Financial Analysts Journal

ISSN: 0015-198X (Print) 1938-3312 (Online) Journal homepage: https://www.tandfonline.com/loi/ufaj20

Machine Learning for Stock Selection

Keywan Christian Rasekhschaffe & Robert C. Jones

To cite this article: Keywan Christian Rasekhschaffe & Robert C. Jones (2019) Machine Learning for Stock Selection, Financial Analysts Journal, 75:3, 70-88, DOI: 10.1080/0015198X.2019.1596678

To link to this article: https://doi.org/10.1080/0015198X.2019.1596678

Research

Financial Analysts Journal | A Publication of CFA Institute
https://doi.org/10.1080/0015198X.2019.1596678

Machine Learning for Stock Selection

Keywan Christian Rasekhschaffe and Robert C. Jones, CFA

Keywan Christian Rasekhschaffe is a senior quantitative strategist at Gresham Investment Management, LLC, in New York City. Robert C. Jones, CFA, is chair and chief investment officer of System Two Advisors, LP, in Summit, New Jersey.

Machine learning is an increasingly important and controversial topic in quantitative finance. A lively debate persists as to whether machine learning techniques can be practical investment tools. Although machine learning algorithms can uncover subtle, contextual, and nonlinear relationships, overfitting poses a major challenge when one is trying to extract signals from noisy historical data. We describe some of the basic concepts of machine learning and provide a simple example of how investors can use machine learning techniques to forecast the cross-section of stock returns while limiting the risk of overfitting.

Disclosure: The authors report no conflicts of interest.

CE Credits: 1

Practitioners using quantitative factor models have struggled since the 2008 financial crisis, and many traditional factors have ceased to be profitable. As a result, some market participants are looking beyond the traditional quantitative approaches to stock selection. As popular quantitative factors have become less reliable, many practitioners are developing models that can dynamically "learn" from past data. Dynamic models and *ad hoc* factor-timing approaches, however, face some valid criticisms (see, e.g., Asness 2016). Investors have used econometric techniques, such as regression, for many years, but few have found success with dynamic models based purely on these techniques. The causes may be that financial data are inherently noisy, that factors can be multicollinear, and that relationships between factors and returns can be variable, nonlinear, and/or contextual. These features make it difficult for a linear regression model to estimate any dynamic relationships between potential predictors and expected returns.

We believe machine learning algorithms (MLAs) may provide a better approach than linear models. These techniques have been available for a long time. Indeed, Frank Rosenblatt invented the perceptron, a neural network that could classify images, in 1957. Over the ensuing decades, a series of developments enabled advances in practical machine learning and its utility:

- Computing power has increased roughly according to Moore's Law since the 1970s.
- Data availability has increased exponentially, and storage costs have decreased significantly.
- Novel techniques from disciplines such as computer science and statistics, in conjunction with increased computing power and data availability, have led to powerful new algorithms.

Machine learning algorithms have proven to be more effective than traditional statistical techniques in many areas outside finance. A few examples are voice recognition (e.g., as used by Siri and Alexa), image recognition (e.g., as in self-driving cars), and recommendation engines (e.g., as used by Amazon). Deep learning algorithms now exceed human accuracy for many image classification tasks. An MLA called Deep Blue first beat the reigning human chess champion, Garry Kasparov, in 1997.

It used "brute force" computational speed to evaluate thousands of possible moves and countermoves. More recently, a deep learning neural network called AlphaZero used pattern recognition techniques to become the world chess champion. Unlike Deep Blue, which was programmed to assess the value of various positions, AlphaZero was given no domain knowledge yet was able to teach itself to become a chess master in only four hours by playing against itself.

What Is Machine Learning?

Machine learning is an umbrella term for methods and algorithms that allow machines to uncover patterns without explicit programming instructions. In the case of stock selection, modelers supply a variety of factors that might help in forecasting future returns and use MLAs to learn which factors matter and how they are related to future returns. Machine learning offers a natural way to combine many weak sources of information into a composite investment signal that is stronger than any of its sources.

In recent years, computer scientists and statisticians have developed and refined several machine learning algorithms—such as gradient boosted regression trees, artificial neural networks, random forests, and support vector machines (see Appendix A for definitions). Most of these algorithms have two important properties:

1. They can uncover complex patterns and hidden relationships, including nonlinear and contextual relationships that are often difficult or impossible to detect with linear algorithms.
2. They are often more effective than linear regressions in the presence of multicollinearity.

Although research on the application of machine learning techniques in finance is relatively active, many articles in the field focus on the application of a specific algorithm. Wang and Luo (2012) provided a detailed overview of using the AdaBoost algorithm to forecast equity returns. Batres-Estrada (2015) and Takeuchi and Lee (2013) explored the use of deep learning to forecast financial time series. Moritz and Zimmerman (2016) used tree-based models to predict portfolio returns. Wang and Luo (2014) demonstrated that forecast combinations from different training windows can be effective. Heaton, Polson, and Witte (2017) discussed the application of deep learning models to smart indexing. Alberg and Lipton (2017) proposed forecasting company fundamentals (e.g., earnings or sales) rather than returns—because the signal-to-noise ratio is higher when forecasting fundamentals—which allowed them to use complex machine learning models.

Several articles have studied the benefits of nonlinear models for timing factor returns. Miller, Ooi, Li, and Giamouridis (2013) and Miller, Li, Zhou, and Giamouridis (2015) found that classification trees can be more effective than linear regressions when predicting factor returns. They also presented evidence that combining linear and nonlinear models can be even more effective. Furthermore, they demonstrated that cross-sectional models that incorporate these factor forecasts can outperform static factor models. We reached similar conclusions in this study, but we followed a different approach. Instead of explicitly forecasting returns of univariate long–short factor portfolios, we used cross-sectional factor scores (characteristics) to predict the cross-section of returns.

Gu, Kelly, and Xiu (2018) examined the efficacy of machine learning techniques in the context of asset pricing. The authors forecasted individual stock returns with a large set of company characteristics and macro variables. Because they used total returns rather than excess returns as the dependent variable, they jointly forecasted the cross-section of expected returns and the equity premium. They examined how various machine learning methods perform and found that nonlinear estimators result in significantly improved accuracy when compared with ordinary least-squares (OLS) regressions. They attributed the improvement to the ability of machine learning models to uncover nonlinear patterns and to their robustness to multicollinear predictors. Although our conclusions are similar, we focused solely on the cross-section of excess returns independent of the equity risk premium. Accordingly, we used only individual equity characteristics and excluded macro variables. We believe this approach reduces noise and the risk of overfitting. In agreement with Gu et al. (2018), we found that many machine learning algorithms can outperform linear regression, but rather than concentrating on the performance of individual algorithms, we highlight the benefits of combining forecasts generated from different algorithms and training windows. We found that forecast combinations outperform constituents in the United States and in other regions.

The Perils of Overfitting

Overfitting occurs when a model picks up noise instead of signals. Overfitted models have good

in-sample performance but little predictability when applied to unseen data. Although machine learning techniques can uncover subtle patterns in past data, overfitting presents a major challenge. When one is training an algorithm, finding patterns in the data that also generalize out of sample is important. Relationships between factors and returns are frequently noisy, and many potential factors exist, which increases the dimensionality of the problem. In contrast, many machine learning applications in other fields, such as image recognition, have high signal-to-noise ratios. For example, some image classification tasks (such as classifying dogs versus cats) have error rates below 1%.

Because of the low signal-to-noise ratios in forecasting stock returns, avoiding overfitting is particularly important. **Figure 1** shows in-sample and out-of-sample error rates for a gradient boosted regression tree classifier with the use of simulated noisy data. The in-sample error is always lower than the out-of-sample error. As the number of boosting iterations increases, the errors always decline in sample, and they become negligible after around 400 boosting iterations. In sharp contrast, the error rate in the holdout sample first decreases but then increases after approximately 50 boosting iterations. This point is where the algorithm begins to overfit the data.

