Stock Prediction Using Random Forests

John Berroa Felix Beering Tobias Petri Peter Naeve
Leo von Heinz

August 1, 2017

University of Osnabrück

Abstract

The goal of the project is to predict whether a given stock will rise or fall during a given interval. The prediction is generated by computing a number of indicators from a dataset consisting only of a list of prices, and then using these indicators to train a Random Decision Forest. This Random Decision Forest is trained with the bagging technique in order to reduce overfitting. Training and testing is done on either a single stock or the entire dataset, which allowed comparisons of various models and stocks, such as predicting a specific stock price using a model trained on the wider market, or using a stock's model to predict a different stock. Results showed fairly decent predictions (around 70%) for 30 day predictions, but chance level for next day predictions. Code can be found at the GitHub page[1]

[1] `https://github.com/johnberroa/RandomForest-StockPredictor`.

Contents

1 Introduction

1.1 Theory

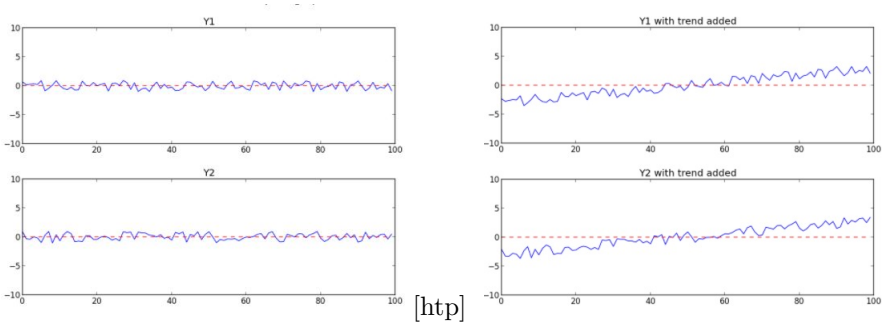
A decision tree is a special type of directed graph; to be more specific, it is a data structure consisting of hierarchically organized nodes and edges. Internal nodes are called split nodes and terminal nodes are called leaves. All nodes except the root have only one incoming edge. Trees can not contain loops. For a given input, a decision tree estimates an unknown property (usually a class label) by asking successive questions about the input's known properties. Each question depends on the answers to the earlier questions. These questions are the split functions in every node, and the terminal leaf node contains the estimation of the unknown property.

The split functions which are stored in each split node can be designed manually for very simple problems, i.e. handpicking properties to split on or calculating a split condition by hand. Common computed split conditions are *information gain* and *gini impurity*, which are based on how good splits are in regards to the unknown property. Nodes are split when these conditions are better than any other node. For more realistic problems, like the one this paper deals with, the splits based on a chosen split computation are computed automatically from example data. To be more specific, in the training phase, the type of parameter the split function splits on is chosen by optimizing a chosen split condition function (e.g. gini impurity) on the training data set. This is done in a greedy manner. Training multiple decision trees in one giant classifier is called a *decision forest*, and is one of many types of ensemble machine learning methods.

To reduce overfitting, bagging is used. Bagging is a short-form of the term "bootstrap aggregation." This works as follows: each tree in the forest is trained on a different subset, which is sampled at random from the training database with replacement. Each tree in the forest will be built slightly different, because of the natural high variance of decision trees. When a decision forest is trained using bagging, it is called a "random decision forest."

1.2 Stock Market Prediction

The stock market is the quintessential example of a time series that people want to predict. This is obvious because there is a lot of money to be made if one could come up with an algorithm that could predict the market. There exists two schools of thought when it comes to market prediction: the efficient markets hypothesis and the behavioral economics. The efficient market theorists believe that all relevant information is already "priced in," or included in the stock price [1]. Therefore, all public information has already been accounted for, including past prices. This means that looking for patterns is futile because anything that the pattern might show, already happened. The behavioral economists think the opposite—the markets are created by thousands upon thousands of slightly irrational beings who have biases and use heuristics to help choose when to



[htp]

(a) Two random time series, $r = -.02$ (b) Same time series with trend, $r = .96$

Figure 1: The Effect of Trend on Correlation [3]

buy and sell [2]. These behaviors can lead to patterns, and these patterns can foreshadow coming "price action" (movement of price).

