國立交通大學

資訊管理與財務金融學系

財務金融碩士班

碩士論文

傳統方法與機器學習在資產定價上的實證比較

An Empirical Comparison between Traditional Methods And Machine Learning in Asset Pricing Models

研 究 生：李育賢

指導教授：鄧惠文 博士

中華民國一〇九年六月

傳統方法與機器學習在資產定價上的實證比較

An Empirical Comparison between Traditional Methods And Machine Learning in Asset Pricing Models

研 究 生：李育賢　　　　Student：Yu-Hsien Li

指導教授：鄧惠文 博士　　Advisor：Dr. Huei-Wen Teng

國立交通大學

資訊管理與財務金融學系財務金融碩士班

碩士論文

A Thesis
Submitted to Graduate Program of Finance
Department of Information Management and Finance
National Chiao Tung University
in Partial Fulfillment of the Requirements
for the Degree of
Master
in
Finance

June 2020
HsinChu, Taiwan, Republic of China

中華民國一〇九年六月

傳統方法與機器學習在資產定價上的實證比較

研究生：李育賢　　　　　　　　　指導教授：鄧惠文 博士

國立交通大學資訊管理與財務金融學系財務金融碩士班

摘要

近幾年，機器學習被廣泛地應用在金融學術界及產業界中。本文同時使用 102 個公司特徵和 8 個總體經濟因子，對傳統線性和機器學習模型的方法，在股票月報酬預測能力上進行實證分析。在我們的分析中顯示，機器學習優於傳統模型，因為機器學習更容易解決高維和多重共線性的問題，而神經網絡模型擁有最好的報酬預測能力。另外，當我們根據報酬預測值建立 Bottom-up 的投資組合後，不論在 Buy-and-hold 或 Long-and-short 的策略中，XGBoost 和神經網絡模型，都具有較高的 Sharpe ratio。

關鍵字：Fama-MacBeth 迴歸，機器學習，報酬預測，彈性網路，隨機森林，迭代迴歸樹，XGBoost，神經網路

An Empirical Comparison between Traditional Methods And Machine Learning in Asset Pricing Models

Student: Yu-Hsien Li Advisor: Dr. Huei-Wen Teng

Graduate Program of Finance
Department of Information Management and Finance
National Chiao Tung University

ABSTRACT

Machine learning has gained a vast amount popularity in financial applications both in academy and industry recently. In this thesis, we conduct empirical comparisons about returns predictability between traditional linear models and machine learning techniques. We focus on using 102 firm characteristics and 8 macroeconomic predictors simultaneously in predicting stock returns in monthly basis. Our analysis shows that machine learning techniques outperform traditional models, possibly because they are able to avoid the multi-collinearity problem when a large number of predictors are used. Specifically, the neural network outperforms the other competitors in terms of predictability for returns of stocks. Moreover, when building bottom-up portfolios based on the predicted stock-level returns for both buy-and-hold and long-short strategies, XGBoost and neural networks produce portfolios with higher Sharpe ratios.

Keywords: Fama-MacBeth Regression, Machine Learning, Return Prediction, Elastic Net, Random Forest, Gradient Boosting Regression Trees, XGBoost, Neural Networks

Contents

List of Figures

List of Tables

1 Introduction

Since Sharpe (1964) proposed the capital asset pricing model (CAPM) to explain the variation of stock returns with market excess returns, there have been numerous studies to understand the relationship between cross-section stock returns and systematic risk. Shortly after Roll and Ross (1980), Fama and French (1993), and Fama and French (1996), firm characteristics are founded as critical roles in empirical asset pricing field as well. Along with the rapid development in data collection and data mining, it is expected that more and more factors will be included in the future. Cochrane (2011) identify which pricing factors are able to provide useful information about stock returns, and Jeremiah et al. (2013) and Campbell R. Harvey (2016) summarize more than three hundreds characteristics from prior articles. Therefore, along with the proliferation of the available factors, it is of considerable importance to propose an adequate method to identify crucial factors for returns prediction and is able to accommodate a large number of factors.

Returns prediction is economically meaningful in empirical asset pricing fields. There has been an increasing number of studies about including hundreds of predictors simultaneously in the prediction models. In addition to the standard problems such as estimation errors in high dimensions and overfit, the collinearity problem among predictors makes the traditional method fail. To overcome the collinearity problem, Lewellen (2015) and Green et al. (2017) use Fama-MacBeth regression and other linear regression to select predictors by dropping highly correlated predictors. With the flexibility and generality, machine learning has been shown to be useful in identify information among a large set of predictors.

Machine learning techniques are capable of variable selection with regularization methods and dimension reduction tasks to avoid to overfit, so they are frequently applied for prediction tasks. For example, Kelly et al. (2019) use instrumental principle component analysis (IPCA) to pick up a linear combination of subset among characteristics for dimension reduction and thus produce less estimation errors. Fengy et al. (2020) apply linear regression with shrinkage method, LASSO, to select important predictors among hundreds of predictors. Gu et al. (2019), Tsang and Wong (2019), and Gu et al. (2020) apply deep learning methods in finance, including tree-based models, feed-forward neural network, and auto-encoder model. The powerful predictability of machine learning has been demonstrated for cross-section and time-series stock returns.

In this thesis, we conduct a comparison analysis between traditional methods and machine learning techniques in asset pricing models. Specifically, we are interested in returns prediction using 102 firm characteristics and 8 macroeconomic factors. For traditional methods, we focus on linear models, and include penalized regression, Fama-MacBeth regression. For machine learning techniques, we consider random forest, boosted regression trees, XGBoost, and neural network. Our goal is to measure the predictability of each model by simultaneously using a large set of pricing factors, which are reported to be statistically significant in prior studies.

As an extension to Gu et al. (2020), we consider Fama-MacBeth regression and XGBoost (Green et al. (2017) and Chen and Guestrin (2016)). We simultaneously include 102 firm characteristics and 8 macroeconomic factors and then compare the prediction performance of each model by evaluating the R^2_{oos} of stock-level returns and predicted portfolio-level returns in out-of-sample. We apply machine learning techniques in the field of asset pricing, and will show that machine learning techniques outperform traditional methods both in stock-level returns prediction the resulting portfolios.

With the ability to accommodate nonlinear predictive relationships, machine learning improves the predictive R^2 substantially. Specifically, the out-of-sample predictive R^2 of neural networks is higher than traditional methods. In contrast, OLS produces negative predictive R^2. The failure of OLS indicates that assumption of a linear regression model is violated. However, the ENet extends the OLS by including shrinkage method, and it improves the predictability greatly. Machine learning techniques outperform traditional methods predictive R^2 in general. For tree-based models, our empirical studies select trees with shallow depth in the cases of random forest, boosted trees, and XGBoost. Although the predictability increases as the neural network model gets ”deeper”, but the improvement with a deeper neural network becomes marginal. This indicates that there remains a limitation of deep neural network.

Portfolio-level analysis can help to understand further difference between machine learning techniques and traditional methods. We build up bottom-up portfolio-level return forecasts based on the stock-level return forecasts from each model. Constructing portfolios based on the predicted returns can avoid the unpredictable stock-level noise and boost the strength of detecting predictable associations. The economic gains could be larger. Indeed, the compound annual growth rate (CAGR) of machine learning methods can reach about 20% of a buy-and-hold investor, relatively to the highest 17% among traditional linear models. Also, XGBoost earns annualized Sharpe ratios higher than 1.68 with the long-short strategy. This shows that tree-based models can distinguish the stocks with highest and lowest return forecasts, even although they are unable to predict stock returns exactly and produce negative predictive R^2. Finally, machine learning techniques outperform traditional linear models when detecting trend of the stocks with higher predicted returns, although they appear to be comparable in distinguishing stocks with lower predicted returns forecasts.

The rest of thesis is organized as follows. Section 2 provides literature reviews. Section 3 describes the details of methods addressed in this thesis. Section 4 provides the empirical results and summarizes our findings. Section 5 concludes, and Section 6 gives directions for future research.

