Philip Demeri

Springboard Data Science Career Track

Capstone Project #1

Predictive Modeling Applied to Rotating between Stock Portfolios that Follow Different Mandates

Abstract

This paper summarizes the findings of applying predictive modeling and machine learning to the age-old question of stock portfolio selection – can a model be devised using macroeconomic and portfolio data that effectively rotates between two underlying portfolios so as to engender performance that is better than simply buying and holding one of the underlying portfolios?

Using over 20 years of monthly data, we partition the dataset into 200 months of training data, 24 months of validation data, and 20 months of test data. We lag many of the features while, at the same time, setting the forward-looking quarterly returns difference between two underlying portfolios as the target, so as to mitigate any lookahead bias. We then implement three supervised learning algorithms: logistic regression, support vector machine classification, and random forest classification.

Tuning the hyperparameters of each model, we ultimately do not settle on any one model that predicts the upcoming returns with any reliable accuracy. However, we uncover interesting observations among the three algorithms in terms of their performance on the training, validation, and test sets: linear models like logistic regression perform marginally on data that are significantly nonlinear; nonlinear models like the support vector machine classifier using the radial basis function and the random forest classifier outperform on the training set, but performance gradually declines on each of the validation and test sets, in connection with more time that elapses between the training set and the unseen data set.

Motivation

Will the stock market go up or down tomorrow? Anyone who can reliably predict this can likely live very comfortably while retiring at an early age. Asset managers employ widely different models that rely on a diverse array of datasets in order to generate. Some devise stock selection model using fundamental data such as a stock’s price-to-earnings-per-share ratio, net operating assets, or enterprise value, many of which are derived from company filings or accounting statements. Others rely on alternative data such as satellite images or natural language processors that, for example, extract trading signals out of the tweets of a company’s executives.

We seek to predict the forward-looking returns difference between two portfolios – and, by extension, invest in the portfolio that we predict is expected to outperform – using machine learning algorithms. We follow the traditional methodology of partitioning historic data that are indexed by time into a training, validation, and test set. We train our models on the training data, tune hyperparameters using the validation data, and then test using the “best” hyperparameters on the test data.

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 macroeconomic variables (in industry, known as a “top-down” 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 the broader performance of the economy in order to beat major market indices.

This project relies on two types of financial data: macroeconomic data, which applies to the broader economy and which has been shown to influence the behavior of asset classes beyond just stocks; and, stock-level data. These data tend to be readily available from numerous financial data vendors, and they are reported with reliable periodicity (in our case, monthly). However, given the easy availability of the data, implementing any model that relies just on these data cannot likely be expected to generate superior performance – the greater the reliability of the data, the greater the number of participants who act on the data, and hence the effect of crowding tends to dilute performance from an investment perspective.

The data

As each of the machine learning algorithms is supervised, a target will be required, in addition to the features. The features for each of the models are:

* Chicago Fed National Activity Index (“Activity”): this monthly index draws from many broad-based indicators that map to manufacturing and industrial activity throughout the United States.
* Consumer Price Index (“Inflation”): more often than not, any news report about inflation likely uses as it source the CPI or a closely related index. This index is updated monthly and is one of the keys metrics that the Federal Reserve relies upon with respect to managing the country’s monetary policy.
* M2 (“Money supply”): discussions about how much of a currency is currently in circulation generally pertain to that which is termed M2, or more colloquially referred to as “money supply.”
* Institute for Supply Management Index (“Manufacturing”): similar to CFNAI, the ISM Manufacturing Index is a broad-based measure of manufacturing activity across the country.
* Butterfly spread: the butterfly spread is a fixed income metric that compares the yields on 1-, 10-, and 30-year notes: 30-year yield – 2 \* 10-year yield + 1-year yield. Interest rates are consistently shown to affect the behavior of the stock market as well as the broader economy.
* Dividend yield spread: this is the only feature that is specific to the underlying constituents within each of the two portfolios. A stock’s dividend yield looks back over the past year at all dividends that were paid over such time period, sums the dividends, and then divides by the current stock price. A portfolio’s dividend yield is a weighted average across the constituents within the portfolio. The spread reflects the difference between the two portfolios.

