Investment and Trading Predictive Model Report

Recommend Stock Tickers

Kelsey Odenthal
September 20th, 2017

I. Definition

Project Overview

The importance of financial forecasting in stock trading has played a huge part in our economic market, challenging invest firms and hedge funds to continuously find better and more strategic ways to predict the outcome of the stock market. The market is one of the most essential aspects of U.S. economics, there are applications in behavioral finance, probability, and statistical analysis for stock trading. The best way to interpret and predict is to look toward past behavior and relate it to present and future behavior, a method utilized in most kinds of predictive modeling.

It's important to pull as much data of stocks as possible, so that there is a wide range of material for the program to learn from. If we relate the range of one year to another, we could potentially give the program insight into the future of any particular stock.

Problem Statement

The intent behind this project is to explore the capabilities of machine learning in terms of navigating the complex stock market. The strategy is to evaluate stocks in the S & P 500 and compare and contrast them with those same stocks in previous years. Understanding the difference between each year and sending that data through a classification/regression model could help with analyzing the data and the path it is on.

The result of this analysis would be a prediction of what stock has improved between years and which stock has depreciated. Based on that classification, we may be able to somewhat accurately predict whether that stock will progress or not in the immediate future. The output would be a list of suggested stocks to pursue or invest in.

The classification model I will utilize for this problem is a Support Vector Machine (http://docs.opencv.org/2.4/doc/tutorials/ml/introduction_to_svm/introduction_to_svm.html), a supervised learning model that takes in classified training data and outputs an optimal vector for categorization of new samples. This "binary classification" (https://machinelearningmastery.com/support-vector-machines-for-machine-learning/) is ideal for what we plan to do and the intended use for Support Vector Machines.

Metrics

The evaluation metrics I am using to analyze the results of the Support Vector Machine model, and most of the other models (Decision Tree, Random Forest) I want to test, will be an “accuracy score” of its prediction, as well as, an Receiver Operator Curve for a visual evaluation. The accuracy score is a computation of subset accuracy by comparing predicted labels to set of true labels, I chose this scoring technique because of its simplicity aligning with the simplistic classification process for the data. The Receiver Operator Curve mathematically works to “show in a graphical way the connection/trade-off between clinical sensitivity and specificity for every possible cut-off for a test or a combination of tests” by having its x-axis showing specificity (false positive reaction, FP/(FP+TN)) and its y-axis showing sensitivity (true positive reaction, TP/(TP+FN)) and taking that area under the curve to measure how well the test can perform, calculated with the trapezoid method (http://journals.lww.com/poctjournal/Fulltext/2012/03000/ROC_Curves_What_are_They_and_How_are_They_Used_.6.aspx). This was selected for its ability to allows the user to see the connection and trade-off for all possible thresholds that maybe couldn’t be acknowledged in a regular classification score, so it’s a more thorough assessment of the model. (https://stats.stackexchange.com/questions/28745/advantages-of-roc-curves).

II. Analysis

Data Exploration

The data used for this problem is mostly sourced from Yahoo! Finance with web scraping. The entirety of the data consists of Date, Ticker, Total Debt to Equity Ratio, Trailing Price to Earning Ratio, Price/Sales, Price/Book, Profit Margin, Operating Margin, Return on Assets, Return on Equity, Revenue Per Share, Market Cap, Enterprise Value, Forward Price to Earning Ratio, Price/Earning to Growth Ratio, Enterprise Value/Revenue, Enterprise Value/EBITDA (Earnings Before Interest, Taxes, Depreciation and Amortization) (http://www.investopedia.com/terms/e/ebitda.asp), Revenue, Gross Profit, EBITDA, Net Income Available to Common Stockholders, Diluted Earnings Per Share, Earnings Growth, Revenue Growth, Total Cash, Total Cash Per Share, Total Debt, Current Ratio, Book Value Per Share, Cash Flow, Beta, Held by Insiders, Held by Institutions, Shares Short, Short Ratio, Short Percentage of Float (or Short Interest Ratio), Stock Change, S&P 500, S&P 500 Change, and Difference. Some of the data pulled from Quandl for extra data, not completely in the essential group but still import, include Date, Open, High, Low, Close, Volume, Ex-Dividend, Split Ratio, Adjusted Open, Adjusted High, Adjusted Low, Adjusted Close, and Adjusted Volume.

