Philip Demeri

Springboard Data Science Career Track

Capstone Project #2

Stock Selection Using Classification and Ensemble Models

Abstract

This project aims to predict the relative future stock returns using data related to the stocks’ historic prices and industry reporting metrics.  At a given time t, given an array of features, the project seeks to predict which stocks will outperform the median stock return over the next month and which will underperform.  On a monthly basis, predictive modeling will be performed on a collection of large-cap US stocks from the financials sector (banks, insurance companies, etc.).  Various supervised predictive models will be implemented, and their accuracies will be reported. The project will also integrate ensemble learning by way of a voting classifier, bootstrap aggregation, and boosting. This workflow will involve the base estimators’ default hyperparameters. The workflow will then be repeated after hyperparameter-tuning the base estimators on a subset of the data. Thus, the scope of the project on an absolute basis is to predict which stocks are expected to outperform the median and which are expected to underperform; the scope of the project on a relative basis is to see if performance improves, measured by mean and/or variance of accuracy, after running the tuned models.

Motivation

This project is adapted from the 2000 Journal of Portfolio Management article “The Decision Tree Approach to Stock Selection,” where the authors implement a decision tree in order to predict which stocks are expected to outperform and which are expected to underperform and then trade those that are expected to outperform.  This project borrows from the article many characteristics, such as the features and the target that were used, as well as the nature of the underlying stocks (US large-cap financials stocks), but this project expands upon the article within more of a machine learning context by employing various predictive models other than just a decision tree in order to explore whether performance can be improved by running models other than the tree-based random forest, aggregating the models’ predictions into an ensemble, or tuning key hyperparameters.

The challenge among many asset managers is to differentiate winners from losers when investing in stocks.  Managers who engage in “bottom-up” trading strategies (i.e., selecting stocks to a portfolio based upon their individual data, not based on broader macroeconomic conditions) generally seek to select those stocks to a portfolio that are expected, based upon different quantitative metrics, to outperform over a future time horizon.  While the underlying data that the metrics are based off of are often continuous, generally they are ranked (or discretized) in order to reduce overfitting.  Additionally, no one model tends to be relied upon when determining final buy or sell decisions; a variety of models is implemented, and then some combination of each model’s result will aggregate into a final decision.

This project implements machine learning models including logistic regression, K-nearest neighbors, and the random forest, and ensemble models including the voting classifier, bootstrap aggregation, and boosting.  The ease with which Python enables their implementation through scikit-learn makes for great attraction to asset managers, who at rebalance need to run a script and return results promptly.

The range of potential clients includes: (1) an asset manager who is looking to launch an investable portfolio that seeks a rules-based allocation scheme involving stock-level fundamental variables (in industry, known as a “bottom-up” model); (2) a sell-side firm that sells research to clients that include banks, insurance companies, and asset managers; and, (3) a retail investor who in his or her own brokerage account is looking to allocate based on which stocks that he or she thinks that will relatively outperform others over the next coming time horizon.

Description of the dataset

The stocks are drawn from the S&P 1500, an equity index that at any given time contains approximately 1,500 stocks (“approximately” due to the presence of splits, spinoffs, etc.). Within the S&P 1500, consistent with the study that the project aims to follow, the financials sector has been chosen (i.e., banks, insurance companies, etc.). Monthly snapshots (more specifically, last trading day of each month) have been gathered from 1994 to 2015, and at each month, anywhere from 100 to 300 financials stocks are members of the S&P 1500.

The stock-level data are acquired from FactSet, a financial data vendor, via FactSet’s Excel add-in. The stock-level data could also have been acquired from other vendors such as Bloomberg or Reuters. The data could also have been downloaded from vendors’ UI’s in the form of a flat file such as an XLS or CSV file.

Data wrangling

The uncleaned data reflect all financials stocks from the S&P 1500. Financials stocks receive a classification of “40” by GICS (Global Industry Classification Standards), as opposed to other classifications for different sectors. Real estate, for example, receives a classification of “60.” Note that the sector classifications are nominal; that is, that 60 is greater than 40 does not suggest that real estate supersedes financials in any manner. Just as easily, the financials sector could have been classified as “FI,” whereas the real estate sector could have been classified as “RE.”

The uncleaned data reflect monthly end-of-month constituents in the index beginning on December 31, 1994, and ending on July 31, 2015. In addition, fundamental data are also provided for each stock: sales-to-price, book-value-to-price, last quarter’s earnings-per-share-to-price, return on assets, current earnings-per-share-to-price, “price momentum,” and the next month’s total return, the latter being the target. Further description of each of these datapoints is provided hereafter.