2 Literature Reviews

In the following, we review the classical empirical asset pricing models, the Fama-MacBeth regression, and machine learning used in the field of asset pricing.

2.1 Empirical Asset Pricing Models

Because more and more new data and pricing factors have been used and discovered, Goyal and Welch (2008) indicate that the results from past asset pricing articles are no longer useful for investors. They reexamine whether pricing factors which are suggested to use in past articles are still useful to predict the risk premium with the same methods. They find these factors perform poorly both in training and testing periods and are unable to provide enough information to help investors make decision. As a result, they don't consider that previous pricing factors would benefit investors to predict returns.

Although Fama-French 3-factor and Carhart 4-factor model have high impacts in the field of empirical asset pricing, Hou et al. (2015) indicate that they have gradually fail to account for a wide array of asset pricing anomalies. As a remedy, Hou et al. (2015) propose the q model including the market, size, investment, and profitability factors to explain the cross-section stocks returns. Also, they prove that q-model can capture more anomlies than Fama-French 3-factor model and Carhart 4-factor model. Besides, many anomalies appear to be related with the investment and profitability effects.

Campbell R. Harvey (2016) propose a higher threshold for evaluating a new factor model with a $t-$statistic greater than 3.0 after estimating 300+ pricing factors summarized from earlier studies. There has been fruitful research on using pricing factors to explain the variation of stock returns. As the cost of the data collection and data mining has decreased a lot in recent year, it is foreseeable to obtain more factors with appearing to be significant with t-statistic 2.0. As a result, it is critical to avoid the data snooping bias when establishing significance for factors.

Mclean and Pontiff (2016) discover lots of factors in the field of finance, accounting, and economics to explain the variation in cross-section stocks returns. It becomes an interesting question to compare the prediction power for a larger set of factors for in-sample, out-of-sample, and post-publication returns. Their goal is to compare return predictability of each factor over three distinct periods and try to figure out the difference between these explanations by observing the changes in returns predictability. They summarize three types of possible reasons resulting in changes of return predictability. First, if the return predictability in published results merely from statistical bias, the predictability shouldn't exist for out-of-sample data. Second, if the return predictability only reflects rational expectation hypothesis, pre- and post- publication return predictability should be equal after removing the influence of statistical biases. This makes investors hold the same investment position after learning the publication. Third, if the return predictability reflects mispricing, and the academic publication help investor learn from and trade against mispricing. Thus, it is expected that the predictability associated with the factors will disappear or least decay after publication. They find that both returns predictability of post-publication and out-of-sample are lower than in-sample. Besides, investors learn about the mispricing from publication since the correlation of portfolio returns, which are constructed based on the post-publication and other published predictors, become higher.

2.2 The Fama-MacBeth Regression

Green et al. (2017) use the Fama-MacBeth (FM) regression in Fama and MacBeth (1973) to examine which characteristics provide the independent information to the average monthly stock returns. Then, they use a Fama-MacBeth regression via rolling windows approach to predict stock returns. They show that 12 characteristics can provide more independent information. Also, the determinants of returns are intrinsically small instead of a small number of characteristics absorb the information from many other characteristics. Besides, they find that microcap and non-microcap stocks have different effects on their monthly returns when using the same firm characteristics to build the forecasting model.

In Lewellen (2015), there is relatively less effort to forecast future stock returns with known predictors instead of attempting to find a new factor in the empirical asset pricing field. In addition, literature shows that many firm characteristics are correlated with stock returns, but little evidence exists on whether the characteristic can actually be used to predict the future returns in real time. Little article guides how investors can simultaneously combine these amount of firm characteristics to forecast returns and then change their trading strategies. As a result, this paper indicates that there are two big issues about forecasting stock returns. (1) How much cross-sectional variation in expected returns can we actually predict? And (2), How reliable are estimates of expected returns from FM regression? This paper considers that FM regressions can provide a effective way to aggregate many firm characteristics to estimate stock future returns. Besides, the coefficients of characteristics also give us a statistical way to describe how they influence stock returns. Thus, they use FM regression with up to 15 firm characteristics without prior screening in order to simulate condition that investors do not know which characteristics are the key factors in advance. They estimate how well investors can use average or cumulative historical intercept and slopes of firm characteristics in FM regression via rolling windows approach. Their finding is similar with past studies that Fama-French three-factor model does not offer reliable estimates of expected returns and characteristic-based regressions have better prediction power in out-of-sample data. Also, similar to Green et al. (2017), they show that there are difference effects on forecasting between different scale firms with the same characteristics.

2.3 Machine Learning in Asset Pricing

Kelly et al. (2019) propose the instrument principle component analysis (IPCA) to pick up a linear combination of characteristics which provides most information about future stock returns. When a large set of factors is used to predict stock returns, the estimation error from the multicollinearity problem usually occurs. IPCA allows to find the latent factors beyond the observable factor loading, and thus can improve the predictability.

The goal of Heaton et al. (2016) is to apply auto-encoder model to select a small group of stocks from components of a specific index and using them to construct a equal-weight portfolio whose value trend is similar to the target index trend. Auto-encoder is a well-known reduction dimension method, and its outputs theoretically are the same as inputs. Its objective is to minimize the difference between inputs and outputs. In this kind of neural network, it first compresses the data information through the bottle-neck structure and then restore the compressed information from the last step. By using auto-encoder model, they can identify stocks with the most common information through the bottle-neck structure and thus have ability to construct a portfolio to replicate the price trend of target index and reduce the tracking error while holding this portfolio.

To predict returns, Gu et al. (2019) indicate that incorrect models or assumptions will cause considerable estimation errors. Also, when the set of firm characteristics become large, it causes the high dimension problem and increases the difficulty of returns prediction. Based on the ”anomaly” view of characteristics asset return prediction, they propose an auto-encoder model to estimate the compensated aggregate risk exposures through these firm characteristics for individual stock returns. Auto-encoder model is a well-known deep learning method and often used for conducting dimension reduction like PCA. It allows us to predict returns through the nonlinear and interactive effects on factor exposure (the coefficients of characteristics covariates) and get a far small prediction errors compared to the Fama-French three-factor model.

In Fengy et al. (2020), as the asset pricing field develops, researchers have produced hundreds of factors. With the proliferation of factors, they propose a framework for evaluating whether the contribution of a new factor provide enough explanatory power for asset pricing. More specifically, they estimate the marginal effect of a new factor beyond what is explained by a set of factors in the testing periods. They indicate that simply applying a model with shrinkage methods, such as ridge, elastic net, LASSO etc., to a large set of factors and check which factors are selected is not a reliable way to determine the actually important factors.

As a remedy, Feng et al. (2019) propose a deep learning framework for improving the out-of-sample forecast performance by using additional ”deep factors” which is generated from a variety of firm characteristics and macroeconomic factors through feed-forward neural network. In the process, they focus on estimating a reduced-form asset pricing model, so feed-forward neural network is suitably applied into characteristics-based factor model for dealing with high dimension problems. They provide a systematic approach for generating ”deep factor” by using a feed-forward neural network, which is a nonlinear combination of the firm characteristics and macroeconomic predictors. After adding this ”deep factor”, prediction results significantly perform better that Fama-French factors. As a result, they believe that deep learning methods are able to provide useful new insights to finance and economic.

Finally, Tsang and Wong (2019) view the multi-period portfolio optimization as a Markov decision process. They propose a deep neural network (DNN) consisting of neural networks, which represent the decision steps, to solve the multi-period portfolio optimization problem in discrete-time subject to investment constraints. The multi-period portfolio optimization is required to meet the investment constraints. In addition, they assume returns to follow multivariate AR(1)-GARCH(1,1) models. Then, they represent the portfolio problem with a utility function and prove for asymptotic convergence of the quadratic utility function. Finally, they provide a DNN architecture to portfolio optimization and show the resulting portfolio performs well.