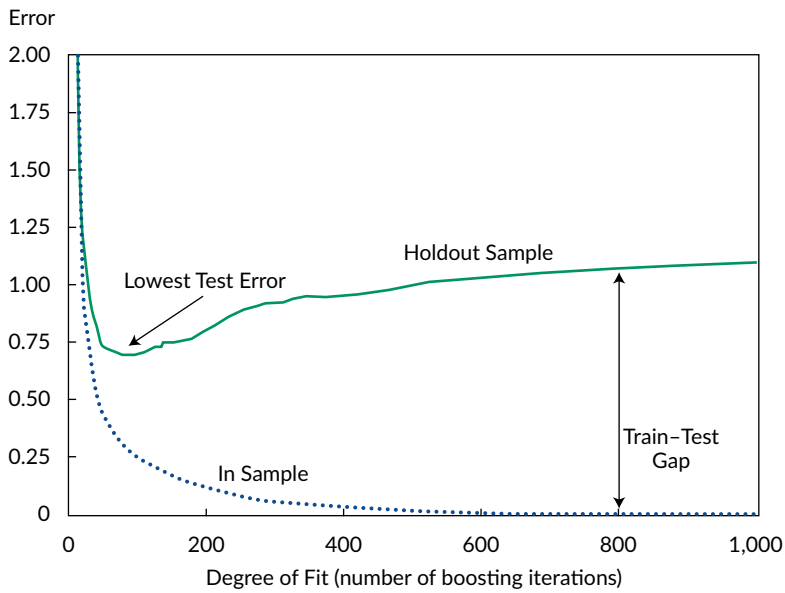
The simulated example has a relatively high signal-to-noise ratio when compared with forecasting stock returns. With lower signal-to-noise ratios, out-of-sample results diverge much faster from in-sample results.

Nevertheless, this example illustrates the perils of evaluating predictive performance on the basis of the training set: Overfitting can make the results look much better than they are likely to be in any real-world application. Next, we discuss two approaches that can help mitigate the perils of overfitting: forecast combinations and feature engineering.

Forecast Combinations. Many successful machine learning algorithms are ensemble algorithms that rely on bagging (e.g., random forests) or boosting (e.g., AdaBoost).[1] These algorithms generate many forecasts from weak learners and then combine the forecasts to form a strong learner. Dropout (see Appendix A and Srivastava and Hinton 2014), a tool related to these ensemble algorithms that is used to prevent overfitting in neural networks, also harnesses concepts of model averaging. We believe we can achieve even greater diversity by combining forecasts from different classes of algorithms and by training them on different subsets of the data. If many different algorithms trained on many different training sets all find similar patterns and reach similar conclusions, we can be more confident that the forecast is robust and not the result of overfitting.

The efficacy of forecast combinations has been widely documented in the statistical literature.

Figure 1. In-Sample/Out-of-Sample Error vs. Degree of Fit: Gradient Boosted Tree Classifier



Notes: Simulated noisy data were used. Note that more iterations on the *x*-axis allow the algorithm to better fit past data.

Clemen (1989) summarized the empirical evidence on forecast combinations as early as 1989:

> The results have been virtually unanimous: combining multiple forecasts leads to increased forecast accuracy. . . . [I]n many cases one can make dramatic performance improvements by simply averaging the forecasts.

Makridakis and Hibon (2000), in an analysis of a competition to forecast 3,003 time series, showed that forecast combinations usually outperform even the best constituent forecasts. Timmermann (2006) provided a framework for determining when forecast combinations are likely to be effective—namely, when different forecasters use different data and/or techniques and the forecast biases are relatively uncorrelated. In these circumstances, forecast combinations can provide both more information and less noise.

Of course, traditional multifactor models already contain multiple forecasts because each factor implies a distinct forecast. The approach recommended here, however, takes this concept much further by including many different forecasting techniques and training sets as well as numerous factors.

We might increase forecast diversity along the following several dimensions.

Combining forecasts from different classes of algorithms. Many machine learning algorithms—especially ensemble algorithms (e.g., random forests)—already use forecast combinations to yield better results than single algorithms can produce. By also combining forecasts from different classes of algorithms, we should be able to detect different types of relationships between features and labels.

Combining forecasts based on different training windows. Forecasts from different estimation windows can pick up different market conditions and often have low correlations. Windows can be defined on a temporal, seasonal, or conditional basis. Combining forecasts from different training windows also reduces forecast variance, potentially increasing risk-adjusted returns.

Combining forecasts that use different factor libraries. By dividing a large factor library into several subsets, algorithms can explore a greater variety of patterns, potentially leading to new insights.

Combining forecasts for different horizons. Different factors are important over different forecast horizons. For example, fundamental factors are usually more important over long horizons, whereas technical factors tend to be more predictive for short horizons.

Feature Engineering. Feature engineering uses domain knowledge to structure a problem so that it is amenable to machine learning solutions. Such engineering requires considerable expertise and can be difficult and time-consuming, but it is essential for developing robust forecasts. Feature engineering determines which problems we ultimately ask the algorithms to solve and which algorithms we use to solve them. It is one of the most effective ways to overcome overfitting because it allows us to increase the signal-to-noise ratio before training the algorithms.

Feature engineering is where domain knowledge flows into the process. In the context of stock selection, it can decide such questions as the following: What are we trying to forecast? Which algorithms are likely to be most effective? Which training windows are likely to be most informative? How should we standardize factors and returns? And which factors are likely to provide valuable information?

In the material that follows, we briefly discuss some of these issues. The goal is to provide an overview rather than a comprehensive discussion. The possible variations among these decisions are as far ranging as the imaginations and expertise of financial modelers. Ultimately, however, the quality of the decisions determines the success of the effort.

What are we forecasting? To limit the risk of overfitting, often the best approach is to forecast discrete variables with machine learning algorithms.[2] Rather than predicting returns, as we would with linear regression, MLAs usually predict categories—that is, outperformers versus underperformers—which tend to be less noisy than returns. Users may want to select an additional category, such as market performance, or even more categories to reflect various levels of performance, but each new category increases the risk of overfitting and may provide little additional accuracy for data that are as noisy as stock returns.

After the selection of categories, a second decision involves how to define these categories. If we are interested in the cross-section of returns, we would define categories by dividing stocks into outperformers and underperformers for each date in the training set. We might also define these categories within sectors or industries to reduce noise. Standardizing the factors (i.e., characteristics) in a similar fashion

is frequently advisable. Most investors want to outperform net of risk, so a natural approach is to define performance categories for risk-adjusted excess returns. These categories might include simple volatility-adjusted returns or alphas from an appropriate risk model, such as the capital asset pricing model, the Carhart (1997) four-factor model, the Fama and French (2017) five-factor model, or the MSCI-Barra model described in Morozov, Wang, and Borda (2012). Using risk-adjusted returns can improve the signal-to-noise ratio and thereby enhance forecasts across both time and market sectors.

A third decision involves the forecast horizon. Selecting a forecast horizon implies optimizing for that horizon. Short forecast horizons are suited to low-capacity, high-turnover strategies; long horizons are more suited to high-capacity, low-turnover strategies. Short horizons offer more training periods, which are helpful when trying to uncover subtle patterns in noisy data. The forecast horizon should also reflect the frequency of the underlying predictive data (or factors). For most stock selection applications, an appropriate forecast horizon runs from daily to quarterly.

Which algorithms should we use? Wikipedia lists more than 100 machine learning methods, and the list is growing all the time.[3] Machine learning is a rapidly evolving field, and discussing the pros and cons of so many algorithms is well beyond the scope of this article. In general, however, we want our final forecast to incorporate a variety of algorithms that use a variety of techniques. Knowing the exact relationship between returns and features is impossible *ex ante*. Combining forecasts from different classes of algorithms provides insurance against misspecification. This aspect is particularly important when dealing with financial data, in which the signal-to-noise ratio is low and relationships are difficult to verify empirically with a high degree of certainty.

Ensemble methods have shown promise as applied to financial data and in many other fields. The goal is to combine weak learners, either by equal-weighting forecasts (bagging) or by accuracy-weighting forecasts (boosting), to produce a strong learner. The strong learner tends to do better than any of its constituent weak learners. Both boosting and bagging can address the bias versus variance trade-off encountered by all supervised learning problems. Bias is caused when the estimation method does not effectively capture fundamental relationships in the data (underfitting). Variance is an error arising from small changes in the training set, which means the estimator does not learn relationships that generalize out of sample (overfitting).

Bootstrap aggregation (bagging) independently fits estimators such as decision trees (weak learners) onto random subsets of the training set. Each of the weak learners is overfitted, but errors resulting from overfitting tend to be reduced when weak learners are combined to create a strong learner. Boosting sequentially fits estimators on the training set and gives more weight to misclassified observations in successive boosting rounds (see Schapire 1990). The strong learner is an accuracy-weighted average of the weak learners. By giving more weight to more successful learners, boosting can address bias. If we allow the boosting algorithm to overweight successful weak learners too aggressively, however, this benefit will be more than offset by increasing variance. Because of this trade-off, boosting algorithms tend to require more careful parameter tuning than bagging algorithms and conservative learning rates tend to be preferable for stock selection out of sample. Most boosting algorithms also take longer to train than bagging algorithms because they often must be run sequentially whereas bagging algorithms can be run in parallel.

