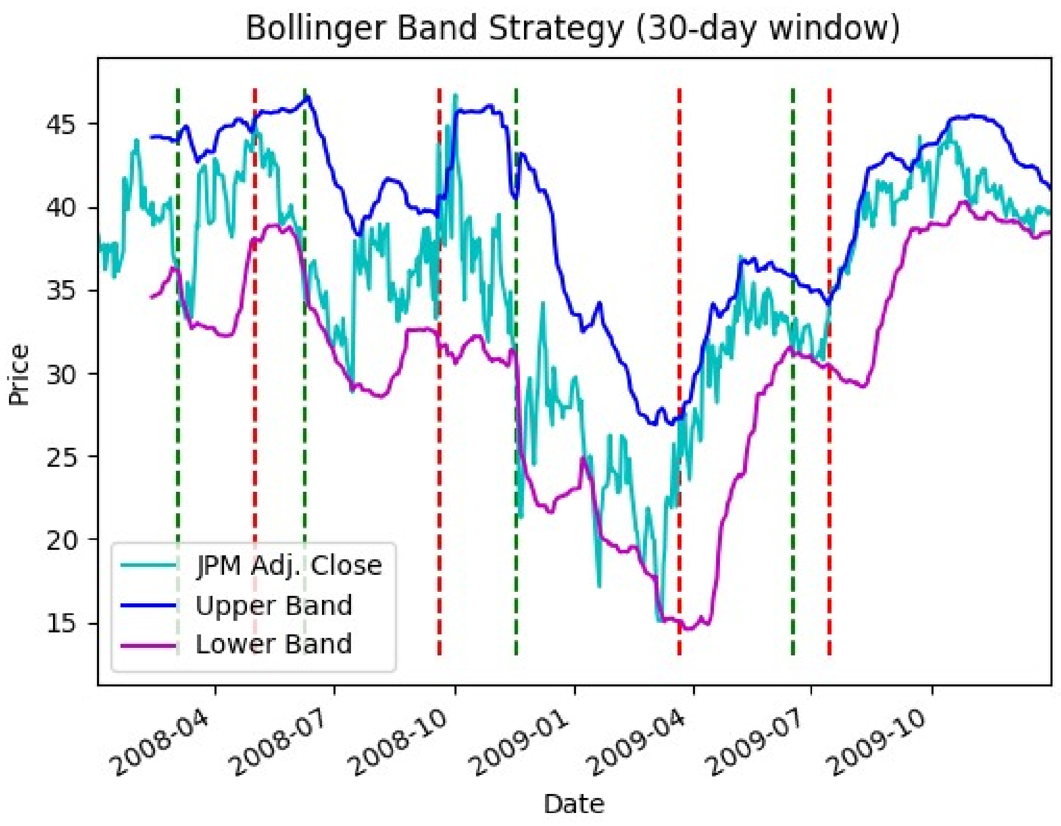
Richard Nelson

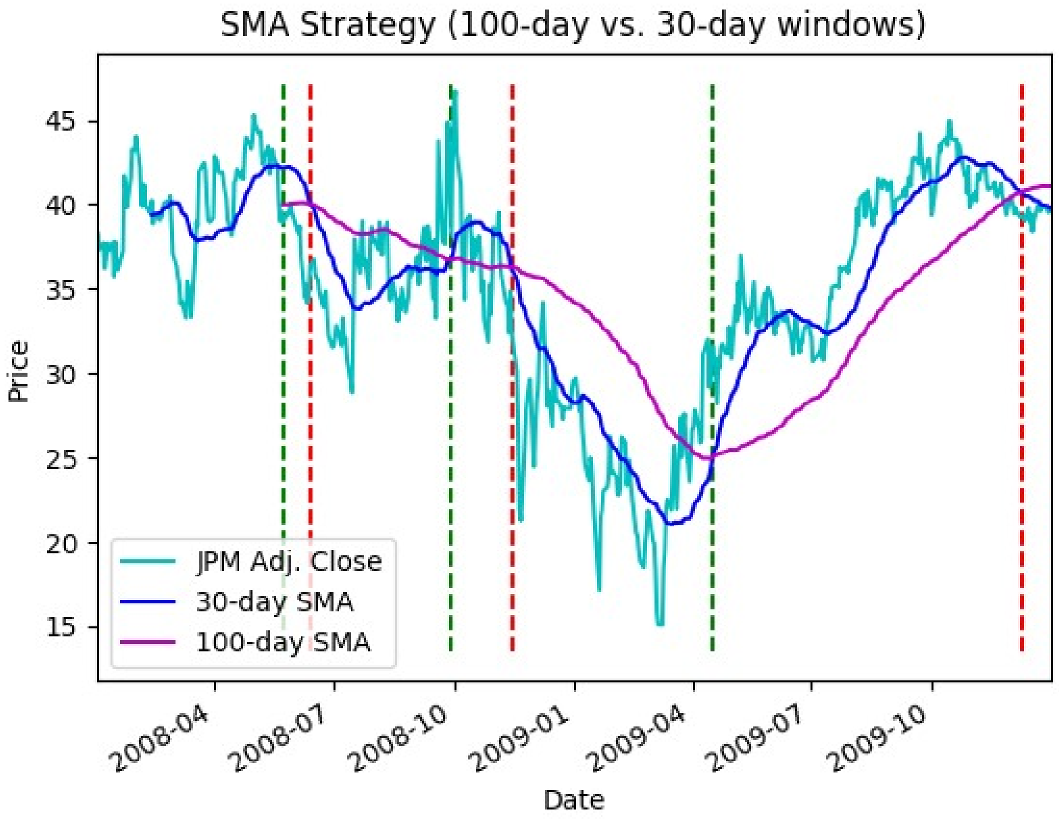
Creating a Machine Learning Algorithm to Trade Stocks

Part 1: Technical Indicators

One system of technical indicators are so-called Bollinger Bands®. In the graph below, the middle line is a 30-day simple moving average. The upper and lower bands are created by multiplying the standard deviation of the 30-day SMA by two. Any time the SMA goes above the upper band or below the lower band, it is considered an event of interest. The theory is that the smaller the distance between the upper band and the SMA, the more likely the stock is overbought, and the closer the stock price is to the lower band, the more oversold the stock. Additionally, the wider apart the bands, the less volatile the stock is likely to be. The green and red vertical lines on the graph below show some examples of events of interest and how one might go long or short depending on when a particular band was crossed.



Another way to use the simple moving average to calculate a technical indicator is to compare a short-term SMA with a longer-term SMA. For example, the graph below shows the 30-day SMA as the blue line and the 100-day SMA as the purple line. Strong momentum is seen when the 30-day SMA crosses the 100-day SMA; each crossover is an event of interest. These crossovers are used by traders to identify shifts in momentum and can be used as a basic strategy for entry or exit. The red and green vertical lines show some long and short entries based upon that type of crossover. A cross below the 200-day SMA can signify the beginning of a downturn, and someone who was long on the stock might exit that position. On the other hand, moving above the longer-term SMA might indicate a pending uptrend for the stock.



