Machine Learning for Stock Selection

Financial Analysts Journal, Forthcoming

KEYWAN C. RASEKHSCHAFFE

Gresham Investment Management in New York, NY. key@greshamllc.com

ROBERT C. JONES

System Two Advisors in Summit, NJ. bob.jones@s2adv.com

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 non-linear relationships, overfitting poses a major challenge when trying to extract signals from noisy historical data. In this article, we describe some of the basic concepts surrounding machine leaning 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.

* We are thankful for comments and suggestions from Stephen Brown (Editor), Daniel Giamouridis (Co-Editor), and two anonymous referees.

Many quantitative factor models have struggled since the financial crisis in 2008 and many traditional factors have ceased to be profitable. As a result, some market participants are looking beyond 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. However, dynamic models and ad-hoc factor-timing approaches also face some valid criticisms (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. This may be because financial data is inherently noisy, factors can be multi-collinear, and relationships between factors and returns can be variable, non-linear and/or contextual. These features can 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) can provide a better approach. 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 have enabled recent 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 like 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., Siri or

Alexa), image recognition (e.g., self-driving cars) and recommendation engines (e.g., Amazon). Deep learning algorithms now exceed human accuracy for many image classification tasks. A machine learning algorithm called Deep Blue first beat the best human chess master, Garry Kasparov, in 1997. It used brute-force computational speed to evaluate thousands of possible moves and counter moves. More recently a deep learning neural network called AlphaZero used pattern recognition techniques to become the world chess champion. Unlike Deep Blue, which was pre-programmed to evaluate 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 Table 1). Most of these algorithms have two important properties:

1. They can uncover complex patterns and hidden relationships, including non-linear and contextual relationships that are often difficult or impossible to detect with linear analysis.
2. They are often more effective than linear regression in the presence of multi-collinearity.

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

Several articles study the benefits of non-linear models for timing factor returns. Miller et al. (2013) and Miller et al. (2015) find that classification trees can be more effective than linear regressions when predicting factor returns. They also present evidence that combining linear and non-linear models can be even more effective. Further, they demonstrate that incorporating these factor forecasts into cross-sectional models can outperform static factor models. We reach similar conclusions in this article, but we follow a different approach. Instead of explicitly forecasting returns of univariate long/short factor portfolios, we use cross sectional factor scores (characteristics) to predict the cross-section of returns.

Gu, Kelly, and Xiu (2018) examine the efficacy of machine learning techniques in the context of asset pricing. The authors forecast individual stock returns with a large set of firm characteristics and macro variables. Since they use total returns rather than market excess returns as the dependent variable, they jointly forecast the cross-section of expected returns and the equity premium. They examine how different machine learning methods perform and find that non-linear estimators result in significantly improved accuracy when compared to OLS regressions.

They attribute the improvement to the ability of machine learning models to uncover non-linear patterns and to their robustness to multi-collinear predictors. While our conclusions are similar, we focus solely on the cross-section of returns independent of the equity risk premium. Accordingly, we use only individual equity characteristics and exclude macro variables. We believe this approach reduces noise and the risk of overfitting. In agreement with Gu, Kelly, and Xiu (2018) we find that many machine learning algorithms can outperform linear regression, but rather than concentrating on the performance of individual algorithms, we highlight the benefits of combing forecasts generated from different algorithms and training windows. We find that forecast combinations outperform constituents both in the US and in other regions.

The main contributions of this article are: (1) to discuss feature engineering and some of the issues that practitioners face when using machine learning models for stock selection; and (2) to demonstrate the benefits of forecast combinations when using these techniques. In particular we highlight the diversification benefit of combining forecasts from different algorithms and training windows.

THE PERILS OF OVERFITTING

Overfitting occurs when a model picks up noise instead of signals. Overfit models have very good in-sample performance, but little predictability on unseen data. Although machine learning techniques can uncover subtle patterns in past data, overfitting presents a major challenge. When training an algorithm, it is important to find patterns in the data that also generalize out of sample. Relationships between factors and returns are frequently noisy, and many potential factors exist, increasing the dimensionality of the problem. In contrast, many other machine learning applications, such as image recognition, have much higher signal-to-noise ratios. For example, some image classification tasks have error rates below 1% (e.g., classifying dogs vs. cats).

Because of the low signal-to-noise ratios in forecasting stock returns, it is particularly important to avoid overfitting. Figure 1 shows in-sample and out-of-sample error rates for a gradient boosted regression tree classifier using simulated noisy data. The X-axis represents the number of boosting iterations (more iterations allow the algorithm to better fit past data), and the Y-axis shows error. The dark line is in-sample; the light line shows performance on a hold-out sample. The in-sample error is always lower than the out-of-sample error. As we increase the number of boosting iterations, the errors always decline in-sample and become negligible after around 400 boosting iterations. In sharp contrast, the error rate in the hold-out sample first decreases, but then increases after approximately 50 boosting iterations. This is where the algorithm begins to overfit the data.

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

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

(1) 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

[1] "Bagging" is an abbreviation for "bootstrap aggregation," or averaging forecasts from different training sets. "Boosting" is the process of re-weighting observations to put more weight on misclassifications from prior forecasting rounds.

weak learners and then combine these forecasts to form a stronger learner. Dropout (see Table 1 and Srivastava and Hinton (2014)), a related tool 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 sub-sets 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. Clemen (1989) summarizes the empirical evidence on forecast combinations as early as 1989: "The results have been virtually unanimous: combining multiple forecasts leads to increased forecast accuracy.... in many cases one can make dramatic performance improvements by simply averaging the forecasts" (p 137). Makridakis and Hibon (2000), in an analysis of a competition to forecast 3003 different time series, show that forecast combinations usually outperform even the best constituent forecasts. Timmermann (2006) provides 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 multi-factor models already contain multiple forecasts, since 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.

There are several dimensions along which we might achieve greater forecast diversity:

- **Combining forecasts from different classes of algorithms:** Many machine learning algorithms, and especially ensemble algorithms (e.g., random forests), already employ forecast combinations to yield better results. 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. Widows 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 longer horizons, while technical factors tend to be more predictive for shorter horizons.