A few of the absolute important features in these datasets are Date, Stock Change, S&P 500, S&P 500 Change, and Difference. These are the features that are directly used for the binary classification for the Support Vector Machine. The Adjusted Close feature is to be transformed into a binary set of outperforming (1) and underperforming (0) by taking the Difference between S&P 500 Change (the difference between the current year and

previous year value, divided by the current year value, multiplied by 100, and rounded to 2) and Stock Change (same as S&P 500 change but with stock price).

The amount of examples being used for this model are 254, pulled from an S&P 500 dataset provided by pythonprogramming.net. The feature distribution is as follows

Minimum S&P 500 Adjusted Close: 682.549988
Maximum S&P 500 Adjusted Close: 1658.780029
Average S&P 500 Adjusted Close: 1276.34079123
Median S&P 500 Adjusted Close: 1260.309998
Standard Deviation S&P 500 Adjusted Close: 203.071786097
Variance S&P 500 Adjusted Close: 41238.1503087
Minimum Difference: -86.05
Maximum Difference: 392.74
Average Difference: 5.52381889764
Median Difference: -0.245
Standard Deviation of Difference: 51.9044819893
Variance Difference: 2694.07525058

Algorithms and Techniques

The algorithm being put into use for this problem is, as mentioned previously, Support Vector Machines which is a binary classifier, perfect for this weighing between outperformed and underperformed stocks. This is one of the strongest recommended algorithms for stock analysis because of its classification methods for learning. The algorithm will mainly utilize Stock Price, S&P 500, and Date which are the main points for classifying and fitting the data.

Benchmark

The model I will pursue is to sort by the Adjusted Close of S&P 500 for the current year and the last, stock price for the current year and the last, and evaluate the difference between stock change and S&P 500 change. Further, the data will be tested with these comparisons to see if they underperform or overperform in their own respective timeline. Using a Naive Bayes Model as a baseline example for a simpler predictive model to evaluate and score initially. The result of the Naive Bayes model was an F1 score of 0.661870503597 and the ROC curve plot presented (part 1 of the conclusion).
The curve shows that while not performing perfectly (which is an incredibly unrealistic expectation), the model is still above the diagonal which demonstrates a good amount of true positive with some false positive. A later iteration of Naive Bayes and it's ROC curve (part 2 in the conclusion) which is steeper which shows an improvement toward a true positive rate.

III. Methodology

Data Preprocessing

The extent of preprocessing is establishing the array of features and transforming them for scaling into X. Using the aforementioned Difference features on the S&P 500 data, the Adjusted Close is sorted by “Underperforming” and “Outperforming.” The issue with some of that data is that it appears as NaN, N/A, nan, etc. and it’s important for the program to go through the data and replace all of those problem areas with a zero or an infinite number with a finite number. Then, using X/X.max, the data is scaled to a particular range to fit the vector on. In Z all of our ticker values are transformed into a list to be available for adding to a main ticker predict array.