Boosting and bagging ensembles can harness diverse base algorithms as their weak learners. Different algorithms capture different features of the data. Some algorithms are relatively simple and linear; others are extremely complex and can potentially uncover highly nonlinear relationships. Furthermore, although we often want to capture complexity, the more complex algorithms typically require a higher signal-to-noise ratio than simple algorithms and/or more training data to learn effectively. The aim of using multiple algorithms and approaches is to capture relationships that are both simple and complex while minimizing the risk of overfitting.

In general, modelers should focus on algorithms that have demonstrated prior success with noisy data and that have well-known benefits and pitfalls. As a practical matter, a good idea is to use algorithms that are available in software libraries and that have been tested in a variety of applications. An off-the-shelf algorithm is not likely to be appropriate, however, without further parameter tuning. Because the signal-to-noise ratio tends to be low for stock selection, algorithms often need to be parameterized in a way that severely limits the algorithm's potential to overfit. As with all investment strategies, practitioners should avoid optimizing models in sample.[4]

Which training windows should we use? As a general rule, we want to train machine learning algorithms on data that are likely to be representative of the expected future environment. For example, we may want to use training sets that are close in time to the expected environment, exhibit similar macroeconomic conditions (e.g., valuation levels, liquidity conditions, or growth dynamics), or occur at the same time of the year (to capture seasonality). Conversely, if we are uncertain about the expected future environment, we will want to train the algorithms on the largest, longest, and broadest dataset possible to capture a variety of environments. Such an approach will require longer runtimes, however, and may fail to capture period-specific patterns.

Another consideration is cross-sectional variation in patterns. For example, if we think different regions or sectors will exhibit different relationships between factors and returns, we will want to train the algorithms separately on data from these different regions or sectors. Conversely, the risk of overfitting grows when we make the training sets too granular. For example, it probably makes sense to have separate training sets for US stocks and Japanese stocks but not for US tech stocks and Japanese auto stocks.

Which factors should we include? Domain knowledge is critical both for selecting factors and for structuring them to increase the signal-to-noise ratio. To minimize runtimes and to limit overfitting, practitioners should feed algorithms only data that are likely to be related to future stocks returns. Such data would include factors related to future economic success (fundamental factors) and factors related to future supply and demand (technical factors). Because MLAs are usually quite good at handling collinear data, we can certainly include many similar factors if we are uncertain as to which ones are most relevant, although piling on too many similar factors will increase runtime.

Domain knowledge also helps when structuring data to maximize the signal-to-noise ratio. For example, if the goal is to select stocks, not to pick industries or sectors, we should adjust the data accordingly. For many factors, forcing them to be sector or industry neutral reduces variance without significantly decreasing average factor returns (see Asness, Porter, and Stevens 2000). Hence, neutralizing factors at the industry level can increase the signal-to-noise ratio, making it easier for algorithms to learn relationships between factors and expected returns.[5]

An Example

In this section, we describe an approach to stock selection that uses some of the techniques discussed previously in a cross-sectional setting. Our goal is to demonstrate the general power of machine learning for stock selection, not to argue for the effectiveness of any specific decision related to feature engineering. Practitioners have plenty of room to apply their own expertise and potentially achieve even better results than those reported here.

Data. **Table 1** provides summary statistics for our sample. It includes small-, medium-, and large-capitalization stocks, with an average of 5,907 stocks per month, in 22 developed markets. Our factor library consisted of 194 factors (i.e., company characteristics) that were assembled by IHS Markit from diverse sources. We included 21 deep value factors, 18 relative value factors, 10 factors focused on earnings quality, 26 factors capturing earnings momentum, 26 factors focused on historical growth, 35 liquidity factors, 29 management quality and profitability factors, and 29 technical price-based factors. Excess returns are defined as returns above the risk-free rate and come from Barra. Our sample period is 1994 through 2016, with walk-forward forecasts beginning in 2004 (to allow a 10-year training period). The forecast horizon and data frequency are both monthly. Combining forecasts with various horizons may be beneficial, but we focused on a monthly horizon to provide an adequate training set and to be consistent with typical factor research. We allowed two days between model estimation and trading to account for runtime and parameter tuning.

Feature engineering. **Figure 2** outlines the general workflow executed each month in a walk-forward framework. We started by defining three training sets:

1. The *recent* training set included all data from the prior 12 months.
2. The *seasonal* training set included all data from the same calendar month over the prior 10 years.
3. The *hedge* training set included data from the bottom half of performance over the prior 10 years based on the first two training sets.

A single month could appear in more than one training set. For instance, the same month last year appears in both the recent and seasonal training sets.

We developed these training sets and trained our algorithms independently for four separate

Table 1. Summary Statistics, 2004–2016

Jurisdiction	Average No. of Stocks	Average Company Size ($ billions)	Average Weight in Global Portfolio
Australia	206	4.6	3.5%
Austria	28	3.6	0.5
Belgium	41	7.1	0.7
Germany	110	11.5	1.9
Denmark	40	4.7	0.7
Spain	65	9.9	1.1
Finland	39	4.8	0.7
France	114	14.7	1.9
Great Britain	356	8.0	6.0
Greece	41	2.2	0.7
Hong Kong SAR	287	4.7	4.9
Ireland	16	5.7	0.3
Italy	75	8.1	1.3
Japan	1,052	3.6	17.8
Netherlands	39	8.8	0.7
New Zealand	26	1.7	0.4
Norway	62	3.7	1.0
Portugal	16	4.7	0.3
Singapore	101	3.1	1.7
Sweden	87	5.1	1.5
Switzerland	56	17.1	0.9
United States	3,050	5.7	51.6

Notes: The average number of stocks represents the average number of securities in the sample each month. US data start in 1988; all other data start in 1994.

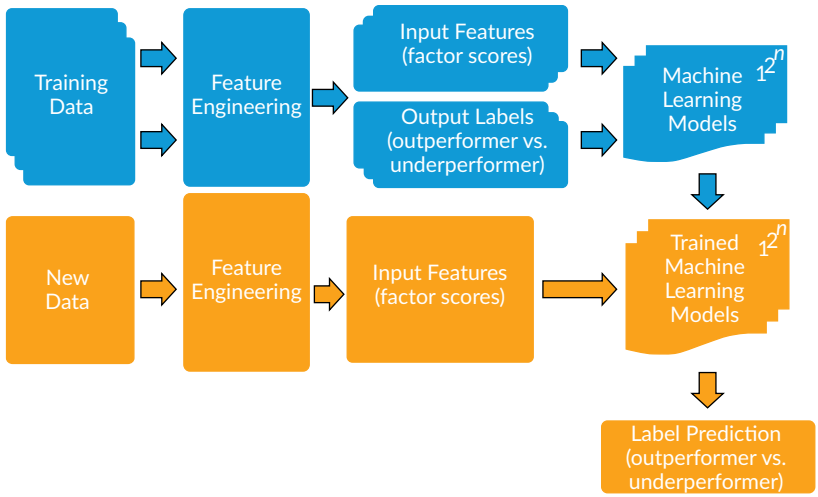
regions: the United States, Japan, Europe, and Asia ex Japan. All factors are percentiles ranked within region and industry buckets for each date. We created risk-adjusted excess returns by dividing each stock's excess return by its past 100-day volatility. We classified stocks as winners or losers by splitting them in half within region/industry buckets based on their risk-adjusted returns.

In the next step, we trained four different algorithms on each of the regional training sets—a bagging estimator that used AdaBoost, a gradient boosted classification and regression tree (GBRT) algorithm, a neural network, and a bagging estimator that used a support vector machine.

A bagging estimator that used AdaBoost as the base learner. AdaBoost uses decision stumps, or trees with a maximum depth of 1, as the base estimators. We ran the algorithm for a total of 50 boosting iterations and with a learning rate of 1. We then used a bagging estimator that combined 20 random AdaBoost forecasts. The algorithm was implemented with the scikit-learn library.