Analyzing these price patterns is called technical analysis, and the patterns themselves can be found using technical indicators. The simplest indicators are of course price, as well as volume, the number of shares traded in a certain given timespan. "Higher level" indicators such as the *relative strength index* or *moving averages* require calculation from prices. Certain patterns or values of these indicators act as predictors of future price. These indicators also may have meaning over different timescales. For instance, the 200 period moving average on the daily chart is a strong indicator of overall trend and a price that "should" hold up against sellers. On the other hand, the 200 period moving average on a 2 minute chart may hold little if no predictive power at all.

1.2.1 Detrending

Comparing time series data has one major pitfall: if the two time series are trended, as in they are both going up or going down over some period, then they will be correlated even though there may be no relation at all. This is a classic example of the mantra "*correlation does not imply causation.*" This effect is shown in Figure 1. The idea behind detrending can also be summed up as such: if all stocks go up over some time period, the algorithm will learn to predict up, and never actually learn the underlying function that determines stock price.

1.3 Khaidem 2016

Our project was based on a 2016 paper by Khaidem, Saha, and Dey [4]. These researchers used random forests in order to predict stock prices as a classification problem. They picked certain timespan windows, such as 1 month, 2 months, 3 months, and then computed whether or not the stock was higher after that time period than it was at the start. This turns the real-valued prediction problem into a binary classification problem.

In order to train the forest, they input technical indicators they had calculated from the prices. These indicators were:

1. Relative Strength Index (RSI)
2. Stochastic Oscillator
3. Williams %R
4. Moving Average Convergence Divergence (MACD)
5. Price Rate of Change (ROC/PROC)
6. On Balance Volume

The also elected to exponentially smooth prices before training the trees. We disagree with this method. First, they computed the technical indicators after the smoothing process. We find this to be contrary to the idea of the technical indicators—there is no inherent meaning in the results of the indicator. What makes them useful is the "*self-fulfilling prophecy*" that they create. If the prices are smoothed, the algorithm will not be seeing the same indicator values (since they will not be calculated on the original prices) as actual traders. Therefore, the algorithm will not see the same patterns the traders see. Second, the researchers specifically state in the paper that:

> This smoothing removes random variation or noise from the historical data allowing the model to easily identify long term price trend in the stock price behavior. [4]

However, as we know from Section 1.2.1, identifying the trend is exactly what we don't want to do.

The results of their random forests were fairly good. More trees and a larger trading period (the timespan to calculate the up/down classification) yielded better performance. From the provided tables of performance, most of their forests averaged around 90% accuracy. We will train on Apple (AAPL) stock, and they did as well, so it is worth mentioning their performance on AAPL, which was 88.264% for a 1 month target.

2 Method

2.1 Data Preprocessing

2.1.1 Obtaining Stock Prices

We obtained a list of the 500 stocks of the S&P500 index, and used a small Python script to scrape the daily prices for the past seven years. Since the list we obtained was slightly out of date, coupled with the fact that stocks contained in the S&P500 are constantly changing (some companies go out of business or get merged with others), some of our data had erroneous or missing values. We dropped these stocks from analysis.

2.1.2 Creating Indicators

We wanted to replicate the indicators used in [4], as well as add some of our own. Due to an unknown and unsolvable calculation error, ROC was excluded. The end list of indicators we used are:

1. Relative Strength Index (RSI)
2. Stochastic Oscillator
3. Williams %R
4. Exponential Weighted Moving Average (EWMA)
5. On Balance Volume

We stepped through the entire dataset and while doing so also calculated the target values for 1 day ahead and 30 days ahead.

2.1.3 Detrending

Detrending was completed using Scipy's detrend function, `scipy.signal.detrend`[2].