Implementation

In order to choose what predictive model would best be implemented on this data I ran several classification models in the program and judged them by their “accuracy score” to choose the best one. I pull my data from my “stats with na” spreadsheet for my predictive model. Applying several different classification models required X to be scaled by dividing X by its own maximum (X = X / X.max(axis=0)) and preprocessing, which wasn’t immediately intuitive, most recommendations for preprocessing don’t suggest that particular scaling for X, but it was the most successful for preprocessing. The variables are X, y, and Z. X and y are both used in the predictive model, but the Z variable is only used for calculating investment returns and market returns from the predictive model. X contains the features we care about in the data and y contains our binary array of “Underperforming” and “Outperforming.” It’s important to ensure that all items in these arrays are converted from “nan_to_num” (NaN to numbers) otherwise the data will not train or scale. We perform all of this building and then implement analysis on the data by arranging a “train test split” on X and y. From these results the training data is fitted to Decision Tree Classifier (default parameters), Gaussian Naive Bayes (default parameters), Random Forest (default parameters), and Support Vector Machines (default parameters). For every classifier I used “accuracy_score” with default parameters and for “predicted probability” for my ROC Curve I passed X_test and plotted the curve with y_test and the predicted probability of X_test. I ran into some issues very early when I first implemented my Support Vector Machine so I used “try” and “exception” statements to clear away the value errors I was receiving. I then pulled from my “forward with na” spreadsheet (which is the most recent rundown of the stocks I am analyzing) to pull from these predictive models and with the same scaling and building as the previous spreadsheet data to create (from the data predicted to succeed) an array of recommended stock tickers with the highest performing predictive model (Support Vector Machines).

Refinement

The refinement process for this program had a lot to do with figuring out how to best prescale X for preprocessing because the original plan of simply running the preprocess scaling was returning conversion errors. Eventually, in deducing the best prescaling option, I came across a recommendation on stackexchange to implement a scaling of having X divided by it's max to create smaller values to scale and train, since the difficulty it was facing was that the numbers were ridiculously big. I implemented several parameter adjustments, but felt that the default parameters had more success in terms of scoring and their ROC Curves.

IV. Results

Model Evaluation and Validation

The accuracy scores of all the classification models and the Naive Bayes model were

Decision Tree: 0.929133858268
Gaussian Naive Bayes: 0.661870503597
Random Forest: 0.814960629921
Support Vector Machines: 0.771653543307

The ROC Curves of all the classification models can be seen in the conclusion. A Receiver Operator Curve is a graphical plot that illustrates the diagnostic ability of a binary classifier system as its discrimination threshold is varied (https://en.wikipedia.org/wiki/Receiver_operating_characteristic), a plot of the true positive rate against the false positive rate. An ROC Curve is read as if the "curve follows the left-hand border and then the top border of the ROC space, the more accurate the test," conversely, if "the curve comes to the 45-degree diagonal of the ROC space, the less accurate the test." The curve "shows the tradeoff between sensitivity and specificity" (http://gim.unmc.edu/dxtests/roc2.htm).

The SVM ROC Curve was the best performing compared to the other classifier prediction models, showing an extremely steep beginning curve, which is as close to ideal as a model could feasibly achieve.

The robustness of the model is evaluated based on how the accuracy and results are affected when the outliers are present compared to when they are not.

Decision Tree: 0.889763779528
Gaussian Naive Bayes: 0.629921259843
Random Forest: 0.807086614173
Support Vector Machines: 0.787401574803

The scores are relatively the same with or without the outliers, so it is evident that the models (particularly Support Vector Machines) are robust.

Justification

The predictive model I finally chose to use (Support Vector Machine) is considerably stronger than the benchmark model (Naive Bayes), as can be seen by comparing accuracy scores (0.771653543307 and 0.661870503597, respectively) and the aforementioned ROC Curves (comparison available in conclusion). The final solution provides details from the model and the data

Accuracy: 57.99999999999999
Total Trades: 40
Ending with Strategy: 400962.90104724222
Ending with Market: 400671.70503018575
Compared to market, we earn: 0.0726769605641% more
Average investment return: 0.240725261811%
Average market return: 0.167926257546%