(2) Feature Engineering

Feature engineering uses domain knowledge to structure a problem so that it is more amenable to machine learning solutions. It requires considerable expertise, and can be difficult and time-consuming, but it is also 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, this can cover decisions such as: 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.

Below we will briefly discuss some of these issues. The goal is to provide an overview rather than a comprehensive discussion. The possible variations across these decisions are as far-ranging as the imaginations and expertise of financial modelers. Ultimately, however, the quality of these decisions will determine the success of the effort.

What are we forecasting? It is often best to forecast discrete variables with machine learning algorithms (MLAs) to limit the influence of outliers. Rather than predicting returns, then, as we would with linear regression, MLAs usually predict categories – i.e., outperformer versus underperformer – which tend to be less noisy. Users may also want to use a third category, such as market performer, or even more categories to reflect different 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.

A second decision involves how to define these categories. If we are interested in the cross-section of returns we would define categories by ranking stocks into outperformers and underperformers for each date in the training set. We may also define these categories within sectors or industries in order to reduce noise. It is frequently advisable to standardize the factors (i.e. characteristics) in a similar fashion. Most investors want to outperform net of risk, so a

natural approach is to define performance categories using risk-adjusted returns. These might include simple volatility-adjusted returns, or alphas from an appropriate risk model, such as the CAPM, 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 produce better 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 more suited to low-capacity, high-turnover strategies, while long horizons are more suited to high-capacity, lower-turnover strategies. Shorter horizons also mean more training periods, which is very 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 different machine learning methods,[2] and the list is growing all the time. Machine learning is a rapidly evolving field; discussing the pros and cons of these many different algorithms is well beyond the scope of this article. More generally, however, we want our final forecast to incorporate a variety of different algorithms, using a variety of different techniques. It is impossible to know the exact relationship between returns and features ex-ante. Combining forecasts from different classes of algorithms provides insurance against misspecification. This is particularly important when dealing with

[2] https://en.wikipedia.org/wiki/Outline_of_machine_learning

financial data, where the signal-to-noise ratio is low, and it is difficult to empirically verify relationships with a high degree of certainty.

Ensemble methods have shown promise on financial data and in many other fields. They combine weak learners into a strong learner by either equal weighting forecasts (bootstrap aggregation, or bagging) or accuracy weighting forecasts (boosting) into 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 vs. 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) on random subsets of the training set. Each of the weak learners is overfit, but errors due to overfitting tend to be reduced when combining forecasts of weak learners into 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. However, if we allow the boosting algorithm to overweight successful weak learners too aggressively, 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 perform better out-of-sample for stock selection. Most boosting algorithms also take longer to train since they must be run sequentially, while bagging algorithms can be run in parallel.

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

More generally, modelers should focus on algorithms that have shown prior success with noisy data, and where the benefits and pitfalls are well known. As a practical matter, it's also a good idea to use algorithms that are available in software libraries that have been tested across a variety of applications. However, it is unlikely that an off-the-shelf algorithm is appropriate without further parameter tuning. Because the signal-to-noise ratio tends to be very low for stock selection, it is often necessary to parameterize algorithms in a way that severely limits the algorithm's potential to over-fit. As with all investment strategies, it is imperative not to optimize models in-sample. We refer interested readers to Chapter 7 in de Prado (2018) for an in-depth treatment of cross-validation for financial data.

Which training windows should we use? As a general rule, we want to train machine learning algorithms using data that is likely to be representative of the expected future environment. For example, we may want to use training sets that are: recent in time; exhibit similar macro-economic conditions (e.g., valuation levels, liquidity conditions, or growth dynamics); or occurred 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 algos using the largest, longest

and broadest data set possible to capture a variety of different environments. Such an approach, however, will require longer runtimes 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 algos 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 versus Japanese stocks, but not for US tech stocks versus 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 runtime, and to limit overfitting, practitioners should only feed algorithms data that are likely to be related to future stocks returns. These would include factors related to future economic success (fundamental factors) and factors related to future supply and demand (technical factors). Because machine-learning algorithms are usually quite good at handling collinear data, practitioners can certainly include many similar factors if they are uncertain as to which ones are most relevant, although piling on too may 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 stock selection, not picking 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.

A note on big data

Machine learning algorithms are well-known for their ability to tease signals from big data, for instance detecting sentiment from text, or predicting future sales from social media posts. Although these are certainly promising applications of machine-learning algorithms, they are not the focus of this paper. Instead, 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.

AN EXAMPLE

In this section, we describe an approach to stock selection using some of the techniques discussed above 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. There is plenty of room for practitioners to apply their own expertise and potentially achieve even better results.

Data

Table 2 provides summary statistics. Our sample includes small, medium and large cap stocks, with an average of 5907 stocks per month, covering 22 developed markets. Our factor library includes 194 factors (i.e. firm characteristics) which are assembled by IHS Markit from diverse sources. We include 21 deep value factors, 18 relative value factors, 10 factors focusing 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 above the US Treasury bill rate and come from Barra, and all returns are in US Dollars. Our sample runs from 1994 to 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 different horizons may be beneficial, but we focus on a monthly horizon to be consistent with typical factor research. We conservatively allow two days between model estimation and trading to account for run time and parameter tuning.

Feature engineering

Figure 2 outlines the general workflow executed each month in a walk-forward framework.

We start by defining three different training sets as follows:

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

It is possible for a single month to appear in more than one training set. For instance, the same month last year will appear in both the recent and seasonal training sets.

We develop these training sets, and train our algos independently, for four separate regions: The US, Japan, Europe, and Asia ex Japan.

All factors are percentile ranked within region and industry buckets for each date.