These two strategies were refined to create four features:

1. The first feature indicates whether the upper Bollinger Band® has dipped *below* the 30-day SMA within the last twenty trading days. If so, the value will be 1. Otherwise, this indicator will be -1.
2. If the lower Bollinger Band® has risen *above* the 30-day SMA within the last twenty trading days of a given trading day, the second feature will also be 1. Otherwise, it will be -1.
3. If the 30-day SMA has fallen *below* the 100-day SMA within the last twenty trading days, the third indicator will be 1; otherwise it will be -1.
4. If the 30-day SMA has risen *above* the 100-day SMA within the last twenty trading days, the fourth and final indicator will likewise be 1. Else, it shall be -1.

Those four indicators were calculated for each trading day within a given window of time. However, for the machine learning method outlined in this paper, the groups of indicators, known as "X-values", need corresponding "Y-values" for our model to train on. To create the Y-values for training, for each date in the in-sample period the algorithm peeks into the future to see if there is a 0.04% increase or decrease in the value of the stock within the next two days. If there is a 0.04% or greater increase within the next two days, the Y-value for that date will be 1, which represents to our algorithm that we would want buy the stock on that day. If the forecast shows a 0.04% decrease in the value of the stock in the next two days, the corresponding Y-value for that date would be -1, which represents going short on a stock. For all other cases the Y-value will be zero, which indicates doing nothing (neither buying nor selling).