3 Traditional methods

3.1 A General Form of The Model

To perform a comparative analysis of methods of empirical asset pricing with traditional methods and machine learning techniques, we consider linear models, tree-based methods, and neural networks, and use 102 firm characteristics and 8 macroeconomic predictors at time t to predict returns at time $t+1$. The general form of the asset pricing model to predict the future return is

$$r_{i,t+1} = E_t(r_{i,t+1}) + \epsilon_{i,t+1}, \tag{3.1}$$

where

$$E_t(r_{i,t+1}) = g^*(z_{i,t};\theta), \tag{3.2}$$

where the p-dimensional predictor $z_{i,t}$ is for the i-th firm at time t for $i = 1, ..., N_t$ and $t = 1, ..., T$. The parameter vector $\theta = (\theta_1, \ldots, \theta_p)'$ is the p-variate vector of parameters.

Note that the subscript in N_t indicates the number of firms at time t may be time-variant. In addition, we assume the conditional expectation $g^*(\cdot)$ is a general function. Equation (3.2) uses predictors (including the firm characteristics and macroeconomic factors) at time t to predict the stock return at time $t+1$. $E_t(r_{i,t+1})$ is the output of model, and the associated parameters are decided to minimize the out-of-sample prediction errors to the realized returns $r_{i,t+1}$.

Note that the function $g^*(\cdot)$ does not depend on i nor t, and the model will average the information from entire panel data by using the same form across firms and times. In other words, this is a pooled model. This model differs from traditional asset pricing models that use cross-sectional data at each time point or consider a time series model for each firm. This model predicts return at time $t+1$ only using information at time t but not at earlier times. Finally, for simplicity, we use the mean squared prediction error (MSE) as the objective function when estimating parameters for all methods.

Because MSE could be highly affected by variation of the predictors, we consider a regularization procedure to impose constrains on the parameters. This problem deteriorates particularly when the number of predictors is large. In fact, regularization allows us to reduce the influence of outliers, avoid the overfitting problem, and to improve the prediction. The regularization procedures are essentially shrinkage methods.

Hyperparameters (also known as tuning parameters) control the model complexity and critically affect the subsequent prediction of a parameter. They have to be carefully selected. Examples of hyperparameters include the scale of the penalization parameter in elastic net, the number of trees in tree-based models, and the number of hidden layers and the number of neurons in each hidden layer in neural network. Hastie et al. (2009) suggest general rules to tune the hyperparameters. Here, we split the original data into three disjoint datasets: training, validation, and testing sets, to manually decide the optimal hyperparameters.

The training set is used to estimate model parameters, and the validation set is used to decide the hyperparameters. To start, we set up a set of combinations of hyperparameters. Given a combination of hyperparamters, we forecast the data in the validation set using parameters estimated from the training dataset and calculate a measure of prediction errors. The optimal hyperparameters are selected to minimize the measure of prediction errors in the above repeated procedures. Finally, the third "testing set" is used for evaluating the model predictive performance, where the hyperparameters and parameters are decided using the training and validation sets.

3.2 The Simple Linear Model with Ordinary Least Squares Objective Function

The simplest linear model is the ordinary least squares (OLS). OLS usually exhibits high variation when the large p problem exists. It is expected the OLS performs poorly. Here, the OLS is used as a benchmark to emphasize the large p problem.

3.2.1 Model

The conditional expectation $g^*(\cdot)$ of the simple linear model is a linear combination of predictors.

$$g(z_{i,t}; \theta) = z_{i,t}'\theta. \tag{3.3}$$

3.2.2 The Objective Function

The objective function is used to estimate parameters. The least squares objective function is

$$\mathcal{L}(\theta) = \frac{1}{T}\sum_{t=1}^{T}\frac{1}{N_t}\sum_{i=1}^{N_t}(r_{i,t+1} - g(z_{i,t}; \theta))^2. \tag{3.4}$$

The parameter θ needs to minimize the above objective function.

3.3 The Simple Linear Model with Huber objective function

From a probabilistic aspect, the objective function in Eq.(3.4) assumes independent and identical Gaussian errors. However, Hou et al. (2015) indicate that firms with market cap below the NYSE 20th percentile only contribute 3% of the value of the U.S. stock market. In addition, the feature of heavy tails of stock returns is well known and may bring a bias on the objective function. Moreover, this objective function is sensitive to extreme values.

To partially solve the above problems, we use the Huber robust objective function:

$$\mathcal{L}_H(\theta) = \frac{1}{T}\sum_{t=1}^{T}\frac{1}{N_t}\sum_{i=1}^{N_t}H(r_{i,t+1} - g(z_{i,t}; \theta); \xi), \tag{3.5}$$

where the Huber loss function is

$$H(s;\xi) = \begin{cases} s^2, & \text{if} \quad |s| \le \xi; \\ 2\xi|s| - \xi^2, & \text{if} \quad |s| > \xi. \end{cases}$$

Note that $H(\cdot)$ combines the squared loss and absolute loss functions. When the error is smaller than the hyperparameter ξ, it uses the squared loss function. Otherwise, it uses the absolute loss function adjusted by ξ. The hyperparameter ξ can be seen as a hurdle to decide whether the difference between the realized return and expected return is large enough. The hyperparameter ξ is decided using procedures discussed in Section 3.1.

3.4 The Penalized Linear Model

The large p problem causes inefficient and inconsistent estimation for the simple linear model. Penalized linear model helps to avoid the large p and overfitting problems with a shrinkage method in the linear model to solve the large p problem. The shrinkage method refers to shrink parameters by imposing a penalty for the size of parameters, and allows to shrink parameters of predictors with less information will to zero. We would like to investigate if regularization helps to improve the prediction of a model in our empirical studies.

3.4.1 The Objective Function

The penalized linear model uses the simple linear model in equation (3.3), but adds a penalty term into the objective function to regularize parameters associated with relatively less informative predictors.

$$\mathcal{L}(\theta;\cdot) = \underbrace{\mathcal{L}(\theta)}_{\text{Loss Function}} + \underbrace{\phi(\theta;\cdot)}_{\text{Penalty}}.$$

The elastic-net penalty function is

$$\phi(\theta;\lambda,\rho) = \lambda(1-\rho)\sum_{j=1}^{P}|\theta_j| + \frac{1}{2}\lambda\rho\sum_{j=1}^{P}\theta_j^2.$$

where λ and ρ are hyperparameters. Note that when $\rho = 0$, the penalized linear model reduces to the LASSO regression, which imposes the L_1 LASSO penalty $\sum_{j=1}^{P}|\theta_j|$ to the parameters. When $\rho = 1$, the penalized linear model reduces to the ridge regression, which imposes the L_2 penalty for the parameters.

3.5 Fama-MacBeth regression

We follow Lewellen (2015) that uses parameters (including the intercept and coefficients) to conduct the well-known Fama-MacBeth (FM) two-step regression to predict future stock returns.

FM regression explains how the risk factors affect the asset returns, and helps to avoid the estimation error in the cross-sectional regression. The FM regression allows us to add a large set of factors when examining the relation between them with stock returns. The goal of the FM regression is to find the risk premium exposed to factors, and accesses how factors forecasts stocks returns.

The FM regression predicts the expected returns using lagged macroeconomic predictors and firm characteristics: lagged data is used to estimate the factor exposure of each risk factor in FM regression.

In the first step, we run a cross-sectional regression at time t:

$$r_{i,t+1} = r_{f,t} + \alpha_t + z'_{i,t}\theta_t + \epsilon_{i,t} \tag{3.6}$$

where $r_{i,t+1}$ is the return for the i-th firm at time $t+1$ period for $i = 1, 2, \ldots, N_t$ and $t = 1, 2, \ldots, T$. In addition, α_t is the intercept, $r_{f,t}$ is the risk-free rate at time t, $z_{i,t}$ is the risk factors of the ith stock at time t. The estimated coefficients are denoted by $\hat{\alpha}_t$ and $\hat{\theta}_t$ (also called as "factor exposure").