We create risk-adjust excess returns by dividing each stock's excess return by its past 100-day volatility. We classify 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 train four different algorithms on each of the regional training sets:

A bagging estimator using AdaBoost as the base learner: Adaboost uses decision stumps, or trees with a maximum depth of 1, as the base estimators. We run the algorithm for a total of 50 boosting iterations and with a learning rate of 1. Further, we use a bagging estimator that combines forecasts from 20 random Adaboost forecasts. The algorithm is 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 is 0.05 and we use 300 boosting iterations. We also limit the maximum depth of the trees to 3 to limit overfitting. In general, we find that low learning rates lead to good results as long as they are compensated by a large number of boosting iterations. However, the number of boosting iterations increases runtime.

A neural network: We implemented a multi-layer perceptron with the Tensorflow library. We use four layers including a bottleneck layer to limit overfitting. Further, we apply dropout of 20% after the first layer. We use 'tanh' activation functions because we found prediction accuracy to be consistently better than the generally preferred 'ReLU' activation functions on our training set.

A bagging estimator using a support vector machine as the base learner. We use a radial bias function (RBF) kernel and we combine forecasts from 20 SVM models using the Scikit Learn bagging estimator. Calculation of predicted class probability is very computationally expensive for Support Vector Machines and we utilize the decision function output instead, which tends to be proportional to probabilities.

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

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

We also provide results for two alternative quantitative strategies**.** In the first, we use predictions from an OLS model that includes all of the factors that appear in any of the MLAs and is estimated over the prior 12 months. In the second, we recursively determine the ten factors with the highest Sharpe ratios up to each point in time and then average the scores of these top-ten factors. In unreported results, we found that equal-weighting all candidate factors performs significantly worse and therefore decided to use these more challenging benchmarks. Both benchmarks also include only a one-day lag for model calculations, versus a two-day lag for the MLAs.

Results

We divide stocks into deciles based on these composite signals. We then calculate decile spreads by taking the difference in return between the top and bottom deciles (i.e. a long/short portfolio). For one variant (1/n), we equal-weight the stocks in each decile; for the other variant (1/vol), we scale positions inversely to each stock's historical prior 100-day standard deviation.

[3] Practitioner could also use an ML model to aggregate the information of the individual signals

Figure 3 shows the benefit of combining forecasts. In all graphs, the thicker line shows the composite decile spreads, while the thinner lines show decile spreads grouped by algos or by training windows. In these graphs, we show the 1/vol variant, but results for the 1/n variant are qualitatively similar. Results are strong for all algos and training sets, but results for the composite forecasts are even stronger. The benefits of diversifying across training windows is particularly evident.

We report rank ICs and t-statistics for all sub models and the composite in Appendix B. The composite outperforms all constituent algo/training window combinations, which further illustrates the benefit of forecast combinations. Looking at individual components, the neural network algorithm and the hedge training set generally have the highest ICs.

Table 3 shows how the average monthly performance of various decile spreads is correlated to the original three Fama and French (1992) factors (MKT, SMB and HML) plus the Carhart (1997) momentum factor (MOM). All results are gross of t-costs except for Panel C. Panel A shows correlations between different machine learning portfolios and the market factor. Interestingly, the equal-weighted US decile spreads have high average negative correlations to the market, implying that the portfolio is often long low-beta names and short high-beta names. As discussed below, however, these correlations and exposures vary considerably over time. If we scale positions by 1/vol we get much smaller average correlations to the market factor. Because our forecasts tend to short high-volatility securities, 1/vol position sizing leads to a portfolio that is slightly net long in dollar terms, thus negating much of the negative average correlation with the market.

Panel B shows average monthly excess returns and alphas (gross of t-costs) relative to the four-factor model. Excess returns and alphas tend to be larger for the 1/vol portfolios, both in the US

and ROW. This is not surprising since we trained the machine learning algorithms to forecast returns scaled by their standard deviations. The 1/n weighted decile spreads still show highly significant returns and alphas, but their negative market exposure reduces their raw returns. This is also reflected in the relatively high R-squared for the US equal-weighted portfolio, indicating that much of its variance is explained by the four risk factors, most notably the market factor (MKT).

We also include results for a benchmark strategy that uses the same factors, but uses linear regression (OLS) to forecast stock returns. Positions are set at 1/n (equal weights). Results for this benchmark OLS strategy are positive, but average returns and alphas are considerably larger for the machine learning algorithms. We also report results for a strategy that finds the ten factors with the highest Sharpe ratio up to each point in time, and then equal weights these top-ten factors. Results for this "top-ten" strategy are better than results for the OLS benchmark, but the machine learning results are better still, both in the US and in the rest of the world. While we would not recommend using simple decile spreads as a trading strategy, we explore whether the decile spreads are significantly reduced when we deduct reasonable estimates for trading costs. Panel C subtracts 30 bps in assumed round-trip trading costs; alphas remain highly significant.

Interestingly, the machine learning portfolios load negatively on value (HML) and small size (SMB) on average, but only the US 1/n loading on SMB is significant. This indicates that the positive results are not driven by common risk factors. Loadings on momentum (MOM) are positive in the US and negative in the ROW, but again only the US 1/n result is significant. These results suggest that portfolio construction has a significant impact on portfolio risk.

Since alphas tend to be more significant than excess returns, it is unlikely that the four-factor model explains the returns to the machine learning strategy. Rather, it appears that the machine

learning composite extracts information from other factors, or exploits time-varying relationships between factors and returns that aren't captured by a linear risk model.

Equity market neutral (EMN) hedge funds had relatively low returns during the sample period. For example, the HFRI EMN Index returned just 0.24% per month from 2004-2016. During the same period, all of the machine learning decile spreads returned more than 1% per month after deducting assumed transaction costs.

In Table 4 we report Fama and MacBeth (1973) multiple regressions coefficients for aggregate machine learning forecasts using both raw monthly excess returns and monthly excess returns scaled inversely to volatility (1/vol). We control for a battery of popular quantitative factors. We conduct the analysis separately for the US and the rest of the world (ROW). All variables are standardized to make coefficients comparable.