Part 2: Training a Model

The X- and Y-values can now be used to train a model. One method of creating a model is using classification-based Random Decision Trees. The goal of the tree would be to accept a given history of stock data and give suggestions on when to buy and sell a given stock. Although a lone random decision tree may not produce a desirable result, if one utilizes bootstrap aggregation and combined the results of many random decision trees, a suitable model can be developed. This is known as a random decision *forest.*

The single random tree will pick one of the four technical features at random, and then split the group of test cases on whether or not the value of the randomly chosen is greater than zero or less than zero. This creates two distinct groups which represent child nodes of the original group, which is the parent node. This process is repeated recursively for each child node until a newly divided group has a certain minimum number of examples. When this "maximum leaf size" occurs, a "leaf" is created. A "leaf" has no child nodes and represents a potential answer for a proposed query: take the mode of the Y-values of the examples referenced within the leaf and you have arrived at the tree's suggested move for a given input.

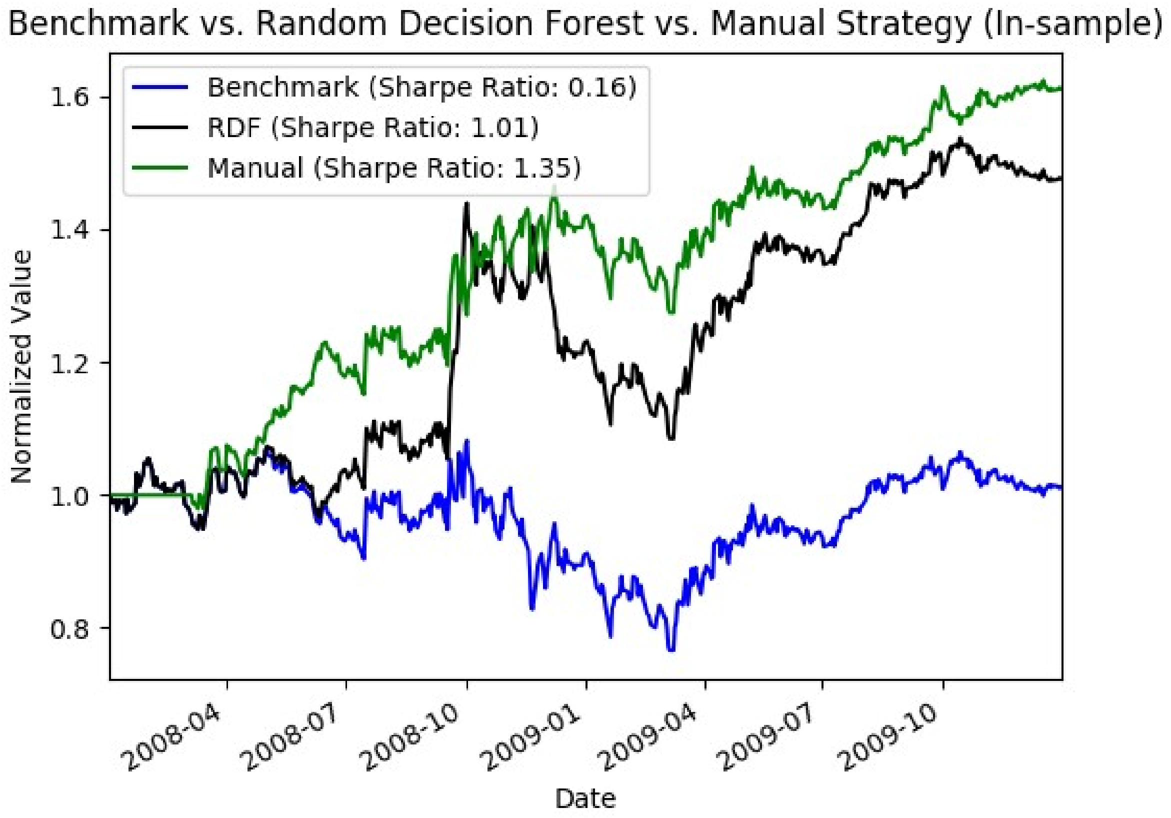
Combining many of these trees and taking the mode of *their* combined suggestions is called bootstrap aggregation, or "bagging." The data shown in this report used a bagging algorithm that created 3,000 different random trees, each with a maximum leaf size of five.