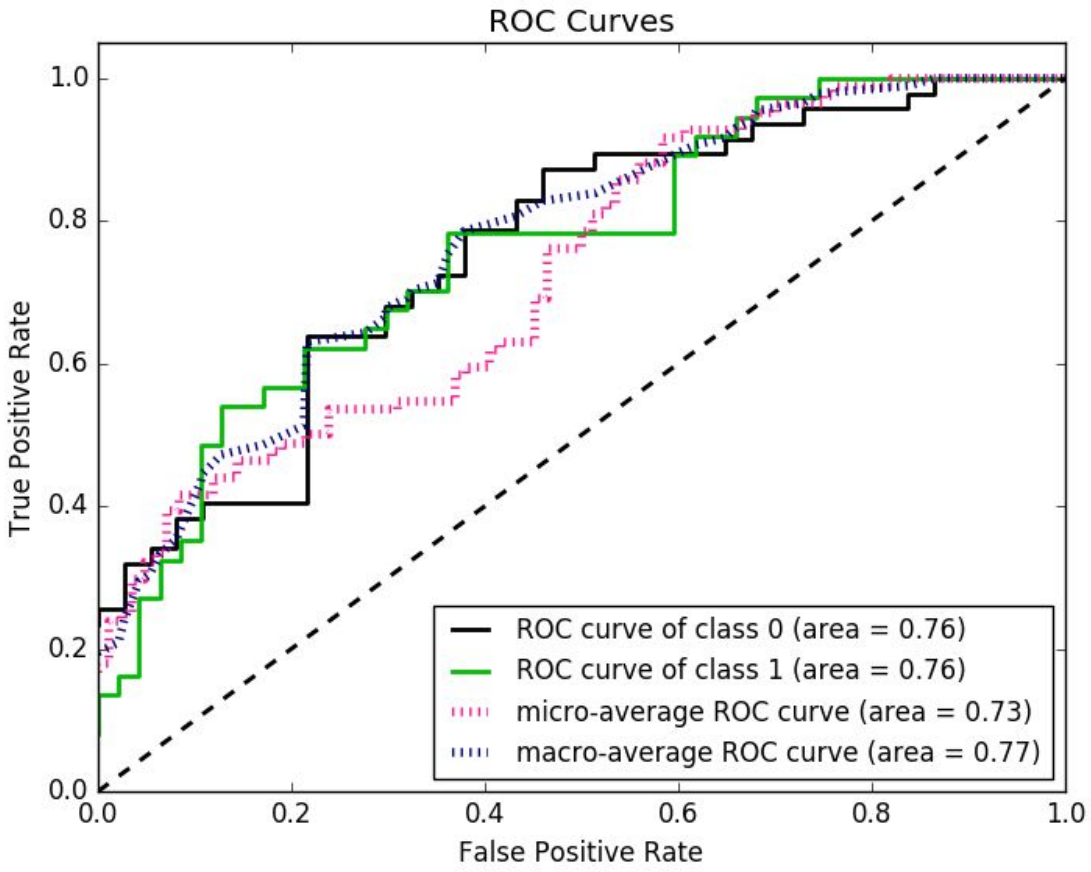
and also includes a list of suggested stocks by their Ticker Symbol.