We find that the ML composite forecasts are significantly related to returns across all specifications, even after controlling for many popular quant factors. Perhaps surprisingly, a few of the control variables remain modestly significant. This is 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, then we'd expect the control factors to offer little incremental value. On the other hand, the algorithms only consider point-in-time information, and it is much easier to identify successful factors *ex-post* than *ex-ante.*

In Appendix C, we report returns and four factor alphas of the long (top decile) and the short (bottom decile) sides of the portfolio separately. Similarly to Gu, Kelly, and Xiu (2018) we find that most of the outperformance comes from the long side. This is not surprising because equity markets outperformed Treasury bills over this period. When we adjust for market and style effects, however, we find that four-factor alphas are significant for both the long and short

portfolios, in both the US and the rest of the world. But looking at the long and short legs independently does not tell the whole story. The t-statistics for the spread alphas in Table 3 are much larger in magnitude than for the sum of the individual long and short t-statistics in Appendix C. Because we constrain the machine learning algorithms to forecast volatility-adjusted, industry-standardized returns, the decile spreads eliminate most industry risk and idiosyncratic volatility, leading to higher risk-adjusted returns for the decile-spread portfolios. This illustrates the potential benefits of feature engineering.

Feature Importance. We examine the most important features for a gradient boosted classification tree model estimated over the last ten years of the sample period in the US. We report the ten most influential features in Appendix A. Gu, Kelly, and Xiu (2018) find that price-trend factors, volatility and liquidity variables are by far the most important features. Our analysis suggests that these categories are important, but we also find that the percentage of shares sold short, the difference between put and call implied volatilities, and characteristics derived from financial statement information are also among the ten most important features.

Time-varying exposures. One potential advantage of machine learning forecasts is that algorithms can dynamically learn changing relationships between factors and returns. Figure 4 shows the time series of monthly cross-sectional correlations between the US machine learning composite forecasts and the factors used to calculate the Carhart (1997) four-factor model – i.e., Beta, Size, BP, and Momentum.

It is clear that the average exposures in Table 4 don't tell the whole story – that is, there is significant time variation in all of these exposures. For example, despite a positive average correlation, we see that the ML 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 tend to coincide with or follow periods of significant underperformance for momentum. For example, following the momentum crash documented in Daniel and Moskowitz (2016), we see that exposure to momentum becomes negative, with a cross sectional correlation of around -0.3 for several months in 2010. As noted above, the average correlations to Size and Beta tend to be negative, but they vary considerably over time and are occasionally positive.

While it is difficult to precisely determine how much of the machine learning strategy's alpha comes from factor timing, these factor exposures are clearly much more variable than the exposures of typical linear factor models.

CONCLUSION

In this article we discuss how practitioners can use machine learning algorithms (MLAs) for stock selection while avoiding the primary problem with these techniques: overfitting. Low signal-to-noise ratios in security selection mean that overfitting is always a risk, especially with techniques like MLAs that provide little structure to the analysis. Conversely, because they don't impose structure, MLAs can uncover complex, non-linear patterns that are hard to tease out with traditional statistical techniques like OLS. They also tend to work better than OLS when considering numerous collinear factors.

We discuss 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 won't replace human experts anytime soon (at least in investing).

In the final section, we present a case study based on some of the proposed techniques. We demonstrate that, properly applied, machine learning algorithms can use a wide variety of firm 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 like OLS or equal weighting the "top ten" factors. These results are robust to various risk adjustments, and work well both in the US and other developed market regions. Although it can be difficult to accurately assess which signals are driving the results, we demonstrate that traditional factor exposures vary considerably over time, implying that factor timing and conditional analyses (i.e., non-linear effects) contribute to these positive results.

Table 1: Glossary:

Activation functions in neural networks determine the output of nodes given inputs. Non-linear activation functions allow neural networks to learn non-linear patterns. Popular activation functions include Rectified Linear Units or 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 with each other making it possible to learn many interactive relationships. However, artificial neural networks can be easy to over-fit, 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 learn deeper architectures without overfitting.

Bagging (Bootstrap Aggregating) algorithms combine forecasts generated from base algorithms on randomly sampled learning sets. Random forests are an example of the application of bagging to CART models. In contrast to boosting, forecasts of the base algorithms are equally weighted. Bagging tends to increase stability of algorithms and helps prevent overfitting.

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

Boosting algorithms combine forecasts from many base algorithms such as CART. 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 since they are easier to overfit. They also take longer to run than bagging because they require sequential processing.

CART (Classification and Regression Tree) models form the basis of many machine learning algorithms. However, CARTs are prone to overfitting, and are rarely competitive on their own. CARTs can detect hierarchical (and hence non-linear) 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 leading to 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.

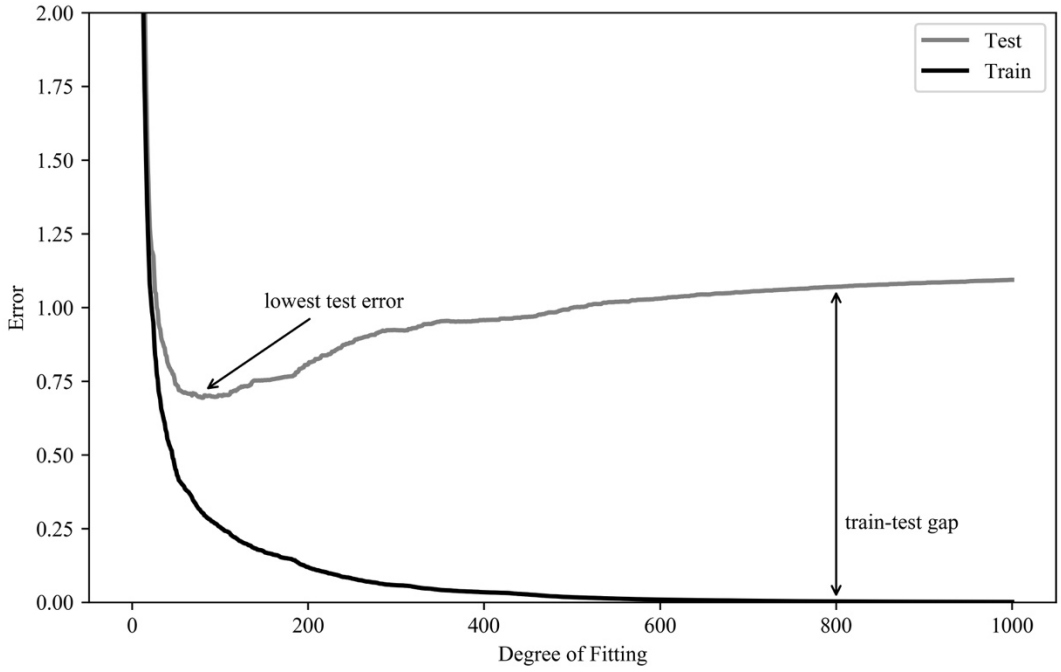
Multi-Layer Perceptron (MLP) is a feed forward neural network that consisting of three or more layers. MLPs are some of the earliest and well-understood neural networks.