A gradient boosted regression tree algorithm. We used an XGBoost classifier for the GBRT composite forecasts. The learning rate was 0.05, and we used 300 boosting iterations. We also set the maximum depth of the trees to 3 to limit overfitting. In general, we found that low learning rates lead to good results

Figure 2. Example Learning Workflow



Notes: For each month, three training samples were created from historical data: recent, seasonal, and hedge. Next, we applied feature engineering that included standardizing features within region and industry. We also categorized stocks as winners or losers in region/industry buckets. Next, we trained our MLAs on the historical data. When new feature data were received, we applied the same feature-engineering steps that were applied to the training set and generated forecasts with each of our previously trained algorithms.

as long as they are compensated for by a large number of boosting iterations. A large number of boosting iterations does, however, increase runtime.

A neural network. We implemented a multilayer perceptron with the TensorFlow library. We used four layers, including a bottleneck layer, to limit overfitting. Furthermore, we applied a dropout of 20% after the first layer. We used "tanh" activation functions because we found their prediction accuracy with our training set to be consistently better than the generally preferred "ReLU" activation functions.

A bagging estimator using a support vector machine as the base learner. We used a radial basis function (RBF) kernel, and we combined forecasts from 20 support vector machine (SVM) models using the scikit-learn bagging estimator. Calculation of predicted class probability is computationally expensive for SVMs, so instead, we used the decision function output, which tends to be proportional to probabilities.

AdaBoost and SVMs are relatively slow and hard to parallelize. Using these MLAs as base learners in bagging (as opposed to boosting) estimators results in significantly shorter runtimes because they can run on multiple CPUs. For each class of algorithm, we used the same parameters across training windows. Parameters were chosen in the period prior to 2004.

For each of the 12 models (3 training windows × 4 MLAs), we got a probability of outperformance for each stock and month (these are continuous variables). In the final step, we percentile-ranked the 12 predicted class probabilities within each region/industry bucket for each date and averaged them to derive the composite machine learning signals.[6]

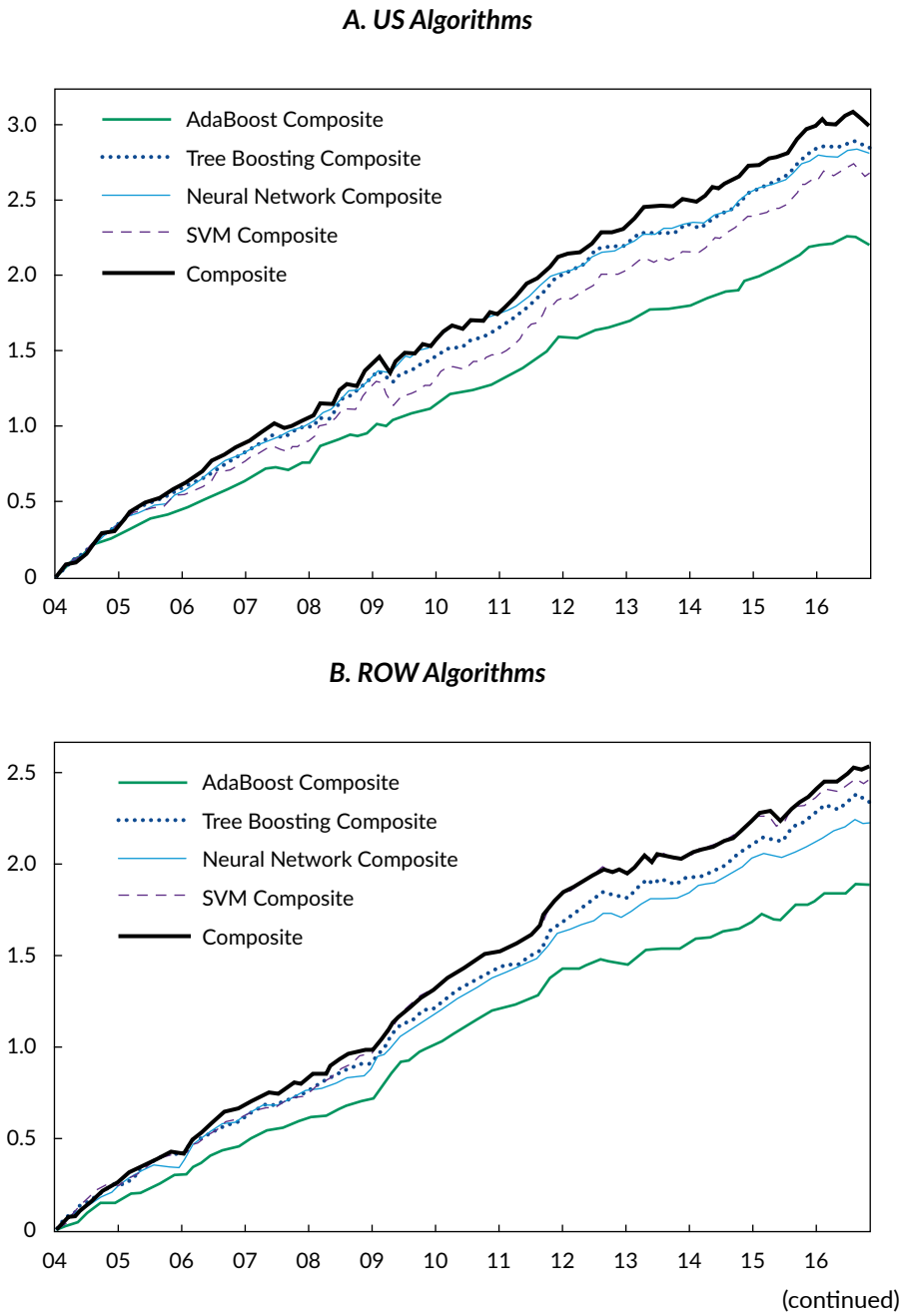
We also analyzed two benchmark models. The first used predictions from an OLS model trained on a 12-month rolling window with the same factors as the other machine learning models and returns that were *z*-scored by date, region, and industry. In the second, we recursively determined the 10 factors with the highest Sharpe ratios up to each point in time in each region by simulating decile spreads for each factor; for each stock, we then averaged the scores of the top 10 factors. In unreported results, we found that equal-weighting all candidate factors performed significantly worse, so we decided to use the more challenging benchmark.

Results. We divided stocks into deciles based on the composite signals. We then calculated decile spreads by taking the difference in return between the top and bottom deciles (i.e., a long–short portfolio). For one variant, we equal-weighted the stocks in each decile; for the other variant, we scaled our positions by each stock's historical 100-day standard deviation.

Figure 3 shows the benefit of combining forecasts in the US zone and the rest of the world (ROW)[7] based on cumulative decile spread returns. In both geographical areas, the heavy line shows the composite decile spread and the thinner lines show decile spreads grouped by algorithm or by training window. In these graphs, we show the risk-weighted variant, but results for the equal-weighted variant are qualitatively similar. Results are strong for all algorithms and training sets, but results for the composite forecasts are even stronger. The benefits of diversifying across training windows is particularly evident.