The data range from September 1998 to December 2018. 244 months of data, indexed monthly, are partitioned into a training set (first 200 months), a validation set (next 24 months), and a test set (last 20 months). The convention in machine learning models that interact with financial data (e.g., stock selection, asset allocation, etc.) is to not exploit train\_test\_split and to instead split into contiguous sets, where the training set precedes the validation set, and the validation set precedes the testing set. “Leakage” (the presence of past data at an existing point in time) has been averted during the wrangling phase, where many of the features were lagged and the target looked ahead to the next quarter’s worth of observations.

The two portfolios

The two portfolios between which the forward-looking quarterly returns spread will be computed are the S&P 600 Growth Index and the S&P 600 Value Index. Attributes such as “growth” and “value” are shown to respond to different factors and correspondingly exhibit different fundamental behavior. Growth stocks tend to be more well-known and outperform during favorable economic times. Growth stocks include tech stocks like Google and Netflix. Value stocks tend to be less well-known and underperform during bull markets. They additionally are more battered during crises. However, coming out of crises, they tend to outperform growth stocks. Value stocks include consumer staples such as Pepsi and Proctor & Gamble.

Both of these indices are drawn from the S&P 600, an index that tracks small capitalization stocks. “Small cap” stocks are at the low end of publicly traded companies in terms of the value of a company’s equity. Companion indices include the S&P 500 for large cap stocks and the S&P 400 for mid cap stocks.

The choice of the S&P 600 rather than the S&P 500 or S&P 400 rests on the increased sensitivity that research has shown to affect stocks at the low end of the market cap spectrum during bull and bear markets. Amid booms, small cap stocks tend to outperform higher cap stocks, and amid drawdowns, small cap stocks are usually more battered.

Wrangling and preparing the data

The data series were obtained from financial data vendors such as S&P, FactSet, and Reuters. These vendors tend the prepare the data in their own right before making it available via API’s, so all of the data was largely self-contained.

Before any exploratory data analysis could be conducted, a fair amount of wrangling needed to be performed:

* It is desired that the data be indexed monthly (though some features or the targets might witness a quarterly or annual percentage change). Not all data series observe monthly periodicity – some are daily or weekly. So, they needed to be downsampled.
* Some of the data series are raw figures and need to be converted to percentage changes. The choice of whether to convert to monthly, quarterly, or annual change rested on intuition and troubleshooting against the target.
* The data are available over a range of time intervals. The time interval that was selected for further analysis is that which contains no missing values for any features or the target. After all of the individual series (imported into Python as dataframes) were concatenated, the relevant (or, said correctly, irrelevant) observations were dropped. Additionally, all indices were converted to datetime and, as stated, indexed monthly, which were considerations for appropriate concatenation.
* One of the series did contain missing observations. Not every day is a trading day for all financial instruments, even for those instruments that map to the same country. Quasi-holidays like Columbus Day or Veteran’s Day witness this occurrence. So, those nulls were forward-filled. Interpolation would not have made sense, given that no price change occurs on a non-trading day.
* The target and a few of the features needed to be computed based off of the raw data. For example, the target is the upcoming quarterly returns spread between two portfolios whose constituents tend to behave differently based on macroeconomic and fundamental data. And after engineering the features, the original features were discarded.
* The computed target was continuous, yet for the eventual logistic regression and SVM classifier, this target needed to be converted to Boolean. The Boolean target was mapped to 1 if the returns spread was positive and 0 otherwise.
* Additional cleaning involved pivoting, re-indexing, and parsing dates.

Exploratory data analysis

The EDA portion of the project was largely concerned with adjusting the features across time in order to find the appropriate lags relative to the target. To elaborate, it is often the case that macroeconomy does not immediately exert influence on financial assets. Sometimes a few months, if not longer, pass before the effects are felt. Up to this point, the features and the target are contemporaneous; that is, all datapoints are as of time t. The only manipulations so far have been that some of the features reflect monthly or annual percentage changes and that the target reflects the forward-looking (to avoid lookahead bias) three-month returns spread. For example, the dataset as of end-of-December 2018 would reflect, e.g., the percentage change in inflation from December 2017 to December 2018 and the forward-looking quarterly returns spread between the value and growth portfolios from December 2018 to March 2019.

The predictive models are classification algorithms that seek to correctly predict in which of two portfolios to invest each month. The more highly correlated the features are with the target, while maintaining low pairwise correlation among themselves so as to reduce the effects of multicollinearity, the more accurate the model’s predictions should be.