Random Forests combine many CART models (or trees) by averaging their forecasts. This can often diversify away errors due to overfitting, and 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.

RBF kernel or radial basis function kernel is a commonly used kernel in support vector machines. The RBF kernel is especially well suited to problems that are potentially non-linear.

Support Vector Machines are effective classifiers in higher dimensional spaces and can employ either linear or non-linear kernels. While often effective, SVMs can run slowly with large datasets.

Figure 1: In-Sample/Out-of-Sample Error vs. Degree of Fitting



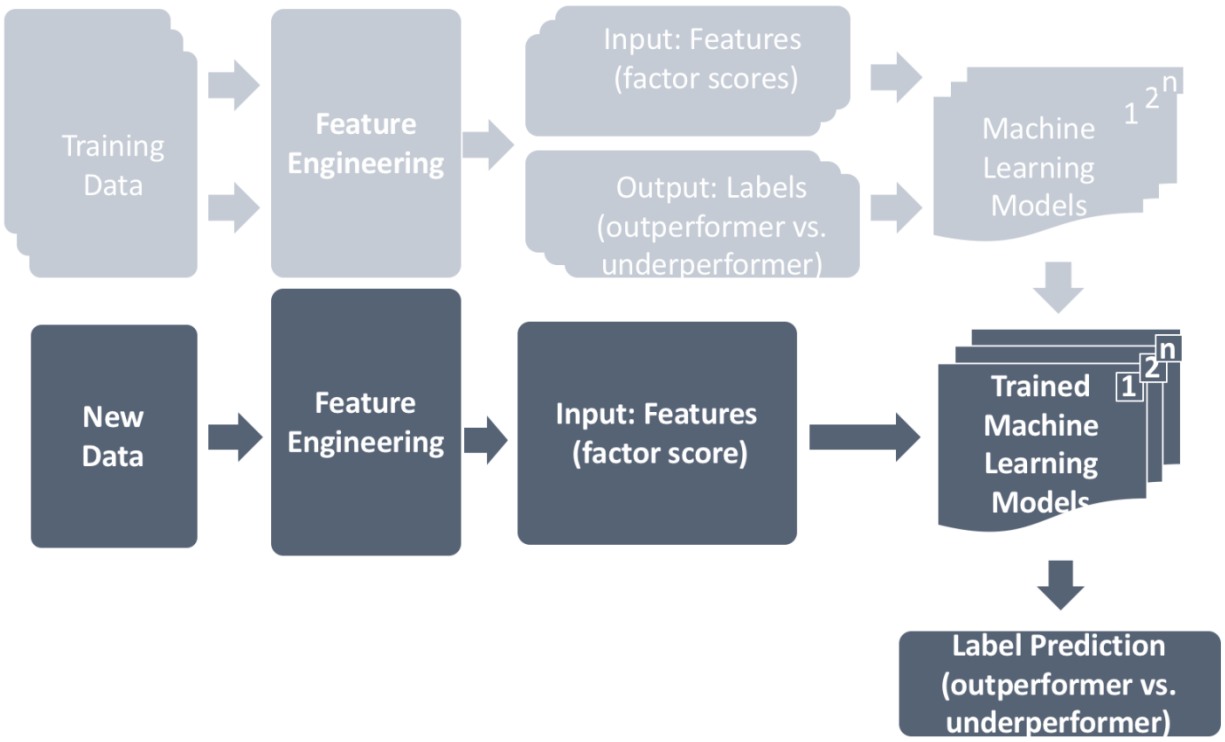
This Figure shows in-sample and out-of-sample error rates for a gradient boosted regression tree classifier using simulated noisy data. The X-axis represents the number of boosting iterations (more iterations allow the algorithm to better fit past data), and the Y-axis shows error. The dark line is in-sample; the light line shows classification performance on a hold-out sample.

Table 2: Summary Statistics

Country Code	Country	Average No. of Stocks	Average Company Size $Bln	Average Weight in Global Portfolio
AUS	Australia	206	4.6	3.5
AUT	Austria	28	3.6	0.5
BEL	Belgium	41	7.1	0.7
CHE	Swizterland	56	17.1	0.9
DEU	Germany	110	11.5	1.9
DNK	Denmark	40	4.7	0.7
ESP	Spain	65	9.9	1.1
FIN	Finland	39	4.8	0.7
FRA	France	114	14.7	1.9
GBR	Great Britain	356	8.0	6.0
GRC	Greece	41	2.2	0.7
HKG	Hong Kong	287	4.7	4.9
IRL	Ireland	16	5.7	0.3
ITA	Italy	75	8.1	1.3
JPN	Japan	1052	3.6	17.8
NLD	Netherlands	39	8.8	0.7
NOR	Norway	62	3.7	1.0
NZL	New Zealand	26	1.7	0.4
PRT	Portugal	16	4.7	0.3
SGP	Singapore	101	3.1	1.7
SWE	Sweden	87	5.1	1.5
USA	United States	3050	5.7	51.6