Part 3: In-sample Performance (Experiment 1)

To measure how successful (or unsuccessful) a strategy might be, it's useful to have some benchmarks in mind for our analysis. For the sake of this experiment, only the JPM stock is traded for the specific period from January 1, 2008 to December 31, 2009, which will be our in-sample period. In the graph below, the "Benchmark" line represents the return if one were to buy and hold 1000 shares of JPM from January 1, 2008 to December 31, 2009. This position results in a Sharpe ratio of 0.42, a volatility of 0.05, an average daily return of 0.0013 and a cumulative return of 0.032.

A manual rule-based strategy is one which looks for a specific case and then takes action if certain conditions are met. Contrasted with a machine learning algorithm, it is more difficult to create a manual strategy that correctly reflects available data. Nevertheless, it is good for benchmark comparison to see how a random forest algorithm performs against a manual strategy.

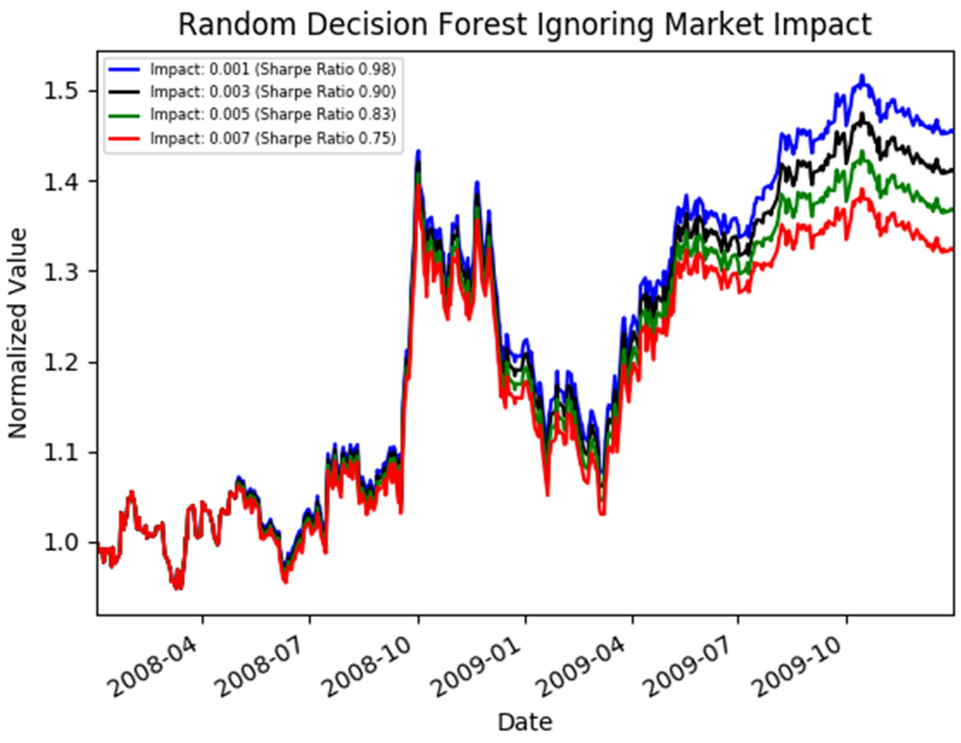
For the manual strategy used in this report, all events of interest for both Bollinger Bands and the 30-day/100-day SMA strategy were calculated. If both sets of results had the same buy or sell indicator within 80 days, the appropriate long or short position was taken. This data series was analyzed first with Bollinger Bands, which created a separate set of potential entry and exit points for going long or short. Then the same series was analyzed using SMA 30-day and 100-day windows, and any crossover that might suggest an entry or exit was recorded. Then the two separate results were compared so that entry or exit events would only be traded upon if *both* the Bollinger band analysis and the SMA analysis made the same suggestion within a certain period of time. It is a somewhat rare event when both Bollinger bands and SMA analysis suggest the same trend within a given period of time, with smaller periods of time of less than 10 days resulting in almost no trading at all. Having that period of time be about 80 days maximized this strategy, resulting in a cumulative return of about 0.6 compared to the benchmark's return of 0.03.



The random forest-based machine learning algorithm did not fare as well as our manual strategy. Although It was able to achieve a respectable 0.45 cumulative return, that did not match the 0.6 cumulative return of the manual strategy; nor did the random forest's 1.01 Sharpe Ratio live up to the 1.35 Sharpe Ratio of the manual strategy. In this case, the manual strategy was both more profitable *and* less volatile.