In the second step, we compute the averages of time series of intercept and coefficients by

$$\begin{aligned} \bar{\alpha} &= \frac{1}{T}\sum_{t=1}^{T}\hat{\alpha}_t, \\ \bar{\theta} &= \frac{1}{T}\sum_{t=1}^{T}\hat{\theta}_t. \end{aligned}$$

Then we predict the stock return for the i-th firm at time $t+1$ using the average factor exposure by

$$\hat{r}_{i,t} = \bar{\alpha} + z'_{i,t}\bar{\theta}$$

4 Machine Learning Techniques

4.1 Regression Tree

Regression tree considers a special type of nonlinear function in Equation (3.1). Regression trees use a tree to represent the recursive partition among predictors. Each of the terminal nodes or leaves of the tree represents a cell of the partition.

To figure out which cell each observation is in, we will ask a sequence of binary questions about the feature at the root node of the tree. Each binary question refers to a single attribute and has a yes or no answer.

The interior nodes are labeled with questions, and the edges or branches between them are labeled by the answers. Then, a new ”branch” continues partitioning the leftover samples from last spaces into the next sub-division until we get ideal space to fit all of samples.

In the tree-based models, we will take an advantage of them to deal with features interaction and forecast stock returns. Each of node of the tree represent a partition, and model will automatically decide partition on the splitting feature and split values by its loss function like minimum sum of squares. Then, each sample will belong to a specific partition if it corresponds to the splitting condition. In this process, we will ask a sequence of questions which will depend on the answers to the previous ones about the features of samples and assign them to their corresponding regions.

4.1.1 The Model

The ideal model is required to distinguish among the potential outcomes in term of minimum forecast errors. A regression tree with K terminal nodes (leaves) and depth L can be formally expressed as

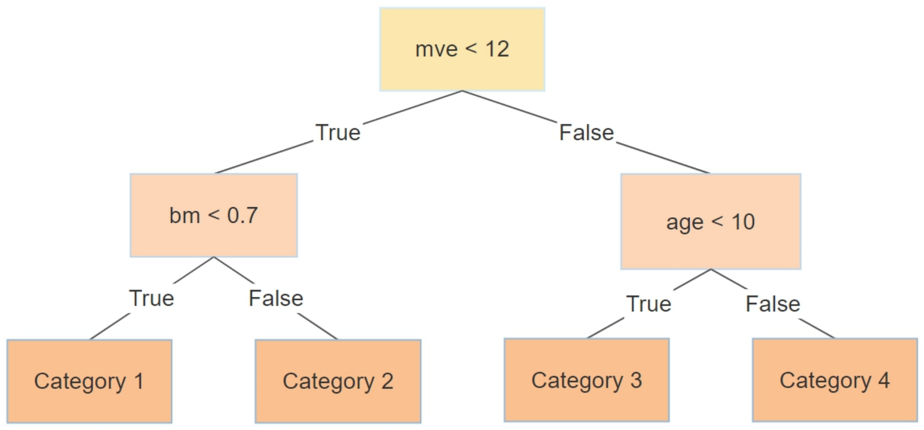
$$g(z_{i,t};\theta,L)=\sum_{k=1}^{2^L}\theta^{(k)}\mathbf{I}\{z_{i,t}\in C_k(L)\}, \tag{4.1}$$

where $C_k(L)$ is a one of the K partitions of the data, the constant $\theta^{(k)}$ iis defined to be the average of outcomes $r_{i,t+1}$ in partition k, and L is depth of the tree.

For example, a observation can fall into 4 specific partitions with questions about "$mve <$ 12", "$b/m < 0.7$", and "$age < 10$" in Figure 1. The output of regression tree model can be written as:

$$\begin{aligned} g(z_{i,t};\theta,4,2) = \; & \theta_1\mathbf{I}_{\{mve_{i,t}<12\}}\mathbf{I}_{\{b/m_{i,t}<0.7\}}+\theta_2\mathbf{I}_{\{mve_{i,t}<12\}}\mathbf{I}_{\{b/m_{i,t}\geq 0.7\}} \\ & +\theta_3\mathbf{I}_{\{mve_{i,t}\geq 12\}}\mathbf{I}_{\{age_{i,t}<10\}}+\theta_4\mathbf{I}_{\{mve_{i,t}\geq 12\}}\mathbf{I}_{\{age_{i,t}<10\}} \end{aligned}$$

Figure 1: Example of Regression Tree Structure



4.1.2 The Objective Function and Computational Algorithm

We choose the most popular l_2 loss function associated with the forecast error for each branch $C_k(L)$ called "impurity" for the model. The parameters $\Theta = \{\theta^{(k)}, C_k(L)\}_k^{2^L}$ are found by minimizing the objective function

$$\mathcal{L}(\Theta)=\arg\min_{\Theta}\sum_{k=1}^{2^L}\sum_{z_{i,t}\in C_k(L)}\mathcal{L}(r_{i,t+1},\theta^{(k)}). \tag{4.2}$$

This is a formidable optimization problem, and we usually settle for approximate sub-optimal solutions. It is useful to divide the optimization problem into two parts:

1. Finding $\theta^{(k)}$ given $C_k(L)$: It is trivial that $\hat{\theta}^{(k)} = \bar{r}_{i,t+1}$, the average value of the $r_{i,t+1}$ falling in region $C_k(L)$.

2. Finding $C_k(L)$: This is the most difficult part. A typical method is to use greedy, top-down recursive partitioning algorithm to find the region $C_k(L)$. It means that we will run all of potential factors to decide which is the best partitioned bound.

In this process, we try to find a idea binary question which locally minimizes the impurity and get root node and two daughter nodes in each branch until reaching certain conditions. The conditions can be that all observations have been assigned into their own partitions, the number of leaves or the depth of tree reach a pre-specified setting, or the model won't get more information if attributing a new branch. Usually, we will control regression tree with the pre-specific number of leaves and the depth of tree which are decided by tuning in validation samples.

Although the advantages of tree model can help us analyze data with complicated features interaction more easily, they have also their limitation. On of drawbacks of regression tree is that we often obtain a high-variance result in testing data since regression trees tend to be deep and complicated for fitting the extreme values if we don't control the growth of them. Model will keep producing new nodes and create too many branches for fitting the extreme values (outliers). This will cause poor prediction accuracy for out-of-sample data. Therefore, we try to improve the model performance by using ”ensemble” methods that combine forecast results from a set of small trees into a single prediction value instead of constructing a complex and deep tree.

4.2 Random Forest

The main idea of ensemble methods (bagging and boosting) is that using a combination of simple models and aggregate them to solve the problems, which can improve the performance. We get prediction results by estimating with several simple models which are mutually independent first. Then, aggregate all of prediction into only one result by using average, voting, or other methods. By doing so, we can avoid the overfiting problem and improve prediction performance rather than constructing a single complicated model.

Random forest model is one kind of application of bootstrap or bagging aggregation. Its basic idea is that using the many different regression trees to fit the bootstrap samples from training data (randomly select samples from the data set), and averaging the prediction results for reducing the prediction errors. In particular, each tree will use different features as splitting criteria because they uses different subsample. This helps us construct a set of relatively lower-correlated trees and avoiding to produce lots of duplicate trees. Finally, we take an average of these predictions and thus improve the predictive performance.

Using different bootstrap samples still has its limitation. For example, it is a double-edged that each tree will select features which are suitable for itself according to different bootstrap samples and thus can involve different features in their structures. However, we can't observe and control the internal condition of each tree. For example, it is well known that "size" is a dominated factor for asset pricing model. This causes most of bagging trees will split on firm size and increase the correlation among the their prediction. To more specific, we assume that the average of B bagging trees identically distribute a random variable, where each with variance σ and positive pairwise correlation ρ, so we can get the variance of the average of trees.

$$\rho\sigma^2 + \frac{1-\rho}{B}\sigma^2$$

We can observe the second term disappears as B increases, but the first term still remain which indicate high correlation will limit the benefits from taking an average of bagging trees.