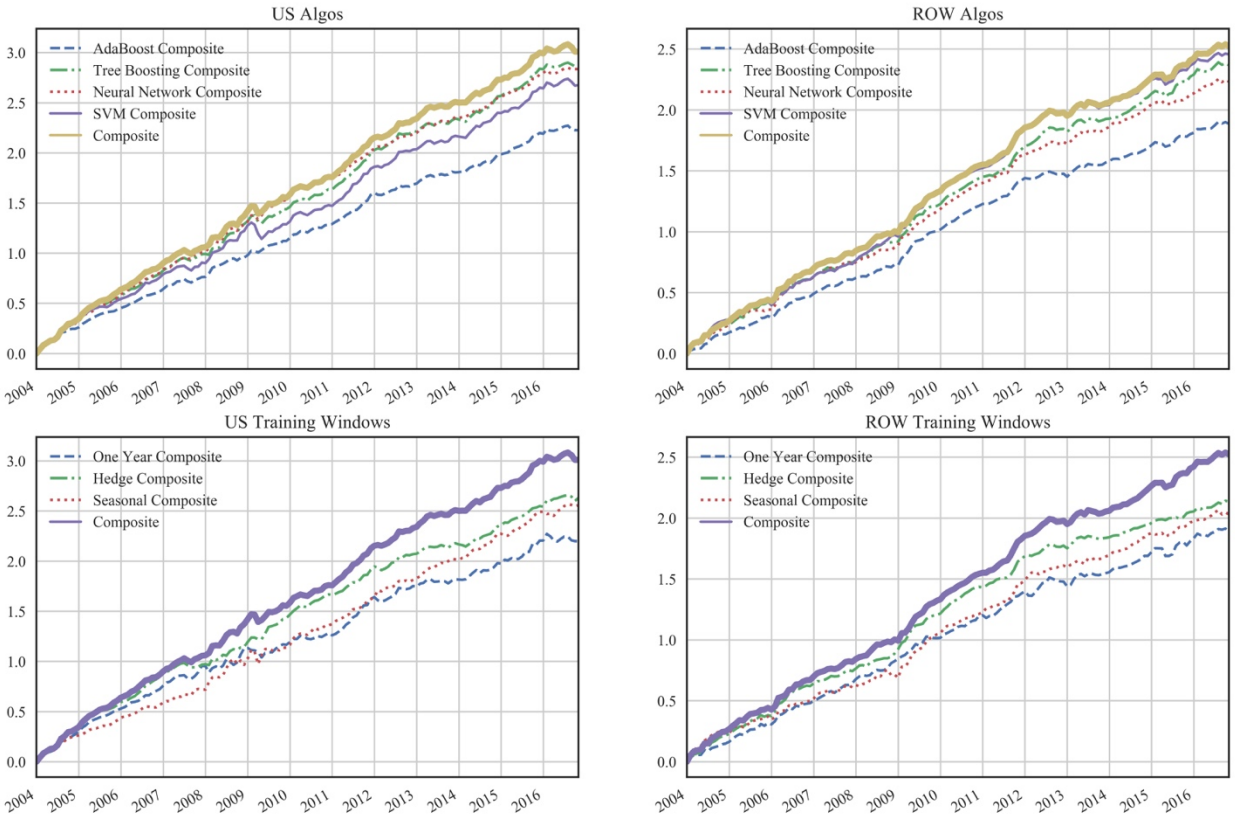
This Table presents summary statistics. The reported numbers are for the period in which we show returns: 2004-2016. The average number of stocks represents the average number of securities in the sample each month. The Average company size column shows the average size in billions. The weight in global portfolio column reports the average weight in the global portfolio for each country. In the US data starts in 1988, in all other countries data starts in 1994.

Figure 2: Example Learning Workflow



This Figure illustrates the machine learning process used for the example. Each month we create three training samples with historical data: recent, seasonal and hedge. Next, we apply feature engineering which includes standardizing features within region and industry. We also categorize stocks into winners and losers in region/industry buckets. Next, we train our machine learning algorithms on the historical data. Once new feature data are received, we apply the same feature engineering steps that were applied on the training set and generate forecasts with each of our previously trained algorithms.

Figure 3: Cumulative Performance of Machine Learning Forecasts by Algorithm and Training Windows (Risk-Weight)



Graphs show cumulative decile spread returns without compounding for different subsets of forecasts. Stocks in each decile are weighted by the inverse of their 100-day volatility. The algorithm subsets use all training windows for the given algorithm, while the training window subsets use all algorithms for the given training window. The composite forecast uses all training windows and algorithms.

Table 3: Regression results vs. FF Factors

A. Correlations

	US Equal-Weight	US Risk-Weight	ROW Equal-Weight	ROW Risk-Weight	Mkt-RF
US Equal-Weight	1.00	0.81	0.32	0.29	-0.57
US Risk-Weight		1.00	0.28	0.30	-0.20
ROW Equal-Weight			1.00	0.96	-0.16
ROW Risk-Weight				1.00	-0.05
Mkt-RF					1.00

B. Regression of ML decile Spreads on Fama-French Factors

	US				ROW			
	Equal-Weight	Risk-Weight	Top 10 Benchmark	OLS Benchmark	Equal-Weight	Risk-Weight	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-RF	-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 Rsq	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