V. Conclusion

Free-Form Visualization

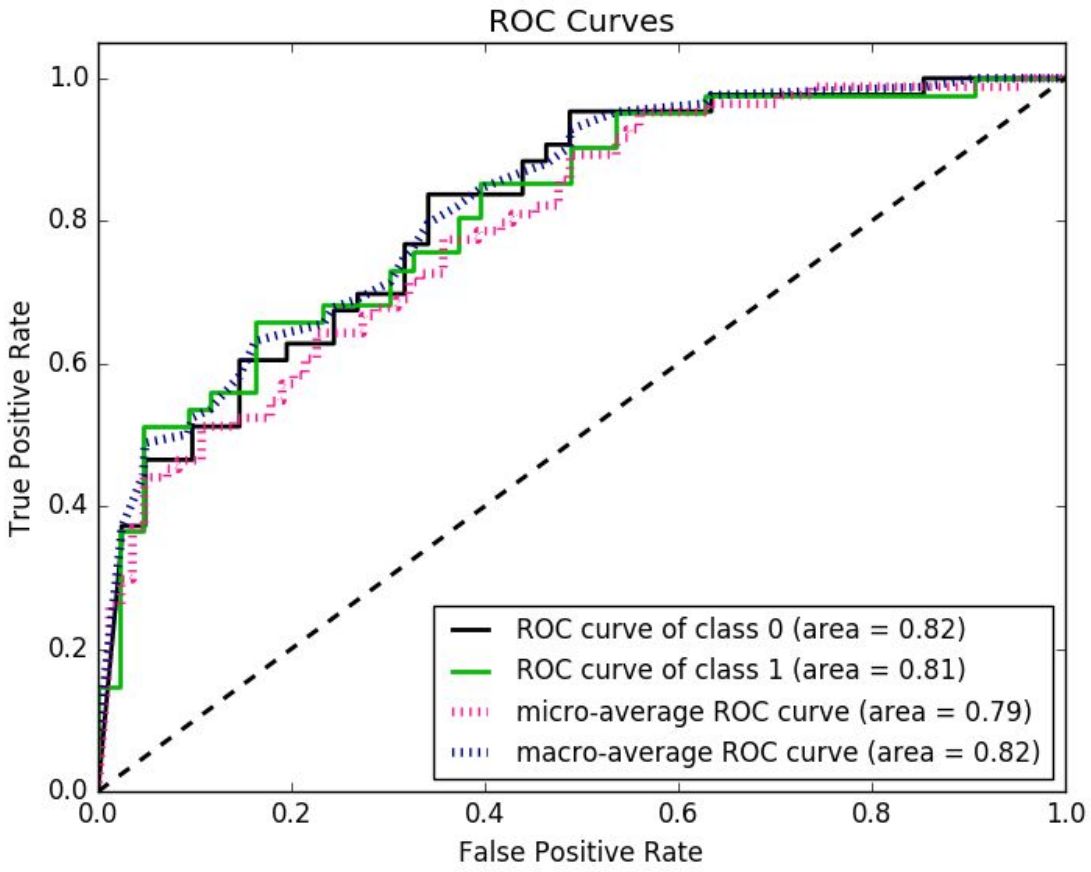
Presented here are the ROC Curves of the previously mentioned classification models that I compared and chose from to predict my data.