4.2.1 Model

Give a tree as:

$$\hat{g}_b(z_{i,t}; \hat{\theta}_b, L) = \sum_{k=1}^{2^L} \hat{\theta}_b^{(k)} \mathbf{I}_{\{z_{i,t} \in C_k(L)\}}$$

In random forest, we need to tune two parameters, the depth D of trees and the number of trees B in the validation samples. The random forest model will aggregate all of trees by the average of the outcomes of them.

$$\hat{g}(z_{i,t}; L, B) = \frac{1}{B}\sum_{b=1}^{B} \hat{g}_b(z_{i,t}; \hat{\theta}_b, L).$$

4.3 Gradient Boosting Regression Tree

Boosted method is analogous to bagging since boosted tree also consists of many simple regression trees. The basic idea of boosting method is that using a combination of a set of weak learners to construct a powerful ”committee”. The weak learner is defined as a model with little predictive power which perform slightly better than random guess. The purpose of boosting method is to sequentially apply the weak learner to repeatedly revise them from the estimation errors and then aggregate the results into one single prediction value.

The details of boosting trees refer to gradient boosting which using idea of additive training. That is, we remain the original model and then add a new model which takes forward stage-wise procedure to fit the prediction residuals from original model. By doing so, we can obtain less estimation biases by combining these two model together into an ensemble prediction of the outcome. The example of additive training process is in the below,

$$\begin{aligned}
\hat{r}_{i,t+1}^{(0)} &= 0 \\
\hat{r}_{i,t+1}^{(1)} &= \hat{r}_{i,t+1}^{(0)} + g_1(z_{i,t}) = 0 + g_1(z_{i,t}) \\
\hat{r}_{i,t+1}^{(2)} &= \hat{r}_{i,t+1}^{(1)} + g_2(z_{i,t}) = g_1(z_{i,t}) + g_2(z_{i,t}) \\
&\dots \\
\hat{r}_{i,t+1}^{(b)} &= \hat{r}_{i,t+1}^{(b-1)} + g_b(z_{i,t}) = \sum_{b=1}^{B} g_b
\end{aligned}$$

At each step in the forward stage-wise procedure one must solve

$$\hat{\Theta}_b = \arg\min_{\Theta_b} \sum_{t=1}^{T}\sum_{i=1}^{N} \mathcal{L}(r_{i,t+1}, g_{b-1}(\cdot) + g_b(\cdot; \Theta_b)) \tag{4.3}$$

for the region set and constants $\Theta_b = \{\theta^{(k)}, C_k(L)\}_1^{2^L}$ of the next tree, given the current model $g_{b-1}(\cdot)$. Given the regions $C_k(L)$ in each tree, we can find the optimal constants $\theta_b^{(k)}$ in each region is equal to the mean of current residuals $r_{i,t+1} - g_{b-1}(\cdot)$ in each corresponding region.

In fact, the weak learner become overfitting when fitting the training data, so it usually needs to choose pre-specific parameter L to control the depth of each weak learner in the boosting procedure. Besides, we not only control the depth of each weak learner (L) but also add a another shrinkage factor into the second step, ν, as regularization strategies (also known as learning rate). This implementation of shrinkage factor, $\nu \in (0, 1]$, is applied to scale the contribution of each tree when it is added to the approximation which fits the residuals from $b - 1$ step. Empirically it has been found that smaller values of ν and correspondingly larger values of depth L make model perform better in testing data.

4.3.1 Model

The boosted tree model is sum of trees,

$$g(z_{i,t}; B, v, L) = \sum_{b=1}^{B} v\hat{g}_b(\cdot). \tag{4.4}$$

Our boosted trees have three tuning parameters (L, ν, B), and we will choose the best ones in validation step.

4.4 XGBoost

In Chen and Guestrin (2016), they purpose a novel tree-based algorithm, XGBoost. XG-Boost is improved from boosting ensemble method, for instance, it considers the regularization terms in the objective function for avoiding the overfitting problem in contract to gradient boosting regression tree (GBRT). Recently, XGBoost has became a popular method in machine learning field and been demonstrated that it can solve a wide range of problem in particularly. We can take the data mining competitions hosted by Kaggle for examples, these competition topics relate with many issues in real world such as house price prediction, credit card default prediction, or product categorization classification. Many of the winners use solely XGBoost or combine the XGBoost with neural networks in ensemble to train the models and provide solutions eventually.

We start XGBoost model from reviewing equation (4.4) in GBRT since they both are the applications of gradient boosting tree-based ensemble method. There are two largest difference points between XGBoost and GBRT. First, we will add weights to each leaf in XGBoost, which represent the specific influence on individual leaf unlike using the same score on individual leaf in the normal regression tree. Secondly, we consider a regularization term in the objective function of XGBoost to control the complexity of the model, which helps us to avoid overfitting.

4.4.1 Model

Given B tree in XGBoost model,

$$\hat{r}_{i,t+1} = g(z_{i,t}; B, v, L, \gamma, \lambda) = \sum_{b=1}^{B} v g_b(\cdot), g_b \in \mathcal{F},$$

where $\mathcal{F} = \{g(z_{i,t}) = w_{q(z_{i,t})}\}(q : \mathbb{R}^P \to M, w \in \mathbb{R}^M)$ is the space of regression trees. M is the number of leaves in tree, and each $w_q(z_{i,t})$ represents the weight of corresponding q-th leaf in each independent tree structure.

$$\mathcal{L}(\Theta) = \sum_{i=1}^{N} \sum_{t=1}^{T} l(r_{i,t+1}, \hat{r_{i,t+1}}) + \sum_{b=1}^{B} \Omega$$
$$\text{where } \Omega(g_b) = \gamma M + \frac{1}{2}\lambda ||w||^2$$

where l means a differentiable convex loss function, which we use mean squared error (MSE) here. The second term Ω is used to penalize the complexity of model structure and makes model simpler and smoother. When we set all of the regularization parameters to zero, the objective function will become the same as traditional GBRT.

Besides, since our objective function is differentiable, we can directly optimize objective function at each step by using second-order approximation during the additive training process.

This special property can also help us to derive how good the model is given the tree structure.

$$\begin{aligned}
\mathcal{L}^{(b)} &= \sum_{i=1}^{N}\sum_{t=1}^{T} l(r_{i,t+1}, \hat{r}^{(b)}_{i,t+1}) + \Omega(g_i) \\
&= \sum_{i=1}^{N}\sum_{t=1}^{T} l(r_{i,t+1}, \hat{r}^{(b-1)}_{i,t+1} + g_b(z_{i,t})) + \Omega(g_b) \\
&\simeq \sum_{i=1}^{N}\sum_{t=1}^{T} [l(r_{i,t+1}, \hat{r}^{(b-1)}_{i,t+1}) + f_i g_b(z_{i,t}) + \frac{1}{2} s_i g_b^2(z_{i,t})] + \Omega(g_b)
\end{aligned}$$

where f_i and s_i are defined as $f_i = \partial_{\hat{r}^{(b-1)}_{i,t+1}} l(r_{i,t+1}, \hat{r}^{(b-1)}_{i,t+1})$ and $s_i = \partial^2_{\hat{r}^{(b-1)}_{i,t+1}} l(r_{i,t+1}, \hat{r}^{(b-1)}_{i,t+1})$, respectively. Then, we continue to expand $\Omega(g_b)$ and simplify the objective by removing the constants.

$$\mathcal{L}^{(b)} = \sum_{i=1}^{N}\sum_{t=1}^{T} [f_i g_b(z_{i,t}) + \frac{1}{2} s_i g_b^2(z_{i,t})] + rM + \frac{1}{2}\lambda \sum_{j=1}^{M} w_j^2$$

Define $\mathbf{I}_j = \{i | q(z_{i,t}) = j\}$ as the instance set of j-th leaf, and we can rewrite the objective function.

$$\mathcal{L}^{(b)} = \sum_{j=1}^{M} [(\sum_{i\in\mathbf{I}_j} f_i) w_j + \frac{1}{2}(\sum_{i\in\mathbf{I}_j} s_i + \lambda) w_j^2] + \gamma M$$

For a fixed structure tree, we could obtain the optimal w_j^*

$$w_j^* = -\frac{\sum_{i\in\mathbf{I}_j} f_i}{\sum_{i\in\mathbf{I}_j} s_i + \lambda}$$

Then, we calculate the corresponding optimal value by

$$\mathcal{L}^{(b)}(q) = -\frac{1}{2}\sum_{j=1}^{M} \frac{(\sum_{i\in\mathbf{I}_j} f_i)^2}{\sum_{i\in\mathbf{I}_j} s_i + \lambda} + \gamma M$$

The above equation can be used as a scoring function to measure the quality of tree structure q.