How to lag the features was based on which lag produced the highest correlation between each feature and the target over the entire time interval. Given the repetitiveness of finding correlations at various lags, a custom function was written that, when a series or list for the feature was passed through, returned the correlation between the feature and the target at no lag, a one-month lag, a two-month lag, and a three-month lag. Almost all of the features witnessed higher correlations with the target when lagged.

The fixed income feature, the “butterfly spread,” is known to more substantially lag equity portfolios than macroeconomic variables do. No custom function was written for this feature. Lags were incrementally tested as far as 12 months, and ultimately the eight-month lag was settled on.

The final dataset (that is, the one that is to be used for predictive modeling) contains the following features at the following lags:

* Activity: two months
* Inflation: two months
* Money supply: two months
* Manufacturing: zero months
* Butterfly spread: eight months
* Dividend yield spread: two months

The heatmap for the features shows that the strongest correlation in terms of magnitude is 0.53, which indicates that the features do not witness multicollinearity.

While the predictive models are classification algorithms, EDA focused on each feature’s linear relationship with the continuous target, and the joint effects of the features on the target. After a simple linear regression was run using each of the features, the output revealed that all of the features are significant at the 5% level and that all but one are significant at the 1% level. A multivariate linear regression revealed that four of the features are significant at the 5% level, two are significant at the 1% level, and the model’s adjusted R-squared is 21.4%.

Also of interest is the behavior of the dataset during the financial crisis. The period from June 2007 to June 2009 has been subset and proxies for the crisis. Whereas the heatmap over the entire period shows no significant correlation among the features, during the crisis, the correlations increase in magnitude precipitously: of the 15 pairwise correlations, only four are less than 0.5, while six are above 0.7. Further, the multivariate regression provides additional evidence of multicollinearity: all features lose their statistical significance (only one is just slightly above 5% while the rest are largely insignificant), whereas the model’s adjusted R-squared rises tremendously to 76.9%. It can be expected that during this period, more accurate predictions should arise. And if so, this would be welcome: especially during times of financial crises, investors tend to sell risky assets and fly to the safety of bonds and cash instruments. A model that can predict with high accuracy between stock portfolios would be highly sought after.

The objective

Largely restating from earlier: given macroeconomic data, and given the performance of two underlying investable indices, we wish to devise a predictive model that beats the performance of merely buying and holding either of the two underlying indices.  More specifically, we wish to allocate between the S&P 600 Growth Index, an index of U.S. small cap growth stocks, and the S&P 600 Value Index, an index of U.S. small cap value stocks, based on macroeconomic and portfolio-level variables. Predictive modeling will be accomplished by three machine learning models: (1) logistic regression, and (2) SVM classification, and (3) random forest classification. Each model will use the macroeconomic and index-level variables as its features, and the target will be the returns spread between the two indices, coerced to Boolean (1 if value outperforms growth, 0 otherwise). The features witness monthly periodicity, lagged as stated above, and the target is quarterly and forward-looking (i.e., T + 3 months relative to T).

Machine learning – logistic regression

Implementing the logistic regression model primarily focused on the hyperparameter C and the penalty term (i.e., l1 versus l2). Looping through values of C ranging from 0.0001 to 1000, and between the l1 and l2 penalty, we find that the accuracy for the training set is slightly increasing as C increases, yet it never exceeds 60%. The accuracy for the validation set is less than that for the training set as C increases, and the decline is not monotonic. A combination of C = 1000 and l1 penalty generates the highest accuracy on the validation set, yet this accuracy is only 33%. Then, using the same hyperparameters, we observe an accuracy of 60% on the test set.

We then re-run the logistic regression model using five-fold GridSearchCV on the training data using the same hyperparameter set. Upon doing so, we now arrive at an optimal combination of C = 0.01 and l2 penalty. Accuracy for the training, validation, and test data are 54%, 50%, and 60%, respectively.

The lack of discernible and reliable trends in accuracy when applying model to the training and unseen data suggests not only that the model is not likely to generalize well to unseen data, but also that the data likely are not linearly separable. We then turn to nonlinear classifiers in hopes of improving performance.

Machine learning – SVM classification

We implement the SVM classifier using the linear kernel and then using the radial basis function kernel.