The first wrangling step is to only keep the necessary columns: effective date, company name, CUSIP, the features, and the target. Company name and CUSIP are not needed for any further steps; they are retained in order to uniquely reference a particular observation. CUSIP is an alphanumeric, nine-character identifier that maps to a particular security. For example, when a company undergoes a name change, the corporate entity structure might remain the same. CUSIP’s provide an effective means of tracking and cataloguing securities.

When data types are checked, the next month’s total return needs to be coerced to float and errors coerced to NaN.

A major consideration regarding stock-level fundamental data gathering is that not always is a given datapoint available at a given point in time. Financial vendors cannot always timely retrieve necessary data on account of, e.g., inconsistencies between corporate filings regarding the same data point, re-statements of a given data point, etc. So, for all of the features and also for the target, the number of nulls needs to be checked. The initial dataframe witnesses 54,435 observations, 50,233 of which are free of NaN’s. It is decided not to impute any of the nulls and instead drop all rows containing at least one missing datapoint.

In order to mimic those features from the study, only one feature needs to be engineered: EPS (earnings per share) momentum, which results when current EPS-to-price is divided by the last quarter’s EPS-to-price. Given that EPS might equal zero, dividing by zero gives rise to inf (or -inf, if the numerator is negative). The decision is to coerce to 9999 or -9999. Practically speaking, given that EPS momentum typically never exceeds 1, and given that the continuous datapoints would eventually be discretized based on quantile, such extreme values of absolute 9999 would almost surely lie in the highest or (if -9999) lowest quantile.

At this point, the previous quarter’s EPS-to-price is not needed for any further analyses or machine learning, and hence the column is dropped.

Consistent with the methodology that this project aims to follow, the target and all of the continuous features are discretized. The target, the next month’s total return for each stock, is coerced to 1 if, for each date, the stock’s return lies above the median return on the same date, and 0 otherwise. So, 1 maps to stocks that “outperform” whereas 0 maps to those that “underperform.”

For each feature, grouped by date, quantiles are demarcated at each 20th-percentile interval: 1 if less than 20%, 2 if less than 40%, …, and 5 otherwise.

So, the dataframe that is then passed onto the EDA phase contains the following columns: date, company name, CUSIP, continuous features, continuous target, discrete features, and discrete target.

The features for this project are:

* Sales-to-price: a company’s last twelve months’ (TTM – trailing twelve months) sales divided by the stock price at time t.
* Book-value-to-price: a company’s TTM book value divided by the stock price at time t.
* ROA: a company’s TTM net income divided by TTM average total assets (average total assets is generally computed by averaging the value of a company’s assets at time t and the value of a company’s assets twelve months ago).
* Earnings-per-share-to-price: a company’s TTM EPS (EPS is computed by dividing TTM net income by the number of common shares outstanding at time t) divided by the stock price at time t.
* Price momentum: a stock’s total return over the past month (total return includes not only share price but also any dividends per share that were paid).
* EPS momentum: a company’s EPS at time t divided by EPS one quarter ago.

As stated earlier, while all features inherently are continuous, consistent with the methodology that this project follows, the features are discretized by percentile based upon each date and are partitioned into five quantiles.

The target for this project is: a stock’s total return over the next month. The target is converted to 1 if the return at a given point in time lies above the median return at the same point in time, and 0 otherwise.

Exploratory data analysis

The analysis focuses on the relationships between and among the features as well as between each feature and the target.

Correlation matrices and heatmaps are constructed for the original features and then for the features when grouped by date, where the average of each feature is computed. The original features witness close to zero correlation amongst each other, whereas the grouped (by date) features witness weakly negative or negative correlations. Note that the matrices and heatmap are applied to the continuous features.

Simple linear regressions are run for each feature versus the target, again in the continuous case. Three of the six features witness statistically significant P-values at the 1% level, yet none of the features attain an R^2 greater than 1.

Line plots of each of the features over time reveal some disturbances around 2000-01 and then again around 2007-08: the former relates to the dot-com bubble, where many web and tech companies observed financial distress, whereas the latter relates to the financial crisis, a period of widespread systematic risk that permeated all aspects of the financial system and the broader economy. To illustrate, for sales-price, for the majority of the period under study, this variable does not exceed 200, yet amid the financial crisis, this variable precipitously rises to above 800. The rise in this variable is intuitive in that it reflects a greater and meteoric decline in stock price (the denominator) relative to periodic company sales (the numerator).

Histograms for four of the six feature witness right skew, as does the target. This finding is consistent with many other phenomena in finance, where purely normally distributed data is rarely seen and instead, right-skewed (few positive outliers) variables dominate.

The right skew is confirmed when skewness is computes: three of the features observe significantly positive skew, two of the features modestly positive skew, and one of the features significantly negative skew.

Kurtosis for four of the features is greater than 1,000. Two of the features witness kurtosis of under 35.

Lastly, the observations are partitioned into “winners” (those where the target equals 1) and “losers” (those where the target equals 0). For each partition, the observations’ quantiles are averaged.