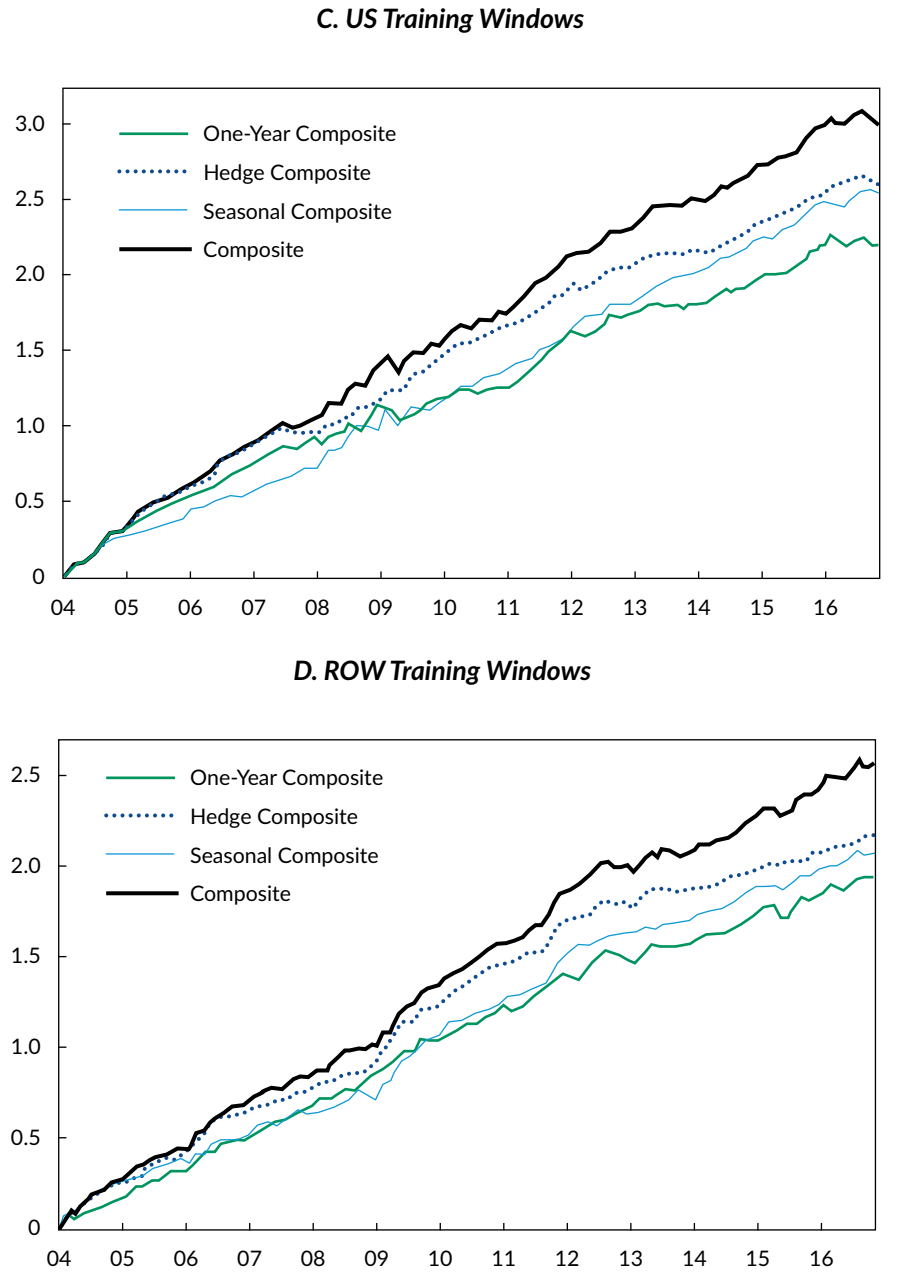
In Figure 3, we used decile spreads to measure the performance of various forecasting approaches. Because of their nonlinear nature, machine learning algorithms do not have interpretable coefficients, but we can frequently tease out which features contribute most to the forecasts. **Figure 4** shows average feature importance (for the 10 most important

Figure 3. Cumulative Performance of Machine Learning Forecasts by Algorithm and Training Window (Risk Weighted), 2004–2016



(continued)

Figure 3. Cumulative Performance of Machine Learning Forecasts by Algorithm and Training Window (Risk Weighted), 2004–2016 (continued)

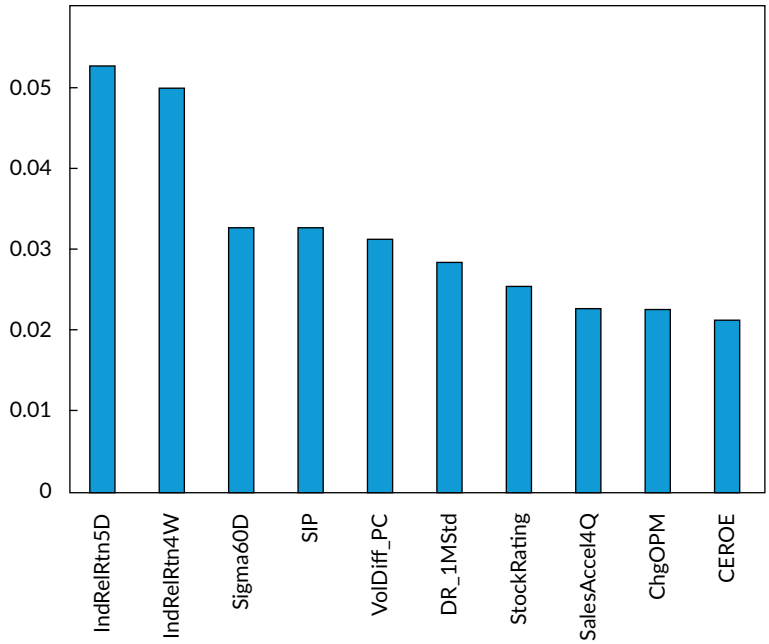


Notes: The figure shows cumulative decile spread returns without compounding for various subsets of forecasts. Stocks in each decile were weighted by the inverse of their 100-day volatility. The algorithm subsets used all training windows for the given algorithm; the training window subsets used all algorithms for the given training window. The composite forecast used all training windows and algorithms.

features) in a gradient boosted classification tree for the US zone, which was assessed by examining how often a feature appeared in a tree and at what level.[8] Because trees are conditional, this metric does not indicate the sign of the relationship. Although it has many limitations, this type of analysis can still provide important insights and also help practitioners decide whether an algorithm seems reasonable.

We report rank ICs (information coefficients) and *t*-statistics for all models and the composite in **Table 2**. The IC measures the Spearman rank correlation between stock rankings and future returns across the full range of forecasts, not just the extreme deciles. As Table 2 shows, the composite outperformed all constituent algorithm/training window combinations, which illustrates the benefit of forecast

Figure 4. Feature Importance for a US Zone Gradient Boosted Classification Tree



Notes: The model was fitted with XGBoost from December 2006 to December 2016. IndRelRtn5D is the industry-relative return over the last five days. IndRelRtn4W is the industry-relative return over the last four weeks. Sigma60D is the standard error of residuals from a regression of a stock's 60 trailing daily returns against corresponding region and sector returns. SIP refers to the number of shares sold short as a percentage of shares outstanding. VolDiff_PC is the difference in implied volatility between at-the-money put and call options. DR_1MStd is the one-month return volatility. StockRating is the consensus analysts' stock rating. SalesAccel4Q is the slope of the regression line between year-over-year sales growth and time. ChgOPM is the most recent quarterly operating profit margin minus that of four quarters ago. CEROE is the trailing 12-month cash flow from operations divided by equity.

combinations. These results are consistent with those presented in Figure 3; that is, the ML composite provided better results than any individual algorithm/training window combination. It also handily outperformed the OLS benchmark and a benchmark that equal-weighted the 10 factors with the highest Sharpe ratios.

Table 3 provides correlations between certain MLA portfolios and the Fama and French (1992) market risk factor (MKT). Interestingly, the equal-weighted US decile spreads have high average negative correlations with the market, implying that the portfolio is often long low-beta names and short high-beta names. As we will discuss later, however, these correlations and exposures vary considerably over time. When we use risk weighting, we get much smaller average correlations with the market factor. Because our forecasts tend to short high-volatility securities, the risk-weighted position sizing leads to a portfolio that is slightly net long in dollar terms, thus counteracting much of the negative average correlation with the market.

Table 4 shows how the average monthly performance of various decile spreads is related to the original three Fama–French factors—MKT, SMB (small minus big), and HML (high book-to-market ratio minus low book-to-market ratio)—and the Carhart (1997) momentum factor (MOM). In Panel A, results are gross of transaction costs; in Panel B, results are net of transaction costs. In Panel A, the excess returns and alphas tend to be larger for the risk-weighted portfolios, both in the US zone and in the ROW. This outcome is not surprising because we trained the MLAs to forecast returns scaled by their standard deviations. The equal-weighted decile spreads show highly significant returns and alphas, but their negative market exposure reduces their raw returns. This aspect is also reflected in the relatively high R^2 for the US zone equal-weighted portfolio, indicating that much of this portfolio's variance is explained by the four risk factors, most notably the market factor.