With the linear kernel, and as we loop over the same values of C as witnessed in the logistic regression model, we see that C equal to 500 attains the highest accuracy over the training set. But when we apply this model to the validation and test sets, we observe accuracies of 33% and 65%, respectively. So as with the linear regression model, over an array of regularization levels, while we expect that performance will decline on the validation set, performance improves, and actually exceeds that on the training set, on the test set.

The RBF kernel witnesses performance that we would expect when we train our model on training data, tune hyperparameters on validation data, and then test on unseen test data. Looping over the same values of C as before, and now integrating values of gamma of 0.00001, 0.0001, 0.001, 0.01, 0.1, 1, 2, 5, and 10, we attain perfect predictions over high values of C and gamma. We see similar behavior when we shift to the validation data, higher accuracy is seen at high values of C and gamma. At C = 1000 and gamma = 10, we attain 71% accuracy. However, performance markedly declines on the test data.

The SVM with RBF kernel performs best with low regularization and less smoothness of the decision boundary in the transformed space. These two conditions engender the overfitting that we see in the accuracy among the data sets: 100% on the training set, 71% on the validation set%, and 40% on the test set.

Machine learning – RF classification

For the random forest classifier, we vary the hyperparameters n\_estimators (100, 150, 200, 250, and 300), max\_features (2, 3, and 4), and max\_depth (2, 3, and 4). Using GridSearchCV over ten folds, we settle on an optimal combination of 150 trees, 2 maximal features, and a max depth of 3. With this combination, accuracy over the entire training set is 80%. Declining performance using these hyperparameters is seen on the validation set (54%) and the test set (45%).

Conclusions

Whether a predictive model is “good” as it relates to performance on within-sample and out-of-sample data is industry-specific and rather subjective. But broadly, specific to asset management, models that cannot be trusted to reliably predict which of two portfolios will outperform the other over an upcoming time horizon will likely not be implemented. While no agreed-upon cutoff exists as to what accuracy threshold constitutes a model that performs reliably on unseen data, 70% tends to be the echelon for most models, provided that the models can reach this threshold on numerous realizations of unseen data. Specific to this project, while the SVM classifier with the RBF kernel did perform at slightly above 70% on the validation set (one unseen path), the same model performed at 40% on the test set (another unseen path). Each of the models in this project witnesses accuracy levels that might be attained as much by randomness as by the power of the models themselves. The linear models are at or above 60% accuracy on the test set, but each falters on the validation set.

An important takeaway from the project is that the logistic regression and SVM models performed better at higher levels of C (and gamma), which are consistent with less shrinkage on the regressors (and a less smooth boundary in the transformed space). Another important takeaway is that while the logistic regression models never exceeded 60% accuracy even on the training data, often the SVM models performed at above 70% for high C and gamma.

An observation found in the EDA and wrangling phases of the project was that the correlations between each feature and the target largely increased in magnitude during the financial crisis (denoted in the project as June 2007 through May 2009). It was hypothesized that predictions might improve during this period. However, not much of a material difference exists between the accuracy during this period and that during the broader training period, during which time the financial crisis occurs.

The logistic regression and linear SVM models witness unusual results in that performance is actually best on the test set. Likely this is due to chance on the part of the test set and the weakness of these models as related to interacting with the inherent nonlinear separability of the data. By contrast, the SVM with RBF kernel and random forest models perform as expected – the highest accuracy is on the training set, with declining performance on the validation and then on the test sets.

As far as next steps, many revisions can be explored in hopes of attaining a more reliable model:

* Alter the training, validation, and test set intervals: although tweaking the lengths of these intervals might be attempted, the hope is to engineer a model that performs well on numerous training, validation, and test intervals. That performance in this iteration was lackluster suggests that altering intervals should not be explored.
* Add or delete features: all but one of the features are macroeconomic and not directly connected to the constituents of the two portfolios. Perhaps including more stock-level data might better reflect the unique features of each portfolio so as to differentiate them.
* Different classification models: Bayesian models were not attempted in this project. To do so, distributions would need to be fit to each feature, rather than merely accept the feature data “as is.”
* Alter the periodicity of the data: shortening the periodicity to weekly or daily would immediately increase the number of observations. However, many of the macroeconomic data are reported monthly, and not one of the features is updated daily. Daily or weekly imputation would be needed, but then the issue arises as to what imputation method is most appropriate.