Naive Bayes ROC Curve Part 1



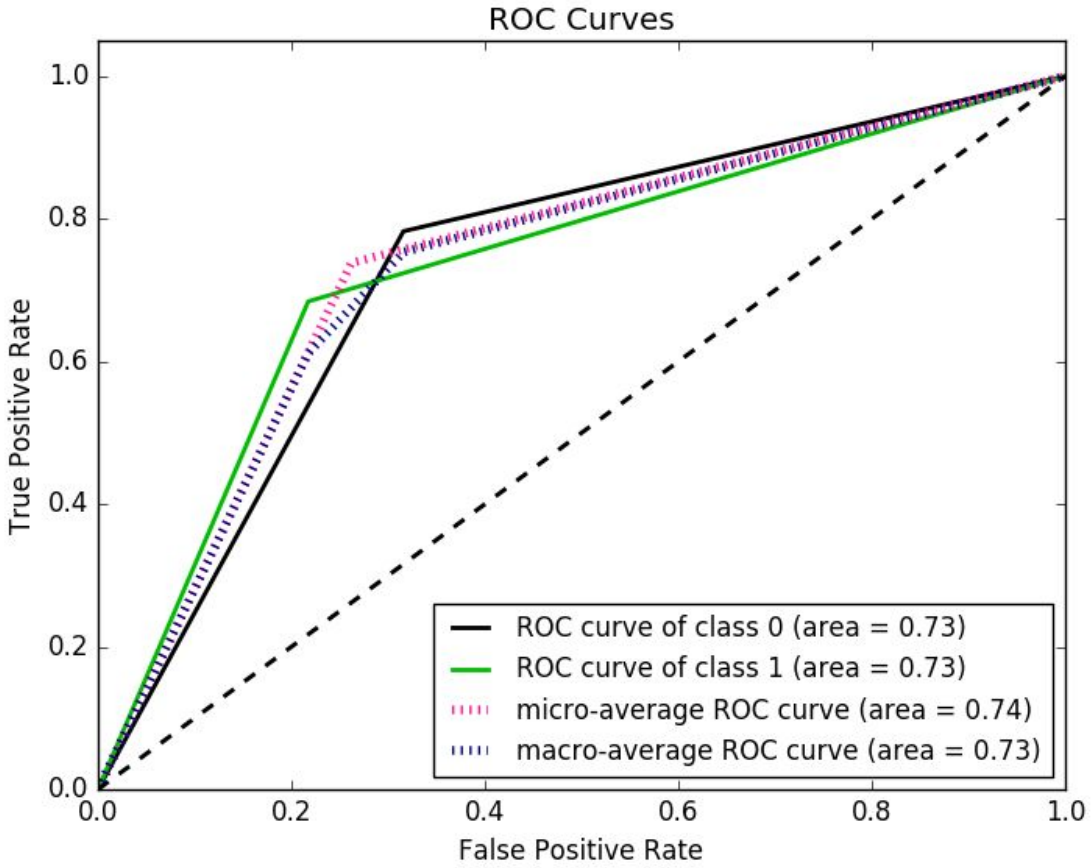
This curve is relatively low and close to the diagonal, while it isn't completely a false positive it still lacks the steep slope of a model with a high true positive rate. Thus, this model doesn't perform ideally.

Naive Bayes ROC Curve Part 2



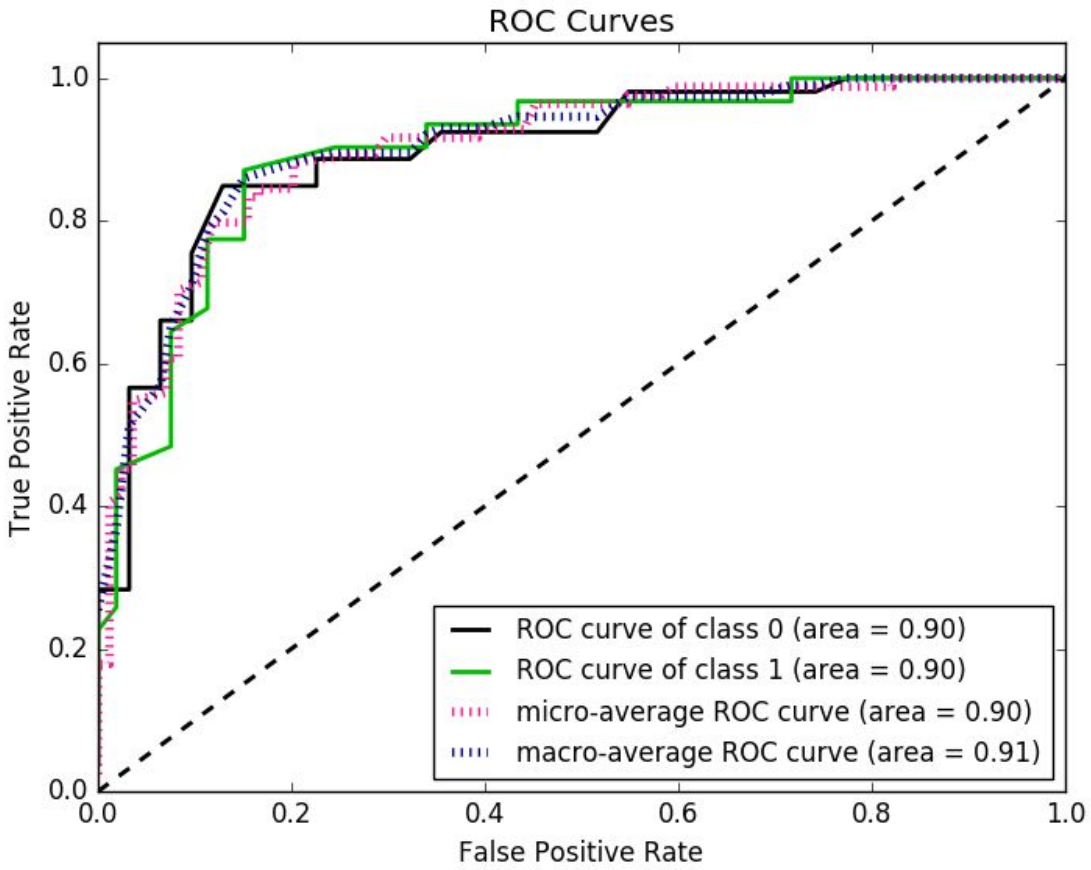
This second iteration graph performs better, but still falls short of expectation, lacking the positive steepness of an ideal curve, as can be seen in the upcoming graphs of well-performing models.