Table 4 also includes results for a benchmark strategy that employed the same features but used linear regression to forecast stock returns. Results for this

Table 2. Information Coefficients for Models, Composite, and Benchmarks

	US Zone		ROW	
Model	Rank IC	*t*-Statistic	Rank IC	*t*-Statistic
ML composite	6.48%	15.87	6.43%	16.37
12-Month period				
AdaBoost	3.19	9.07	3.54	10.98
SVM	4.61	8.00	5.08	10.35
Tree boosting	4.66	9.59	4.79	11.12
Neural network	3.99	12.94	3.17	11.97
Seasonal				
AdaBoost	3.17	11.53	3.25	11.75
SVM	5.01	9.65	4.79	10.56
Tree boosting	5.00	12.04	4.53	11.89
Neural network	5.20	15.39	4.41	11.92
Hedge				
AdaBoost	3.92	14.06	3.53	13.52
SVM	4.94	13.21	5.23	14.36
Tree boosting	5.00	14.14	4.78	16.16
Neural network	4.58	14.24	4.47	14.61
Top 10 factors benchmark	2.81	6.33	4.49	9.44
OLS benchmark	3.36	5.78	2.71	4.72

Table 3. Correlations between Machine Learning Portfolios and the Market Factor

Portfolio	US Equal Weighted	US Risk Weighted	ROW Equal Weighted	ROW Risk Weighted	MKT
US equal weighted	1.00	0.81	0.32	0.29	–0.57
US risk weighted		1.00	0.28	0.30	–0.20
ROW equal weighted			1.00	0.96	–0.16
ROW risk weighted				1.00	–0.05
MKT					1.00

Note: MKT refers to the Fama–French (1992) market risk factor.

OLS benchmark strategy are positive, but average returns and alphas are considerably larger for the machine learning approaches. The gap increases for the *t*-statistics, which suggests that the MLA strategies are more attractive on a risk-adjusted basis. We also examined a strategy that found the 10 factors with the highest Sharpe ratios up to each point in time and then equal-weighted the factors. Results for this naive strategy tend to be better than results for the OLS benchmark, but the machine learning composites are significantly better both in the US zone and in the ROW.

Although we would not recommend interpreting these results as describing a fully fledged trading strategy, we explored whether the conclusions would be significantly affected by the inclusion of trading costs. Results net of transaction costs are shown in

Table 4. Regression Results vs. Fama–French–Carhart Factors (*t*-statistics in parentheses)

A. Regression of MLA decile spreads on Fama–French–Carhart factors

	US Zone				ROW			
	Equal Weighted	Risk Weighted	Top 10 Benchmark	OLS Benchmark	Equal Weighted	Risk Weighted	Top 10 Benchmark	OLS Benchmark
Excess return	1.60	1.95	1.23	1.10	1.50	1.64	1.06	0.79
	(6.04)	(11.32)	(6.32)	(3.56)	(9.32)	(11.61)	(5.91)	(5.23)
Alpha	1.84	2.01	1.17	1.03	1.53	1.65	1.11	0.85
	(9.48)	(11.93)	(7.05)	(3.79)	(9.48)	(11.53)	(6.19)	(6.18)
MKT	–0.38	–0.09	0.07	0.02	–0.04	0.01	–0.07	–0.12
	(–6.93)	(–1.91)	(1.53)	(0.28)	(–0.80)	(0.14)	(–1.44)	(–3.03)
SMB	–0.23	–0.07	0.03	–0.08	–0.07	–0.05	–0.1	0.02
	(–2.44)	(–0.84)	(0.38)	(–0.63)	(–0.93)	(–0.67)	(–1.22)	(0.27)
HML	–0.06	0.00	0.37	0.05	–0.09	–0.11	0.1	0.01
	(–0.67)	(0.05)	(4.97)	(0.34)	(–1.25)	(–1.70)	(1.18)	(0.19)
MOM	0.18	0.08	–0.11	0.22	–0.06	–0.04	0.05	0.14
	(3.76)	(1.90)	(2.71)	(3.43)	(–1.44)	(–1.15)	(1.16)	(4.10)
Adjusted R^2	0.47	0.07	0.29	0.32	0.01	0.00	0.03	0.20
No. of observations	154	154	154	154	154	154	154	154

(continued)

Table 4. Regression Results vs. Fama–French–Carhart Factors (*t*-statistics in parentheses) (continued)

B. Regression of MLA spreads with transaction costs on Fama–French–Carhart factors

	US Zone				ROW			
	Equal Weighted	Risk Weighted	Top 10 Benchmark	OLS Benchmark	Equal Weighted	Risk Weighted	Top 10 Benchmark	OLS Benchmark
Excess return	1.35	1.67	1.00	0.85	1.24	1.38	0.93	0.61
	(5.15)	(9.76)	(5.16)	(2.77)	(7.74)	(9.83)	(5.17)	(4.04)
Alpha	1.61	1.74	0.94	0.77	1.27	1.39	0.98	0.67
	(8.34)	(10.37)	(5.69)	(2.90)	(7.90)	(9.74)	(5.45)	(4.89)
MKT	−0.38	−0.09	0.07	0.02	−0.04	0.01	−0.07	−0.12
	(−6.95)	(−1.99)	(1.59)	(0.21)	(−0.86)	(0.16)	(−1.41)	(−3.06)
SMB	−0.23	−0.06	0.03	−0.08	−0.06	−0.04	−0.1	0.02
	(−2.51)	(−0.75)	(0.36)	(−0.61)	(−0.85)	(−0.57)	(−1.23)	(0.29)
HML	−0.06	0.00	0.37	0.05	−0.09	−0.11	0.1	0.01
	(−0.66)	(0.02)	(4.99)	(0.38)	(−1.26)	(−1.70)	(1.2)	(0.21)
MOM	0.18	0.08	−0.11	0.22	−0.05	−0.04	0.05	0.14
	(3.76)	(1.97)	(−2.70)	(3.45)	(−1.37)	(−1.07)	(1.19)	(4.15)
Adjusted R^2	0.48	0.07	0.29	0.32	0.01	0.00	0.03	0.20
No. of observations	154	154	154	154	154	154	154	154

Notes: The table shows average monthly excess returns and alphas. The label "US Zone" includes US stocks; the label "ROW" (rest of the world) includes all other regions (i.e., Europe, Japan, and Asia ex Japan). In Panel B, transaction cost estimates are 15 bps per side.

Panel B of Table 4. We found that alphas remained highly significant.

Interestingly, the machine learning portfolios generally loaded negatively on the value (HML) and small size (SMB) factors, but only the US zone equal-weighted loading on SMB is significant. This outcome indicates that the positive results are not driven by common risk factors. Loadings on momentum (MOM) are positive for the US zone and negative for the ROW, but again, only the US zone equal-weighted result is significant. These outcomes confirm that portfolio construction can have a significant impact on portfolio risk.

Because alphas tend to be no less significant than excess returns, the four-factor model probably does not explain the returns to the machine learning strategy. Rather, the MLA composite probably extracts information from other factors or exploits time-varying relationships between factors and returns that are not captured by a linear risk model.

Equity market-neutral funds had relatively low returns during the sample period (1994–2016). For example, the Hedge Fund Research equity market-neutral index returned only 0.24% a month from 2004 through 2016. During the same period, all the machine learning decile spreads returned more than 1% a month *after* estimated transaction costs.

In **Table 5**, we report Fama and MacBeth (1973) multiple regression coefficients for aggregate machine learning forecasts based on both raw monthly excess returns and monthly excess returns scaled inversely to volatility (risk weighted). We controlled for a battery of popular quantitative factors and conducted the analysis separately for the US zone and the rest of the world. All variables were standardized to make coefficients comparable. Table 5 shows that the machine learning composite forecasts are significantly related to returns across all specifications, even after controlling for many popular quantitative factors. Perhaps surprisingly, a few of the control variables remain modestly significant. This result was a bit unexpected because all of these control factors are included in our factor library. If the algorithms efficiently use the information embedded in these factors, we would expect the control factors to offer little incremental value. The algorithms consider only point-in-time information, however, and identifying successful factors *ex post* is much easier than doing so *ex ante*.

In **Table 6**, we report returns and four-factor alphas for the long and the short sides of the MLA portfolio. Similar to Gu et al. (2018), we found that most of the outperformance came from the long side. This finding is not surprising because equity markets performed very well in the period. Alphas are significant for both the long and short portfolios in both the US zone and the ROW. Looking at the long and short legs independently, however, does not tell the whole story. The *t*-statistics of the alphas are much larger for the long–short portfolios presented in Table 4 than for the individual long and short legs presented in Table 6. Because we constrained the machine learning algorithms to forecast risk- and industry-standardized returns, the decile spreads eliminate most industry risk and idiosyncratic volatility and increase risk-adjusted returns. This outcome illustrates the potential benefits of feature engineering.