4.5 Neural Networks

Neural network has been one of the most well-known and powerful machine learning techniques for prediction. Feng et al. (2019) and Gu et al. (2020) provide deep learning neural network for cross-section and time-series stock returns prediction using a factor model.

To determine the architecture of a neural network, two main hyperparameters need to be determined: the number of hidden layers and the number of neurons in each hidden layer. The number of hidden layers controls the depth of the network. A “deep” neural network has a large number of hidden layers, whereas a “shallow” neural network has a small number of hidden layers. Neurons in the hidden layer connect input variables and contribute to output variables through activation functions.

We apply the traditional ”feed-forward” neural network. Our network consists of an input layer of firm characteristics and macroeconomic factors, several hidden layers of pricing parameters, and an output layer of predicted stock returns. The model will nonlinearly transform the factors among the hidden layers and aggregate the information of the previous hidden layers to predict stock returns in the output layer. The architecture of neural network will influence the prediction results a lot. To be more specific, if the network is too ”shallow”, the results wouldn’t fit the data well. In constrast, if using a ”deep” network, the results would be overfitting, and we will obtain bad results as well. Therefore, it is the most important thing to decide the architecture of neural network when using it to solve prediction problems.

These two hyperparameters determine the complexity of the model. However, there is no known theory providing guidline to decide the optimal architecture. In general, we conduct a series of experiments to determine the optimal architecture.

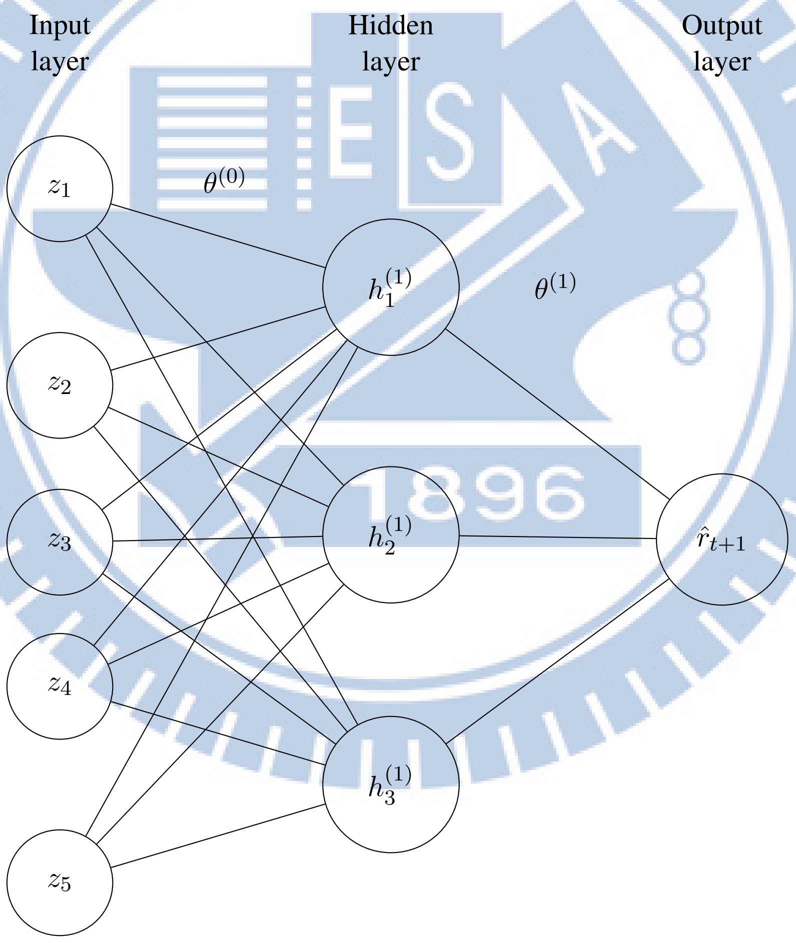
There are various types of activation functions, including sigmoid, softmax, and rectified linear unit function (ReLU), etc. We use ReLU as our activation function because it could overcome the vanishing gradient problem and allows the neural network to be learned faster. The ReLU activation function is defined as follows:

$$\text{ReLU}(h) = \begin{cases} 0, & \text{if } h < 0. \\ h, & \text{otherwise.} \end{cases}$$

Figure 2 provides an example of a simple neural network architecture with only one hidden layer. There are 5 predictors in the input layer (denoted $z_1, \ldots, z_5$). Each of the predictor signals will be amplified or attenuated by multiplying a four-dimensional parameter vector, θ, which includes an intercept and one weight parameter per predictor. Each neuron draws information linearly from all of the input units which applies a nonlinearly "activation function" $ReLU$, to its aggregated signal before sending its output to the next layers. The second neuron in the hidden layer transforms input information into output as $h_2^{(1)} = \text{ReLU}(\theta_{2,0}^{(0)} + \sum_{j=1}^{5} z_j \theta_{2,j}^{(0)})$. Lastly, the results from each neuron are linearly aggregated into ultimate output forecast:

$$g(z;\theta) = \theta_0^{(1)} + \sum_{j=1}^{3} h_j^{(1)} \theta_j^{(1)}.$$

Figure 2: Example of Neural Network Architecture



In order to decide the best hyperparameters, we use a variety of architectures of networks with up to totally five hidden layers. The simplest network contains a single hidden layer of 32 neurons denoted NN1, and NN2 has 2 hidden layers with 32 and 16 neurons, respectively. Sequentially, NN5 has five hidden layers with 32, 16, 8, 4, and 2 neurons, respectively. After comparing the performance of these models, we can determine the depth of network for forecasting returns.

4.5.1 Model Structure

The recursive output formulation for each neuron in our neural network:

$$h_k^{(l)} = \text{ReLU}(h^{(l-1)'}\theta_k^{(l-1)}), \tag{4.5}$$

with final output

$$g(z;\theta) = h^{(L-1)'}\theta^{(L-1)}. \tag{4.6}$$

Define $K^{(l)}$ as the number of neurons in each layer $l = 1,\dots,L$ and $h_k^{(l)}$ as the output of neuron k ($k = 1,\dots,K$) in layer l ($l = 1,\dots,L$). Then, we define the output vector of l-th layer as $h^{(l)} = (h_1^{(l)},\dots,h_k^{(l)})$, which includes an intercept and the predictors. Thus, we can recursively initialize the network by setting the input layer as $h(0) = (1, z_1,\dots,z_N)'$.

4.5.2 The Objective Function and Optimization Algorithm

We choose MSE as the objective function to estimate the weight parameters. Due to the lack of linearity and convexity in the objective function of a neural network, it is challenging to implement the optimization procedure to obtain the estimated weights parameters. In principle, the gradient descent algorithm is used in the optimization procedure for most of machine learning techniques. In the training process, the weight of the network network is updated to iteratively minimize the objective function. First, we calculate what the direction of change in each weight should do to reduce the loss function by using partial derivatives. Next, adjust a small amount of value of each weight parameters according to the its gradient.