For example, a suppose a stock has a feature array of [1, 4, 4, 5, 4, 3] and a target of 1. This stock witnesses superior feature data (greater than 60th percentile for four of the features, and among, greater than 80th percentile for one of the features) and its next-month total return outperforms the next-month median return. The quantile average is 3.5 ([1 + 4 + 4 + 5 + 4 + 3] / 6). Such averaging is performed for all winning stocks and for all losing stocks, and within each partition, those averages are then averaged.

The quantile average for the winning stocks is 3.01451, whereas the quantile average for the losing stocks is 3.00796 (recall that the 3rd quantile represents percentiles 40 to 60). The difference of means test is conducted in order to gauge how meaningful this difference is. As it is expected that winning stocks should be accompanied by features that belong to higher quantiles, a one-tailed test is conducted. The resulting t-stat and P-value are 1.21 and 0.1122, respectively. Thus, it cannot be concluded that the features’ quantiles between winning and losing stocks significantly differ.

The objective

The project is ready to begin the machine learning phase, where all models are supervised classification models. In each case, the objective is to train on a subject of stocks at the end of a given month and then test on the unused subset at the same end-of-month.  The features are as of the end of the month, whereas the target is each stock’s total return over the next coming month; the target is equal to 1 if the return is greater than the median return, and 0 otherwise.  For each model, its accuracy will be used in order to predict its reliability (e.g., predict 1 when the stock does underperform, or predict 0 when the stock does underperform).

In the case of the voting classifier, bagging, and boosting, the aim is to improve performance over the early standalone cases and/or reduce the variance in the outcome of the accuracies that the base estimators output.

All of the base and ensemble models are then re-run after the key hyperparameters of the base models are tuned, in an attempt to improve performance at the base and/or ensemble level.

Machine learning – overview

The machine learning phase largely involves, at each month, training classification models on 75% of the observations and testing on the remaining 25%, and then looping over all months. The 75/25 partition is accomplished by train\_test\_split.

The three base estimators selected for this project are the logistic regression, K-nearest neighbors, and random forest classifiers. A random state is initialized for purposes of reproducibility.

Ensemble models that this project integrates include the voting classifier, bagging, and boosting. The AdaBoost classifier is selected for boosting. All three classifiers are legs of the voting classifier, whereas only the random forest classifier is the base estimator for bagging and boosting.

From a high level, the machine learning phase is subdivided into (1) running the base and ensemble models using the base models’ default hyperparameters, and (2) tuning the base models’ hyperparameters on a subset of the data (the first 25 months), and then re-running the base and ensemble models on the remaining N – 25 months.

In all cases, the objective is to predict, at a given month, which stocks will outperform (target = 1) and which will underperform (target = 0) in the next month. Consideration is given to the accuracy on the test set, but as a sanity check, the accuracy on the training and validation sets will briefly be examined.

Machine learning – default hyperparameters

The project begins by implementing the logistic regression, KNN, and random forest models, using default hyperparameters, at each month, looping over all months, and then reporting the mean test set accuracy over all months. The results for logistic regression, KNN, and random forest, respectively, are: 58.6%, 56.8%, and 56.1%. Plots of these accuracies when indexed by time indicate a slight incline. Correlations between each of the accuracies and the number of stocks at each month reveal that the accuracies are weakly positively related to stock count (0.26 to 0.37).

The models are then re-run over all months, this time implementing 10-fold cross validation on the training set. The purpose is to gauge the reliability of the results of the test set. For all models, the validation set accuracies closely mirror those of the test set. Further, the training set accuracies are all higher than those of the validation and test sets: 63.3% for logistic regression, 72.3% for KNN, and 97.0% for random forest. These results are intuitive in that the nonlinear KNN and random forest models outperform the logistic regression model when applied to the same set of data. No reason, then, exists to scrutinize the integrity of the data, and hence the project moves forward to ensemble learning.

The voting classifier rests on the hard outcomes of each of the three classifiers without regard for the probability of each outcome. The mean accuracy of the voting classifier over time is 58.1%, which outperforms the accuracy of each of the KNN and random forest models but underperforms that of the logistic regression model.

Bootstrap aggregation uses the random forest as its base estimator. Accuracy is reported both on the test set and on out-of-bag realizations. The accuracies are close to each other, and close to those of other models: 58.1% for test set accuracy and 58.4% for out-of-bag accuracy.

The AdaBoost classifier is selected for boosting, using the random forest as in bagging. The reported accuracy is 56.1%, which lies on the low end of those witnessed thus far.