A potential advantage of machine learning forecasts is that algorithms can dynamically learn changing relationships between factors and returns. **Figure 5** shows the time series of monthly cross-sectional correlations between the US zone machine learning composite forecasts and the Fama–French–Carhart factors. Clearly, the average correlations in Table 3 do not tell the whole story; there is significant time variation in all of these correlations. For example, despite a positive average correlation, we see that the machine learning forecasts were negatively correlated with momentum in three periods: (1) the beginning of the sample, (2) between 2009 and 2011, and (3) intermittently between 2013 and 2015. These periods of negative correlation tended to coincide with or follow periods of significant underperformance for momentum. For example, following the momentum crash documented in Daniel and Moskowitz (2016), exposure to momentum became negative, with a cross-sectional correlation of around −0.3 for several months in 2010. As noted previously, the average correlations with size and beta tended to be negative, but they varied considerably over time and were occasionally positive.

Although determining precisely how much of the machine learning strategy's alpha comes from factor timing is difficult, these factor exposures are clearly much more variable than the exposures found in typical linear factor models.

Table 5. Fama–MacBeth Regressions (*t*-statistics in parentheses)

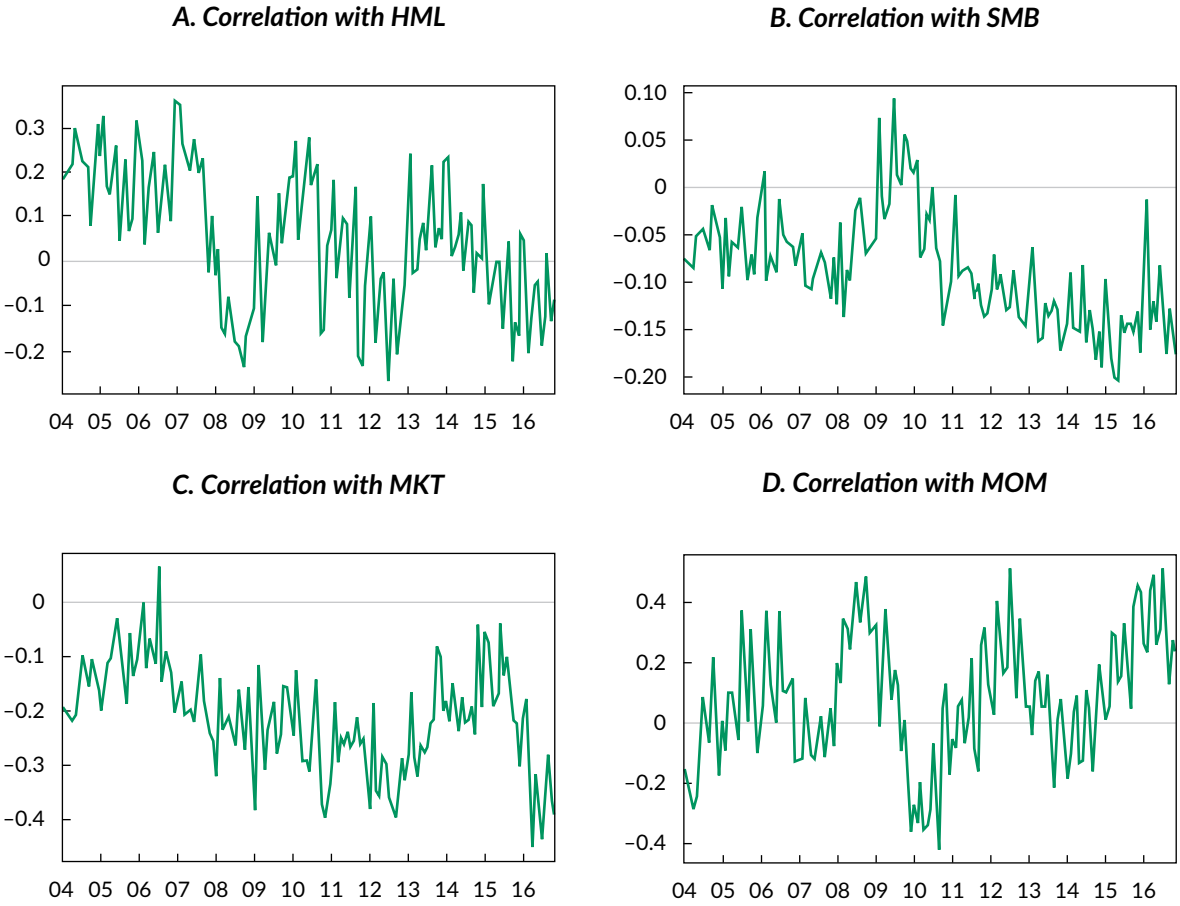
	US Zone		ROW	
Forecast/Characteristic	Equal Weighted	Risk Weighted	Equal Weighted	Risk Weighted
ML composite	1.71	1.77	1.00	1.12
	(10.04)	(12.05)	(9.22)	(10.44)
Earnings revision	0.21	0.15	0.33	0.32
	(2.17)	(1.57)	(4.46)	(4.25)
Dividend yield	0.11	0.09	0.16	0.19
	(0.94)	(0.91)	(1.38)	(1.71)
Return on equity	0.28	0.10	−0.00	−0.04
	(1.81)	(0.80)	(−0.03)	(−0.48)
Book-to-price ratio	0.29	0.19	0.44	0.37
	(1.47)	(1.32)	(3.17)	(2.78)
Momentum	−0.44	−0.32	0.27	0.33
	(−1.21)	(−1.29)	(1.08)	(1.45)
Growth in earnings per share	0.12	0.05	−0.02	−0.04
	(1.08)	(0.64)	(−0.35)	(−0.65)
1-Month reversal	0.21	0.12	0.16	0.12
	(0.87)	(0.65)	(0.83)	(0.67)
Low volatility	−0.10	0.16	−0.13	0.05
	(−0.30)	(0.80)	(−0.58)	(0.28)
Earnings yield	−0.38	−0.17	0.09	0.25
	(−1.94)	(−1.34)	(0.86)	(2.52)
Accounting accruals	0.21	0.15	0.29	0.28
	(2.24)	(1.93)	(3.77)	(3.39)
Intercept	0.06	0.01	−0.31	−0.35
	(0.09)	(0.02)	(−0.66)	(−0.80)

Notes: Monthly returns were regressed on lagged company characteristics and ML forecasts. For the equal-weighted specifications, the dependent variable is excess returns; for the risk-weighted specifications, the dependent variable is excess returns scaled by their 100-day trailing standard deviation.

Table 6. Long and Short Portfolios (*t*-statistics in parentheses)

	US Zone		ROW	
Measure	Long	Short	Long	Short
Excess return	1.90	−0.03	1.50	−0.13
	(3.54)	(−0.05)	(4.18)	(−0.33)
Fama–French–Carhart four-factor alpha	1.13	−0.96	0.95	−0.69
	(4.43)	(−3.28)	(4.72)	(−2.97)

Figure 5. Cross-Sectional Correlations of Fama–French–Carhart Factors with Composite US Zone Machine Learning Forecasts



Conclusion

We have discussed how practitioners can use machine learning algorithms for stock selection while avoiding the primary problem with these techniques—namely, overfitting. Low signal-to-noise ratios in security selection mean that overfitting is always a risk, especially with such techniques as MLAs, which impose little structure to the analysis. However, because they do not require structure, MLAs can uncover complex nonlinear patterns that are hard to tease out with traditional statistical techniques such as OLS. They also tend to work better than OLS when numerous collinear factors must be considered.

We discussed two primary ways to reduce the risk of overfitting—feature engineering and forecast combinations. Feature engineering can increase the signal-to-noise ratio by correctly framing the problem and transforming the data to produce cleaner signals. Forecast combinations reduce noise by focusing on relationships that are robust to different forecasting techniques (MLAs) and training windows. A successful machine learning application requires considerable domain expertise to address these issues. MLAs will not replace human experts anytime soon (at least not in investing).

In the final section, we presented a case study based on some of the proposed techniques. We demonstrated that, properly applied, MLAs can use a wide variety of company characteristics to forecast stock returns without overfitting. With sensible feature engineering and forecast combinations, MLAs can produce results that dramatically exceed those derived from simple linear techniques such as OLS. These MLA results are robust to various risk adjustments and work well both in the US market and in other developed markets. Although accurately assessing which signals are driving the results can be difficult, we demonstrated that traditional factor exposures vary considerably over time,

implying that factor timing contributes to these positive results.

The main contributions of this article are (1) a discussion of feature engineering and some of the issues that practitioners face when using machine learning models for stock selection and (2) a demonstration of the benefits of forecast combinations when using these techniques. In particular, we highlighted the benefit of combining forecasts from various algorithms and training windows and showed that the MLAs can produce results that are superior to simple linear models.