The major challenge in neural network is how to select the hyperparameters (the number of hidden layers and the number of neurons in each layers), activation function, and optimization function to train the model. In our model, we use Adam (adaptive moment estimation) to train our neural network. Adam is different from the classic stochastic gradient descent (SGD) optimization algorithm. SGD only maintains a single learning rate to controls the step size of descend of gradients in the model. This causes SGD usually produces noises which dominate the direction of gradient and cause the loss function of model can't converge after training iteratively. As a result, it is necessary to gradually shrink the learning rate toward zero as the gradient gets closer with zero. Adam computes individual adaptive learning rates for different parameters from estimates of first and second moments of the gradients, and this can help us to control the learning rate in the training process. Besides, we also consider several regularization techniques in our model such as batch normalization and early stopping.

Batch normalization is a general technique that standardizes the inputs to a layer for each mini-batch. This can help us control the negative influence from the phenomenon of internal covariate shift, which will make the neural network to chase a moving target and cause the performance become worse. To be more specific, the weights of a layer are updated given an expectation that the prior layer outputs values are given a distribution. However, that distribution is likely changed after the weights of the prior layer are updated, and this will cause that the current layer needs to constantly readjust to new distributions. Eventually, the update procedure continue to chase a moving target. In order to stabilize the learning process, we standardize the inputs in each batch in our model.

Early stopping is a compromise regularization method to deal with the situation that model trains too much. Although we have already known that the number of training iteration (epochs) substantially influences the model, we are unable to directly choose a specific number of epochs before training the model. Early stopping allows us to use a large number of training epochs at the beginning of training process. Then, it will automatically stop training the model when the model performance has stopped improving on the validation dataset for several epochs. This avoid our model to overfit the training dataset.

Following Gu et al. (2020), we also adopt an ensemble approach in training our neural network to reduce the prediction variation. Random initial weights and the stochastic optimization lead to different weights inferred from the same data of the same neural network. The ensemble method uses M random seeds to generate initial weights, and produces M different weights. Then, the ensemble method predicts the value by taking average of predicted values using these M weights.

5 Empirical Analysis

5.1 Data descriptions

We obtain the monthly stock returns of all of firms on Center for Research in Security Prices (CRSP), which include data in the primary stock markets such as NYSE, AMEX and Nasdaq. The number of stocks is roughly a total of 18,000, and the average number of stocks exceeds 4,500 each month. Then, we follow Green et al. (2017) to use accounting data from CRSP, Compustat, and I/B/E/S to calculate the 102 firm characteristics. Moreover, we also collect 8 macroeconomic factors, as defined in Goyal and Welch (2008), including the dividend-price ratio (dp), earnings-price ratio (ep), book-to-market (bm), net equity expansion (ntis), Treasury-bill rate (tbl), term spread (tms), default spread (dfy), and stock variance (svar). We list all of variables in Table 1. Although we follow previous studies to calculate these variables, there remain differences compared with the original papers. In addition, we set one-month Treasury-bill rate as proxy for the risk-free rate to calculate the excess returns.

Our data covers a study period of 38 years from January 1980 to December 2017. While most firm characteristics are evaluated in monthly basis, the frequencies of the accounting data differ. For example, In the 102 firm characteristics, 63 of which are updated annually, 15 are updated quarterly, and 24 updated monthly. As a result, we need to synchronize these characteristics to monthly basis. Besides, because these data are released publicly with a delay, we make the following assumption to avoid the forward-looking bias. That is, if we want to predict returns at month $t+1$, we use the most recent monthly-updated firm characteristics at the end of month t, quarterly-updated data at the end of month $t-4$, and the annual-updated data at the end of month $t-6$.

Table 1: List of 8 macroeconomic predictors and 102 firm characteristics

Acronym	Definition	Author(s)	Year, Journal	Data Source	Frequency	Type
Macroeconomic predictors						
dp	Dividend Price Ratio (d/p) is the difference between the log of dividends and the log of prices.	Goyal and Welch	2008, RFS	Goyal's website	Monthly	Ratio
$ep\_GoyalWelch$	Earnings Price Ratio (e/p) is the difference between the log of earnings and the log of prices.	Goyal and Welch	2008, RFS	Goyal's website	Monthly	Ratio
$bm\_GoyalWelch$	Book-to-Market Ratio (b/m) is the ratio of book value to market value for the Dow Jones Industrial Average.	Goyal and Welch	2008, RFS	Goyal's website	Monthly	Ratio
$ntis$	Net Equity Expansion (ntis) is the ratio of 12-month moving sums of net issues by NYSE listed stocks divided by the total end-of-year market capitalization of NYSE stocks.	Goyal and Welch	2008, RFS	Goyal's website	Monthly	Ratio
tbl	Treasury Bills (tbl) rate is from the 3-Month Treasury Bill	Goyal and Welch	2008, RFS	Goyal's website	Monthly	Value
tms	The Term Spread (tms) is the dfference between the long term yield on government bonds and the T-bill.	Goyal and Welch	2008, RFS	Goyal's website	Monthly	Value
dfy	The default Yield Spread (dfy) is the difference between the BAA and AAA-rate corporate bond yield.	Goyal and Welch	2008, RFS	Goyal's website	Monthly	Value
$svar$	Stock Variance (svar): Stock Variance is computed as sum of squared daily returns on S&P 500.	Goyal and Welch	2008, RFS	Goyal's website	Monthly	Value
Firm characteristics						
$absacc$	Absolute value of acc	Bandyopad, Huang, and Wirjanto	2010, WP	Compustat	Annual	Value
acc	Annual income before extraordinary items minus operating cash flows divided by average total assets	Sloan	1996, TAR	Compustat	Annual	Ratio

$(continued)$

Table 1 – continued from previous page

Acronym	Definition	Author(s)	Year, Journal	Data Source	Frequency	Type
aeavol	Abnormal earnings announcement	Lerman, Livnat, and Mendenhall	2008, WP	Compustat+CRSP	Quarterly	Value
age	Number of years since first Compustat coverage	Jiang, Lee, and Zhang	2005, RAS	Compustat	Annual	Value
agr	Annual percent change in total assets	Cooper, Gulen, and Schill	2008, JF	Compustat	Annual	Ratio
baspread	Monthly average of daily bid-ask spread divided by average of daily spread	Amihud and Mendelson	1989, JF	CRSP	Monthly	Ratio
beta	Estimated market beta from weekly returns and equal weighted market returns for 3 years ending month $t-1$ with at least 52 weeks of returns	Fama and MacBeth	1973, JPE	CRSP	Monthly	Value
betasq	Market beta squared	Fama and MacBeth	1973, JPE	CRSP	Monthly	Value
bm	Book value of equity divided by end of fiscal year-end market capitalization	Rosenberg, Reid, and Lanstein	1985, JPM	Compustat+CRSP	Annual	Ratio
bm_ia	Industry adjusted book-tomarket ratio	Asness, Porter, and Stevens	2000, WP	Compustat+CRSP	Annual	Ratio
cash	Cash and Cash equivalents divided by average total assets	Palazzo	2012, JFE	Compustat	Annual	Ratio
cashdebt	Earnings before depreciation and extraordinary items divided by average total liabilities	Ou and Penman	1989, JAE	Compustat	Annual	Ratio
cashpr	Fiscal year-end market capitalization plus long-term debt minus total assets divided by cash and equivalents	Chandrashekar and Rao	2009, WP	Compustat	Annual	Ratio
cfp	Operating cash flows divided by fiscal-year-end market capitalization	Desai, Rajgopal, and Venkatachalam	2004, TAR	Compustat	Annual	Ratio
cfp_ia	Industry adjusted *cfp*	Asness, Porter, and Stevens	2000, WP	Compustat	Annual	Ratio
chatoia	2-digit SIC - fiscal-year mean-adjusted change in sales divided by average total assets	Soliman	2008, TAR	Compustat	Annual	Ratio
chcsho	Annual percent change in shares outstanding	Pontiff and Woodgate	2008, JF	Compustat	Annual	Ratio