Decision Tree ROC Curve



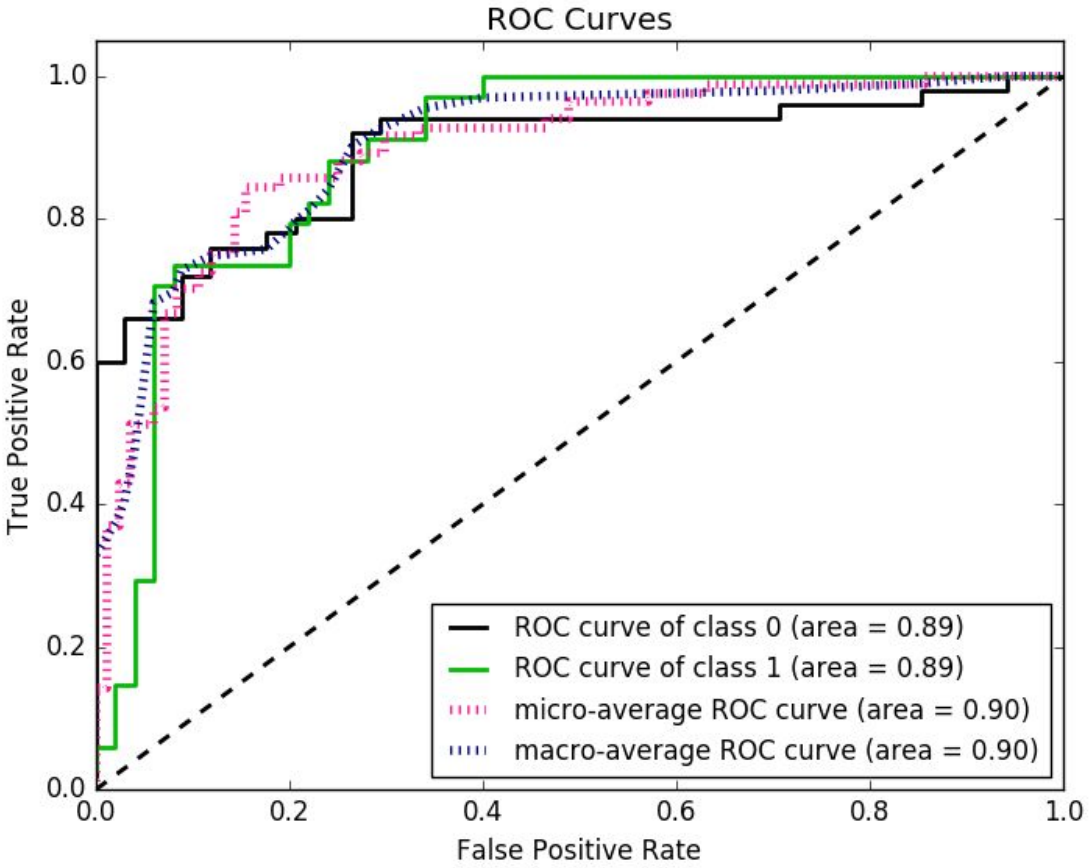
The Decision Tree graph is very interesting and a little odd compared to the other curves we are presented. It does show a steep climb, but not as steep as desired, it's true positive rate fails to meet expectation, proportionally it climbs on the false positive rate too quickly.