Appendix A. Glossary

Activation functions in neural networks determine the output of nodes given the inputs. Nonlinear activation functions allow neural networks to learn nonlinear patterns. Popular activation functions include Rectified Linear Units (ReLU), tanh, and sigmoid. Sigmoid activations are often used in the output layer of binary classification problems because they can map inputs to probabilities between 0 and 1.

Artificial neural networks and deep learning are algorithms loosely modeled on the human brain. Neural units are organized in layers and connected to each other, making possible the learning of many interactive relationships. Artificial neural networks can be easy to overfit, however, and finding the correct architecture for a given problem is often difficult. Some recent innovations related to deep learning, such as dropout, make it possible to apply deeper architectures without overfitting.

Bagging (bootstrap aggregating) algorithms combine forecasts generated by base algorithms on randomly sampled learning sets. Random forests are an example of the application of bagging to classification and regression tree (CART) models. In bagging (in contrast to boosting) algorithms, forecasts of the base algorithms are equally weighted. Bagging tends to increase the stability of algorithms and helps prevent overfitting.

Boosting algorithms combine forecasts from many base algorithms, such as CARTs. In contrast to random forests, boosting gives more weight to more successful models. Boosting algorithms have the potential to learn more efficiently from data than random forests but require greater care when tuning parameters because they are more prone to overfitting. They also take longer to run than bagging algorithms because they require sequential processing.

Bottleneck layers in a neural network have fewer neurons than the layers above and below. Having a bottleneck layer encourages the network to reduce the dimensionality of features.

Classification and regression tree (CART) models form the basis of many machine learning algorithms. CARTs are prone to overfitting, however, and are rarely competitive on their own. CARTs can detect hierarchical (nonlinear) relationships.

Dropout in neural networks is a technique to limit overfitting. Similar to bagging, dropout effectively is a model-averaging technique. When training a neural network, the algorithm drops out elements of layers, producing models that often generalize better on unseen data.

Gradient boosted trees use decision trees as base learners. Subsequent trees are trained on residuals from earlier iterations. The learning rate and number of boosting iterations are key parameters that influence how aggressively the model can learn and also overfit. The depth of the base learners also is an important parameter.

A **multilayer perceptron (MLP)** is a feed-forward neural network consisting of three or more layers. MLPs are some of the earliest and best-understood neural networks.

The **radial basis function (RBF) kernel** is a commonly used kernel, or similarity function, in support vector machines. The RBF kernel is especially well suited to problems that are potentially nonlinear.

Random forests combine many CART models (or trees) by averaging their forecasts. This approach can often diversify away errors resulting from overfitting. Therefore, random forests usually have more signal and less noise than the individual trees. Random forests are quite robust to overfitting and often work well out of sample.

Support vector machines (SVMs) are effective classifiers in high-dimensional spaces and can use either linear or nonlinear kernels. While often effective, SVMs can run slowly with large datasets.

Editor's Note

Submitted 19 July 2018

Accepted 30 January 2019 by Stephen J. Brown

Notes

1. "Bagging" is an abbreviation for "bootstrap aggregating," or averaging forecasts from different training sets. "Boosting" is the process of reweighting observations to put more weight on misclassifications from prior forecasting rounds.
2. An alternative approach is to use a robust objective function, such as the pairwise rank correlation between returns and forecasts in a regression setting.
3. https://en.wikipedia.org/wiki/Outline_of_machine_learning.
4. We refer interested readers to chapter 7 in López de Prado (2018) for an in-depth treatment of cross-validation for financial data.
5. Machine learning algorithms are well known for their ability to tease signals from big data—for instance, detecting sentiment in text or predicting future sales from social media posts. Although these applications are certainly promising, they are not the focus of this article. Our goal is to show how MLAs can be more effective than traditional quantitative techniques even when using widely known quant signals to forecast security returns.
6. Practitioners could also use a machine learning model to aggregate the information of the individual signals.
7. Results for individual regions are available on request.
8. Gu et al. (2018) found that price trend, volatility, and liquidity are by far the most important features. Our analysis suggests that these categories are important, but we also found that the percentage of shares sold short, the difference between put and call implied volatilities, and characteristics derived from financial statement information are among the 10 most important features.

References

Alberg, John, and Zachary Lipton. 2017. "Improving Factor-Based Quantitative Investing by Forecasting Company Fundamentals." Working paper, Cornell University. https://arxiv.org/abs/1711.04837v2.

Asness, Clifford S. 2016. "The Siren Song of Factor Timing aka 'Smart Beta Timing' aka 'Style Timing.'" *Journal of Portfolio Management* 42: 1–6.

Asness, Clifford S., R. Burt Porter, and Ross L. Stevens. 2000. "Predicting Stock Returns Using Industry-Relative Firm Characteristics." Working paper, AQR Capital Management.

Batres-Estrada, Gilberto. 2015. "Deep Learning for Multivariate Financial Time Series." Master's thesis, KTH Royal Institute of Technology. https://www.math.kth.se/matstat/seminarier/reports/M-exjobb15/150612a.pdf.

Carhart, Mark. 1997. "On Persistence in Mutual Fund Performance." *Journal of Finance* 52 (1): 57–82.

Clemen, Robert T. 1989. "Combining Forecasts: A Review and Annotated Bibliography." *International Journal of Forecasting* 5 (4): 559–83.

Daniel, K., and T.J. Moskowitz. 2016. "Momentum Crashes." *Journal of Financial Economics* 122 (2): 221–47.

Fama, Eugene F., and Kenneth R. French. 1992. "The Cross-Section of Expected Stock Returns." *Journal of Finance* 47 (2): 427–65.

——. 2017. "International Tests of a Five-Factor Asset Pricing Model." *Journal of Financial Economics* 123 (3): 441–63.

Fama, Eugene F., and J.D. MacBeth. 1973. "Risk, Return, and Equilibrium: Empirical Tests." *Journal of Political Economy* 81 (3): 1–31.

Gu, Shihao, Bryan T. Kelly, and Dacheng Xiu. 2018. "Empirical Asset Pricing via Machine Learning." NBER Working Paper 25398 (December).

Heaton, J.B., N.G. Polson, and J.H. Witte. 2017. "Deep Learning in Finance: Deep Portfolios." *Applied Stochastic Models in Business and Industry* 33 (1): 3–12.

López de Prado, M. 2018. *Advances in Financial Machine Learning.* Hoboken, NJ: John Wiley & Sons.

Makridakis, Spyros, and Michèle Hibon. 2000. "The M3-Competition: Results, Conclusions and Implications." *International Journal of Forecasting* 16 (4): 451–76.

Miller, K.L, Hong Li, Tiffany G. Zhou, and Daniel Giamouridis. 2015. "A Risk-Oriented Model for Factor Timing Decisions." *Journal of Portfolio Management* 41 (3): 46–58.

Miller, K.L., Chee Ooi, Hong Li, and D. Giamouridis. 2013. "Size Rotation in the U.S. Equity Market." *Journal of Portfolio Management* 39 (2): 116–27.

Moritz, B., and T. Zimmerman. 2016. "Tree-Based Conditional Portfolio Sorts: The Relation between Past and Future Stock Returns." Working paper (March).

Morozov, A., J. Wang, and L. Borda. 2012. "Barra Global Equity Model (GEM3)." MSCI. https://www.msci.com/documents/10199/242721/Barra_Global_Equity_Model_GEM3.pdf.

Schapire, R. 1990. "The Strength of Weak Learnability." *Machine Learning* 5 (2): 197–227.

Srivastava, N., and G. Hinton. 2014. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting." *Journal of Machine Learning Research* 15: 1929–58.

Takeuchi, L., and Y.Y.A. Lee. 2013. "Applying Deep Learning to Enhance Momentum Trading Strategies in Stocks." Working paper, Stanford University. http://cs229.stanford.edu/proj2013/TakeuchiLee-ApplyingDeepLearningToEnhanceMomentumTradingStrategiesInStocks.pdf.

Timmermann, A. 2006. "Forecast Combinations." In *Handbook of Economic Forecasting*, edited by Graham Elliott, Clive W.J. Granger, and Allan Timmerman, 1: 135–96. Amsterdam: Elsevier.

Wang, S., and Y. Luo. 2012. "Signal Processing: The Rise of the Machines." Deutsche Bank Quantitative Strategy (5 June).

——. 2014. "Signal Processing: The Rise of the Machines III." Deutsche Bank Quantitative Strategy.