Machine learning – tuned hyperparameters

The original dataset ranges from December 1994 to July 2015, with monthly periodicity. To tune the hyperparameters for the logistic regression, KNN, and random forest classifiers, the dataset is partitioned into (1) the first 25 months (December 1994 to December 1996) for tuning, and (2) the remaining N – 25 months (January 1997 to July 2015) in order to run the newly tuned models using the same workflow as described earlier (in this phase, however, the 10-fold cross validation phase is skipped, as its function was to measure the integrity of the data within a machine learning workflow). GridSearchCV is implemented over the entire 25-month period using ten-fold cross validation. The tuned hyperparameters are: logistic regression – C; KNN – n\_neighbors; and, random forest – n\_estimators, max\_depth, and max\_features.

With the now-tuned models, re-running each from January 1997 to July 2015 leads to a decline in performance for the logistic regressor but an improvement for KNN and the random forest: 58.1% for logistic regression (previously 58.6%), 58.8% for KNN (previously 56.8%), and 58.8% for random forest (previously 56.1%).

The ensemble models also witness performance improvement, even though no hyperparameters were tuned at the ensemble level: voting classifier accuracy is now 59.4% (previously 58.1%), bagging now 59.1% for both test set and out-of-bag accuracy (previously 58.1% and 58.4% for test set and OOB accuracy, respectively), and boosting now 56.4% (previously 56.1%).

Regarding the variance of the accuracies between the tuned and untuned implementations, no appreciable difference of tendency exists, as is the case for the mean of the accuracies. Hence, the tuned models, broadly speaking, more accurately predict the next month’s outperformers from underperformers without witnessing greater variation among predictions.

Finally, as in the case of the untuned models where accuracy is weakly positively correlated with the number of stocks at each month, such correlation manifests itself in the case of the tuned models, where the correlation ranges from 0.30 to 0.39.

Conclusions

The top-level conclusions are that the classification and ensemble models do predict outperforming stocks at an accuracy of between 55% and 60%. The accuracy increases with adequate hyperparameter tuning, except in the case of the logistic regressor, and generally increases as the number of stocks at a given month increase. The variance of the predictions is not mitigated when ensemble models are introduced, nor do average predictions considerably increase.

The model that predicts most accurately is the voting classifier using the tuned base estimators. The strongest base estimator is the tuned random forest.

Attaining accurate predictions of almost 60% might be considered significant depending on industry standards. A consoling measure is that each base and ensemble model did not witness any aberrant results either with or without tuning. Thus, one might wish to move forward with some of all of these models and the overall setup that governs the data without worrying that one model’s performance might not manifest itself as the actual results play out over time.

Next steps

Many modifications can be made to different aspects of this project.  Regarding the data, the choice to discretize the features into 5 quantiles and the target into 2 (greater/less than the median) is set forth by the methodology that the project wishes to follow.  It is possible to alter the number of quantiles for both the features and the target – perhaps, say, 7 quantiles for the feature and 4 quantiles for the target.  Care should be taken to not increase the quantiles too significantly: the methodology states that design behind discretization is to prevent overfitting by settling on relatively few quantiles.  In the alternative, reducing the number of quantiles likely is not prudent, as it introduces more bias into the outcomes.

All stocks that are observed in this project are drawn from the financials sector.  There are eleven sectors to which stocks belong, such as industrials, telecommunications, and utilities.  It would be interesting to see how this project behaves should a different sector be chosen.  Further, rather than selecting stocks from just one sector, stocks can just as easily be chosen from multiple sectors.  However, the article points out that stocks’ sensitivities to various fundamental and macroeconomic factors are largely, though not entirely, driven by their sector.  The design behind choosing just one sector is to see if powerfully predictive models can successfully separate winning stocks from losing stocks when their separateness is countered by belonging to just one sector.  Stated in the alternative, if the stocks were drawn from different sectors, then factors tend to more easily distinguish among, and by extension predict, winning stocks from losing stocks.

It was observed that accuracy is slightly increasing as the number of stocks at each month increased. An obvious next step is be to select a sector that contains more stocks than the financials sector does.

The decision that the target be the next month’s total return relative to the median total return is in deference to the methodology.  But rather than simply attempt to predict the next month, which generally encompasses just over 20 trading days, a longer future time horizon might be selected, such as the forward-looking quarterly or six-month return.  Considerations then focus on the features: fundamental data are usually reported quarterly; if a longer-duration target is selected, then the features that are captured at time t are then replaced by those that are reported at time t + 3 (months).  In this scenario, likely there would be a shift away from a stock’s own fundamental data as features and instead toward historical time series or macroeconomic data as features.

The above considerations pertain to modifying the underlying data.  Separately, different classification models can be introduced, or even Bayesian models.  Or, there could be more of an emphasis on hyperparameter tuning, whereas this project keeps tuning to a minimum. The project tunes at the base estimator level; it is possible to expand the tuning to the ensemble models (e.g., hard vs. soft voting for the voting classifier).