3 Results

We trained different random forest classifiers with different parameters, utilizing the Skilearn class `skilearn.ensemble.RandomForestClassifier`[3]. First we trained without detrended data to get an idea about the different datasets as well as compare to Khaidem et. al., and second with the detrended data to rule out any upward biases. In both cases we bagged and used the OOB score to calculate our error.
We tested four different training paradigms:

[2] `https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.detrend.html`

[3] `http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html`

1. Train a random forest for each stock and test their accuracy on the same stock (Stock-Same)

2. Train a random forest on the entire market and test its accuracy on the market (Market-Same)

3. Train a random forest on the entire market and test its accuracy on a single stock (Market-Stock)

4. Train a random forest for a single stock and test it on another stock (Stock-Stock)

In this document, we only show our result tables for Market-Stock and Stock-Stock. This is based on the fact that most other results are mostly just chance level, and also that Stock-Same is included in the table of Stock-Stock.

3.1 Normal Data

We trained a random forest classifier with 65 trees (same as best performance in [4]), and used six features (the five technical indicators listed in Section 2.1.2, and the Closing price).

Our results were chance level for predicting the next day, but were fairly above chance for predicting a month into the future. See results in Tables 1 and 2.

Table 1: Detrended Market-Stock results

Stock	Market-Stock (1)	Market-Stock (30)
AAPL	89.29%	94.67%
CAT	89.35%	89.13%
BA	90.12%	91.71%
SBUX	89.73%	93.30%

Table 2: Stock-Stock results

$\frac{Stock}{Model}$	AAPL	CAT	BA	SBUX
AAPL(1)	49.05%	50.85%	51.61%	49.28%
AAPL(30)	79.67%	56.11%	54.69%	64.32%
CAT(1)	50.93%	50.28%	47.28%	51.64%
CAT(30)	57.57%	70.54%	50.52%	65.69%
BA(1)	51.15%	48.76%	54.98%	49.34%
BA(30)	62.01%	54.52%	69.13%	65.75%
SBUX(1)	48.07%	51.34%	48.43%	47.89%
SBUX(30)	44.95%	45.25%	40.97%	77.36%

3.2 Detrended Data

Our results were again still chance level for predicting the next day, and were again fairly above chance for predicting a month into the future. 30 day Stock-Same accuracy increased pretty much across the board. See results in Tables 3 and 4.

Table 3: Detrended Market-Stock and Stock-Same results

Stock	Market-Stock (1)	Market-Stock (30)
AAPL	88.86%	96.65%
CAT	90.29%	93.64%
BA	88.92%	94.62%
SBUX	87.98%	93.80%

Table 4: Detrended Stock-Stock and Stock-Same results

$\frac{Stock}{Model}$	AAPL	CAT	BA	SBUX
AAPL(1)	52.21%	49.45%	50.16%	48.57%
AAPL(30)	83.68%	54.77%	64.27%	57.63%
CAT(1)	49.31%	52.48%	49.7%	48.93%
CAT(30)	55.90%	73.01%	57.65%	55.35%
BA(1)	48.27%	48.88%	50.70%	49.04%
BA(30)	54.85%	47.83%	83.93%	53.1%
SBUX(1)	51.54%	50.33%	51.65%	51.13%
SBUX(30)	63.17%	65.15%	65.42%	82.43%

4 Conclusion

Our results were similar to Khaidem et. al.'s results, but in some cases slightly worse. However, we believe our results to be more "true" in the sense that we eliminated the trend and thus prevented the random forests from learning a general rule of "up is always best." It was very interesting to see that the long timespan predicted better than short timespan, especially in the Stock-Stock tests. We believe possible explanations for this would be shorter timespans contain more noise, and/or predictions based on daily timescale technical indicators might take a few days to materialize. It might be possible that training a random forest on hourly or minute data might reveal good results for the next day's closing price, because the data would have more resolution in such a short timescale, and also would have time to "develop", if our second hypothesis mentioned above is true.

References

[1] Efficient market hypothesis. http://www.investopedia.com/terms/e/efficientmarkethypothesis.asp/. Accessed: 2017-07-28.

[2] Behavioral finance. http://www.investopedia.com/terms/b/behavioralfinance.asp/. Accessed: 2017-07-28.

[3] Tom Fawcett. Avoiding common mistakes with time series. https://svds.com/avoiding-common-mistakes-with-time-series/. Accessed: 2017-07-17.

[4] Luckyson Khaidem, Snehanshu Saha, and Sudeepa Roy Dey. Predicting the direction of stock market prices using random forest. *CoRR*, abs/1605.00003, 2016.