C. Regression of ML decile Spreads with Transaction Costs on Fama-French Factors

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

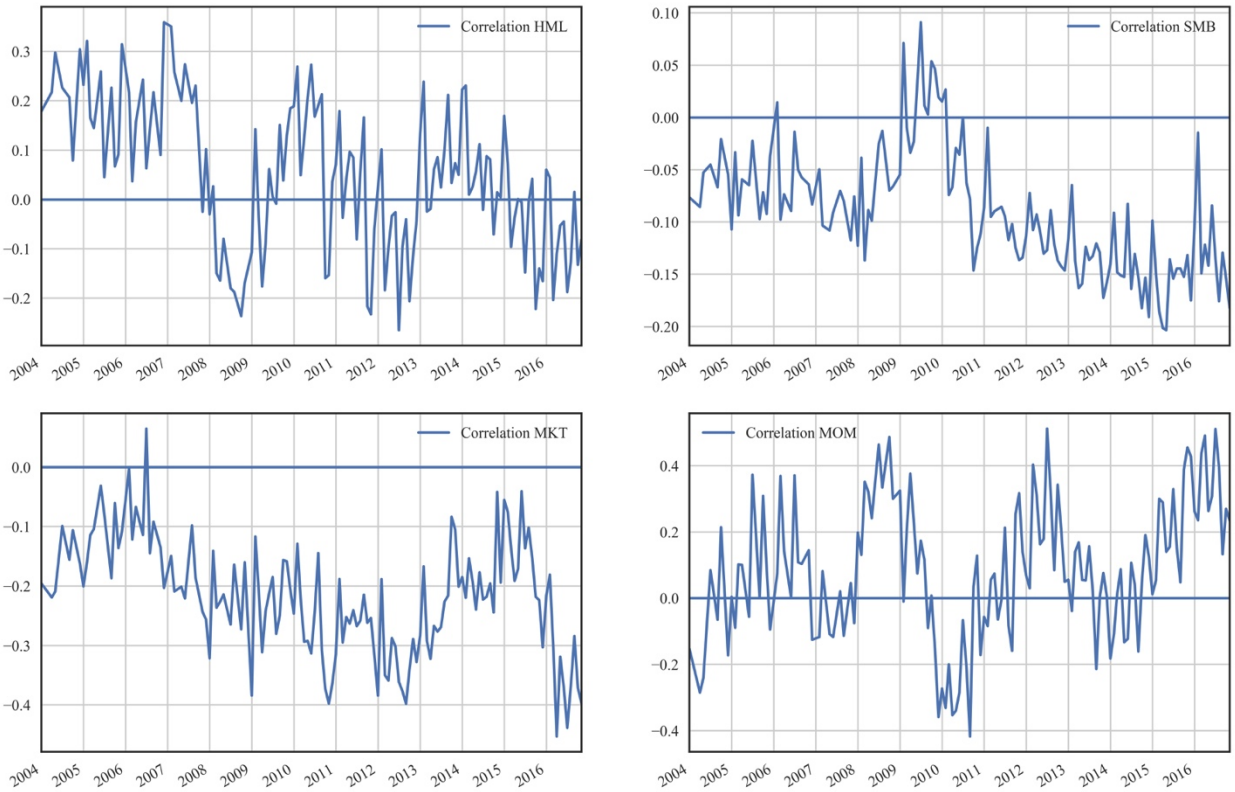
Panel A shows correlations between a number of machine learning portfolios and the market factor. Panel B shows average monthly excess returns and alphas, as well as factor loadings from regressions of machine learning returns on the Carhart 4-factor model. t-Statistics are in parenthesis. For comparison, we also present results of a benchmark strategy that estimates returns using ordinary least squares (OLS) and for a strategy that recursively finds the ten factors with the highest Sharpe ratio, and then equal-weights them. US includes domestic stocks, while ROW (rest of the world) includes all other developed-market regions (i.e., Europe, Japan and Asia ex Japan). Panel C subtracts transaction cost estimates of 15 bps per side.

Table 4: Fama Macbeth Multivariate Regressions

	US		ROW	
	Equal-Weight	Risk-Weight	Equal-Weight	Risk-Weight
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	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)
EPS Growth	0.12	0.05	-0.02	-0.04
	(1.08)	(0.64)	(-0.35)	(-0.65)
1M Reversal	0.21	0.12	0.16	0.12
	(0.87)	(0.65)	(0.83)	(0.67)
Low Vol	-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)

This exhibit summarizes monthly Fama-Macbeth cross-sectional regressions (average slopes and t-statistics in parentheses). Monthly returns are regressed on lagged firm characteristics and machine learning forecasts. We present results for the US and the rest of the world (ROW). For the equal-weighted specifications the dependent variable is excess returns; for the risk-weighted specifications the dependent variable is excess return scaled by the 100-day trailing standard deviation.

Figure 4: Cross-Sectional Correlations of Composite US Machine Learning forecasts to Book-to-Price Size, Beta and Momentum

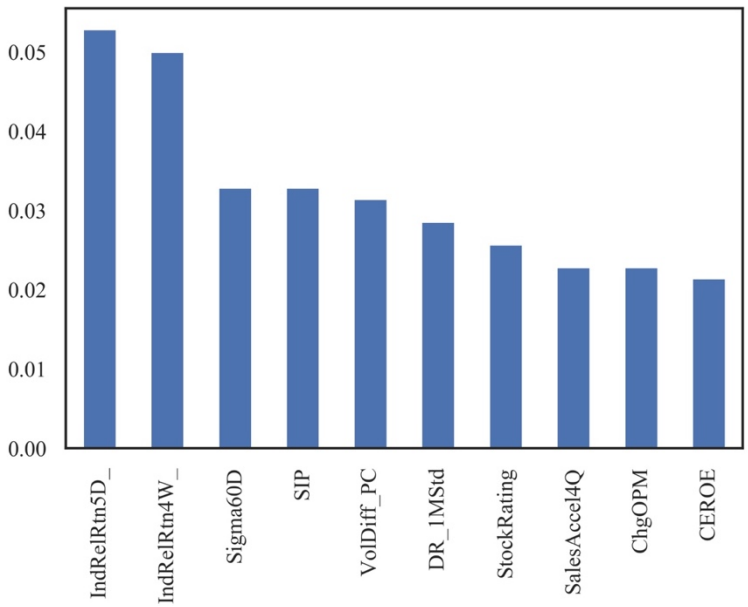


This figure shows time series of cross-sectional correlation coefficients of the composite scores with the characteristics used to construct the Fama and French (1992) factors and the Carhart (1997) momentum factor: book-to-market, small size, beta, and 12-month momentum.

Appendix A: Feature Importance

Due to their non-linear nature, MLAs do not have interpretable coefficients. However, we can frequently tease out which features contribute most to the forecasts. Below we show average feature importance for a gradient boosted classification tree in the US, which is assessed by examining how often a feature appears in a tree and at what level. 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.