Part 4: Compensation for Market Impact (Experiment 2)

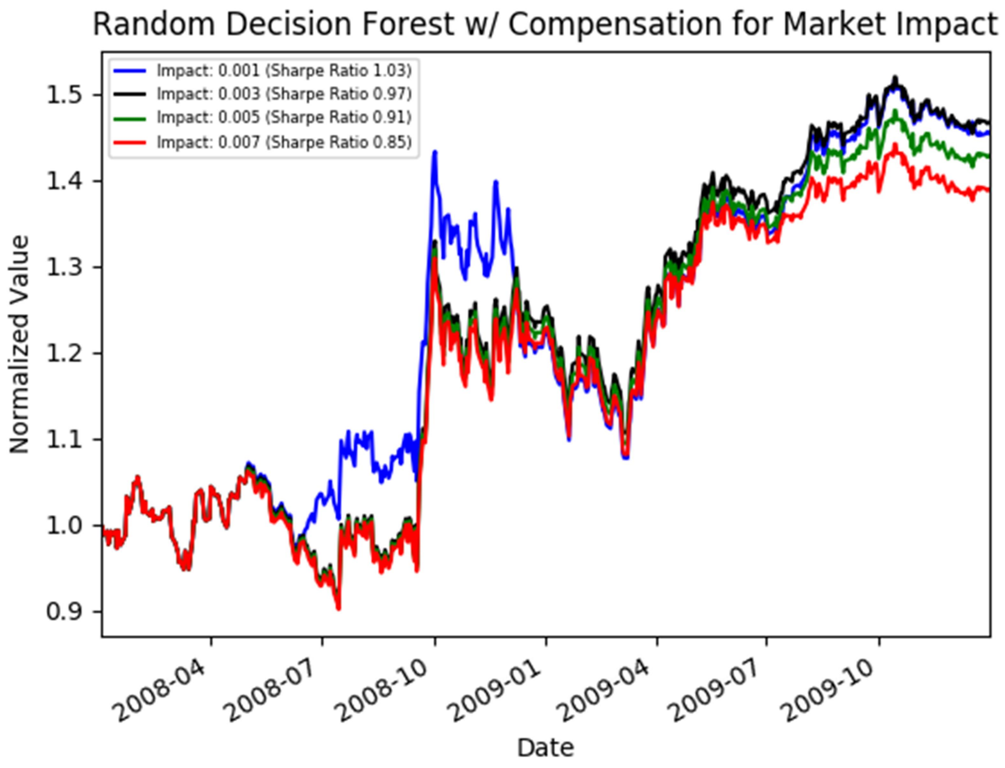
When performing in-sample testing, one effect that is easily overlooked is market impact; that is to say, every trade has some effect on the market. Buying or selling a given number of shares may cause the buy or sell book value for that stock to rise or fall, respectively. Failing to account for market impact can cause your in-sample performance to fall, depending on the amount of impact. The graph below shows the respective returns if the random forest algorithm did not attempt to account for market impact.



Let unaccounted for, the effect of Market Impact has a negative impact on both cumulative returns and Sharpe Ratio.

Generally, the higher market impact, the lower your returns and the higher your volatility will be. However, how much to compensate for when considering market value can be challenging. If you over-compensate then you risk making your algorithm to conservative and thus, not as profitable as it could be.

One way for our algorithm to offset the effect of market impact is to adjust the future return we are looking for when we create our Y-values for training. For example, if we are typically looking for a return of +/-0.04% in a two-day period as the minimum return to make a buy/sell order, we will want that return to increase as market impact increases. By making our algorithm more conservative when there is a greater market impact, we can curb some of the negative effects of high market impact.



Adjusting our "YBUY" and YSELL" values accordingly can aid in mitigating the negative effect of market impact

In the above graph we can see that if we compensate for market impact, it is possible to use to mitigate its negative effects. Even when the market impact was as high as 0.005, we are able to maintain a 0.91 Sharpe Ratio and 0.42 cumulative return, compared to a 0.36 cumulative return and 0.83 Sharpe Ratio if we had not accounted for market impact at all. Between the market impacts of 0.001 and 0.003 there is hardly a difference in cumulative return, although the portfolio was slightly more volatile when there was a 0.003 impact.