Random Forest ROC Curve



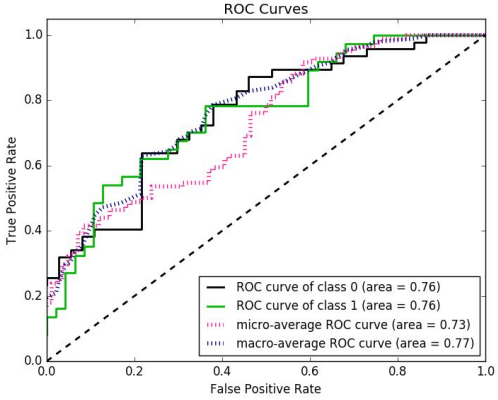
Here we can an ROC Curve for Random Forests, which shows a wild and substantial improvement compared to Decision Tree and Naive Bayes. The difference is a steeper climb on true positive, an important aspect of any well-performing ROC Curve for a predictive model.

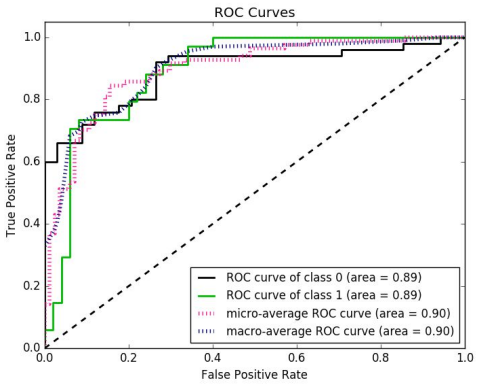
SVM ROC Curve



The Support Vector Machines curve has the steepest slope and the highest and strongest performance of any other model.

Side by side comparison of Naive Bayes ROC Curve and Support Vector Machines shows how a somewhat mediocre curve compares to a high-performing curve between two predictive models. The steep slope is essential to a model that achieves a high proportion of true positive compared to false positive.





Reflection

The model addresses a small segment of an incredibly complex stock system that, frankly, can be a little over a freshly taught machine learner's head. The most difficult aspect of this project was absolutely the web scraping that was required to gather the data that felt necessary for this problem. Even trying to know what data would be necessary was difficult, especially with little prior knowledge about stocks. I originally came into this project hoping to understanding and predict bitcoin but found data difficult to find compared to the average S&P 500 stocks.

I began my work by understanding how to manipulate stock data by exploring Quandl and how I could integrate the API into a program that could evaluate that data. I worked through web-scraping tutorials that helped me find the full data I needed and how to transform the complexities into a binary problem. I taught myself as thorough of an understanding of stock data and classification learning models as possible to create a program that worked as best as it feasibly could in a binary classification circumstance. I sought out plots that could best represent the accuracy of these models, while also seeking the most appropriate and approachable scoring method for the models and fussed with parameters when running into a million errors. Learning the syntax and methods for some of these new systems that I had never used before was at times frustrating, but the process of discovering how to use them was exciting and fun.

Improvement

Once again I would pursue a non-binary classification model in the future, though I am not sure how I would implement, I would be eager to see what the results of that would be. While the current tool is good for what it accomplishes, it doesn't entirely get around to the complexities that make stock prediction so fascinating.