Figure A1: Feature Importance for a Gradient Boosted Classification Tree (US)



The graph shows feature importance for the 10 most important features. The model was fit with XGBoost from Dec 2006 to Dec 2016. IndRelRtn5d is the industry-relative return over the last 5 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 trailing 60 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 latest 1-month return volatility. StockRating is the census 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 4 quarters ago. CEROE is the trailing 12-month cash flow from operations divided by the book value of common equity.

Appendix B: Information Coefficients (ICs)

In Figure 3 and Table 3, we use decile spreads to measure the performance of various forecasting approaches. Below we show average monthly information coefficients for these same approaches. The IC measures the Spearman rank correlation between stock rankings and future returns across the full range of forecasts, not just the extreme deciles.

These results are consistent with those in Figure 3 and Table 3. That is, the ML composite has better results than any individual algorithm/training window combination. It also handily outperforms ordinary least squares (OLS) and a benchmark that equal-weights the ten factors with the highest prior Sharpe ratios.

	US		**ROW**	
	Rank IC	T-Statistic	Rank IC	T-Statistic
ML Composite	6.48%	15.87	6.43%	16.37
Adaboost (12M)	3.19%	9.07	3.54%	10.98
SVM (12M)	4.61%	8.00	5.08%	10.35
Tree Boosting (12M)	4.66%	9.59	4.79%	11.12
Neural Network (12M)	3.99%	12.94	3.17%	11.97
AdaBoost (Seasonal)	3.17%	11.53	3.25%	11.75
SVM (Seasonal)	5.01%	9.65	4.79%	10.56
Tree Boosting (Seasonal)	5.00%	12.04	4.53%	11.89
Neural Network (Seasonal)	5.20%	15.39	4.41%	11.92
AdaBoost (Hedge)	3.92%	14.06	3.53%	13.52
SVM (Hedge)	4.94%	13.21	5.23%	14.36
Tree Boosting (Hedge)	5.00%	14.14	4.78%	16.16
Neural Network (Hedge)	4.58%	14.24	4.47%	14.61
Top 10 Factors Benchamrk	2.81%	6.33	4.49%	9.44
OLS Benchamrk	3.36%	5.78	2.71%	4.72

Appendix C: Long and Short Portfolios

We show excess (to T-bill) returns and 4-factor alphas for the long and short legs of the risk-weighted ML decile spreads. Most of the excess returns come from the long portion, but four-factor alphas are significant for both the long and the short sides.

	US		**ROW**	
	Long	**Short**	**Long**	**Short**
Excess return	1.90	-0.03	1.50	-0.13
	(3.54)	(-0.05)	(4.18)	(-0.33)
4 Factor Alpha	1.13	-0.96	0.95	-0.69
	(4.43)	(-3.28)	(4.72)	(-2.97)

Bibliography

Alberg, John, and Zachary Lipton, 2017, Improving Factor-Based Quantitative Investing by Forecasting Company Fundamentals, *arxiv.org*

Anon., 2016, Tree-Based Conditional Portfolio Sorts: The Relation Between Past and Future Stock Returns,, 1–81.

Asness, Clifford S, 2016, The Siren Song of Factor Timing aka "Smart Beta Timing" aka 'Style Timing', *The 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, *SSRN Electronic Journal*.

Batres-Estrada, Bilberto, 2015, Deep learning for multivariate financial time series.

Carhart, Mark, 1997, On Persistence in Mutual Fund Performance, *Journal of Finance* 52, 57–82.

Clemen, Robert T, 1989, Combining forecasts: A review and annotated bibliography, *International Journal of Forecasting* 5, 559–583.

Daniel, K, and T J Moskowitz, 2016, Momentum crashes, *Journal of Financial Economics* 122, 221–247.

de Prado, M Lopez, 2018, *Advances in Financial Machine Learning* (John Wiley & Sons).

Fama, E F, and Kenneth French, 2017, International tests of a five-factor asset pricing model, *Journal of Financial Economics* 123, 441–463.

Fama, Eugene F, and Kenneth R French, 1992, The cross-section of expected stock returns, *The Journal of Finance* 47, 427–465.

Fama, Eugene, and JD MacBeth, 1973, Risk, Return, and Equilibrium Empirical Test, *The Journal of Political Economy*, 1–31.

Gu, Shihao, Bryan T Kelly, and Dacheng Xiu, 2018, Empirical Asset Pricing via Machine Learning, *SSRN Electronic Journal*.

Heaton, J B, N G Polson, and J H Witte, 2016, Deep Learning in Finance.

Makridakis, Spyros, and Michèle Hibon, 2000, The M3-Competition: results, conclusions and implications, *International Journal of Forecasting* 16, 451–476.

Miller, Keith L, Chee Ooi, Hong Li, and Daniel Giamouridis, 2013, Size Rotation in the U.S. Equity Market, *The Journal of Portfolio Management* 39, 116–127.

Miller, Keith L, Hong Li, Tiffany G Zhou, and Daniel Giamouridis, 2015, A Risk-Oriented Model for Factor Timing Decisions, *The Journal of Portfolio Management* 41, 46–58.

Morozov, A, J. Wang, and L Borda, 2012, *The Barra Global Equity Model (GEM3)*.

Schapire, R., 1990, *The Strength of Weak Learnability*. Ed. Kluwer Academic Publishers. Vol. 5.

Srivastava, N, and G Hinton, 2014, Dropout: A simple way to prevent neural networks from overfitting, *The Journal of Machine Learning Research*.

Takeuchi, L, and YYA Lee, 2013, Applying Deep Learning to Enhance Momentum Trading Strategies in Stocks, *Technical Report, Stanford University*.

Timmermann, A., 2006, Forecast combinations, *Handbook of economic forecasting* 1, 135–196.

Wang S., and Y Luo, 2012, The Rise of the Machines, *Deutsche Bank Quantitative Strategy*.

Wang, S, and Y Luo, 2014, The rise of the machines, III, *Deutsche Bank Quantitative Strategy*.