(*continued*)

Table 1 – continued from previous page

Acronym	Definition	Author(s)	Year, Journal	Data Source	Frequency	Type
$chempia$	Industry-adjusted change in unmber of empolyees	Asness, Porter, and Stevens	1994, WP	Compustat	Annual	Value
$chfeps$	Mean analyst forecast in month prior to fiscal period end date from I/B/E/S summary file minus same mean forecast for prior fiscal period using annual earnings forecasts	Hawkins, Daniel, and Chamberlin	1984, FAJ	I/B/E/S	Annual	Value
$chinv$	Change in inventroy scaled by average total assets	Thomas and Zhang	2002, RAS	Compustat	Annual	Ratio
$chmom$	Cumulative returns from months $t-6$ to $t-1$ minus months $t-12$ to $t-7$	Titman, Wei, and Xie	2006, WP	CRSP	Monthly	Value
$chnanalyst$	Change in $nanalyst$ from month $t-3$ to month t	Scherbina	2008, RFS	I/B/E/S	Quarterly	Ratio
$chpmia$	2-digit SIC - fiscal-year mean-adjusted change in income before extraordinary items divided by sales	Soliman	2008, TAR	Computstat	Annual	Ratio
$chtx$	Percent change in total taxes from quarter $t-4$ to t	Thomas and Zhang	2011, JAR	Computstat	Quarterly	Ratio
$cinvest$	Change over one quarter in net PP&E divided by sales minus average of this variable for prior 3 quarters	Titman, Wei, and Xie	2004, JFQA	Compustat	Quarterly	Ratio
$convind$	An indicator equal to 1 if company has convertible debt obligations	Valta	2016, JFQA	Compustat	Annual	Indicator
$currat$	Current assets divided by current liabilities	Ou and Penman	1989, JAE	Compustat	Annual	Ratio
$depr$	Depreciation divided by PP&E	Holthausen and Larcker	1992, JAE	Compustat	Annual	Ratio
$disp$	Standard deviation of analyst forecasts in month prior to fiscal period end date divided by the absolute value of the mean forecast	Diether, Malloy, and Scherbina	2002, JF	I/B/E/S	Quarterly	Ratio
$divi$	An indicator equal to 1 if company pays dividends but did not in prior year	Michaely, Thaler, and Womack	1995, JF	Compustat	Annual	Indicator
$divo$	An indicator equal to 1 if company does not pay dividend but did in prior year	Michaely, Thaler, and Womack	1995, JF	Compustat	Annual	Indicator

(*continued*)

Table 1 – continued from previous page

Acronym	Definition	Author(s)	Year, Journal	Data Source	Frequency	Type
$dolvol$	Natural log of trading volume times price per share from month $t-2$	Chordia, Anshuman, and Subrahmanyam	2001, JFE	CRSP	Monthly	Value
dy	Total dividends divided by market capitalization at fiscal year-end	Litzenberger and Ramaswamy	1982, JF	Computstat	Annual	Ratio
ear	Sum of daily returns in three days around earnings announcement	Kishore et al.	2008, WP	Compustat+CRSP	Quarterly	Value
egr	Annual percent change in book value of equity	Richardson et al.	2005, JAE	Compustat	Annual	Ratio
ep	Annual income before extraordinary items divided by end of fiscal year market cap	Basu	1977, JF	Compustat	Annual	Ratio
$fgr5yr$	Most recently available analyst forecasted 5-year growth	Bauman and Dowen	1988, FAJ	I/B/E/S	Quarterly	Value
gma	Revenues minus cost of goods sold divided by lagged total assets	Novy-Marx	2013, JFE	Compustat	Annual	Ratio
$grCAPX$	Percent change in capital expenditures from year $t-2$ to year t	Anderson and Garcia-Feijoo	2006, JF	Compustat	Annual	Ratio
$grltnoa$	Growth in long-term	Fairfield, Whisenant, and Yohn	2003, TAR	Compustat	Annual	Value
$herf$	2-digit SIC - fiscal-year sales concentration	Hou and Robinson	2006, JF	Compustat	Annual	Value
$hire$	Percent change in number of employees	Bazdresch, Belo, and Lin	2014, JPE	Compustat	Annual	Ratio
$idiovol$	Standard deviationof residuals of weekly returns on weekly equal weighted market returns for 3 years prior to month end	Ali, Hwang, and Trombley	2003, JFE	CRSP	Monthly	Value
ill	Average of daily absolute return divided by dollar volume	Amihud and Mendelson	2002, JFM	CRSP	Monthly	Ratio
$indmom$	Equal weighted average industry 12-month returns	Moskowitz and Grinblatt	1999, JF	CRSP	Monthly	Value
$invest$	Annual change in gross property, plant, and equipment + annual change in inventories all scaled by lagged total assets	Chen and Zhang	2010, JF	Compustat	Annual	Ratio

(*continued*)

Table 1 – continued from previous page

Acronym	Definition	Author(s)	Year, Journal	Data Source	Frequency	Type
IPO	An indicator equal to 1 if first year available on CRSP monthly stock file	Loughran and Ritter	1995, JF	CRSP	Monthly	Indicator
lev	Total liabilities divided by fiscal year-end market capitalization	Bhandari	1988, JF	Compustat	Annual	Ratio
lgr	Annual percent change in total liabilities	Richardson et al.	2005, JAE	Compustat	Annual	Ratio
$maxret$	Maximum daily return from returns during calendar month $t-1$	Bali, Cakici, and Whitelaw	2011, JFE	CRSP	Monthly	Value
$mom12m$	11-month cumulative returns ending one month before month end	Jegadeesh	1990, JFE	CRSP	Monthly	Value
$mom1m$	1-month cumulative return	Jegadeesh and Titman	1993, JF	CRSP	Monthly	Value
$mom36m$	Cumulative returns from months $t-36$ to $t-13$	Jegadeesh and Titman	1993, JF	CRSP	Monthly	Value
$mom6m$	5-month cumulative returns ending one month before month end	Jegadeesh and Titman	1993, JF	CRSP	Monthly	Value
ms	Sum of 8 indicator variables for fundamental performance	Mohanram	2005, RAS	Compustat	Quarterly	Indicator
mve	Natural log of market capitalization at end of month $t-1$	Banz	1981, JFE	CRSP	Monthly	Value
mve_ia	2-digit SIC industry-adjusted fiscal year-end market capitalization	Asness, Porter, and Stevens	2000, WP	Compustat	Annual	Value
$nanalyst$	Number of analysts covering stock	Elgers, Lo, and Pfeiffer	2001, TAR	I/B/E/S	Quarterly	Value
$nincr$	Number of consecutive quarters (up to 8 quarters) with an increase in earning over same quarter in the prior year	Barth, Elliott, and Finn	1999, JAR	Compustat	Quarterly	Value
$operprof$	Revenue minus cost of goods sold - SG&A expense - interest expense divided by lagged common shareholders' equity	Fama and French	2015, JFE	Compustat	Annual	Ratio
$orgcap$	Capitalized SG&A expenses	Eisfeldt and Papanikolaou	2013, JF	Compustat	Annual	Value

(continued)

Table 1 – continued from previous page

Acronym	Definition	Author(s)	Year, Journal	Data Source	Frequency	Type
pchcapx_ia	2-digit SIC - fiscal-year mean-adjusted percent change in capital expenditures	Abarbanell and Bushee	1998, TAR	Compustat	Annual	Ratio
pchcurrat	Percent change in *currat*	Ou and penman	1989, JAE	Compustat	Annual	Ratio
pchdepr	Percent change in *depr*	Holtahausen and Larcker	1992, JAE	Compustat	Annual	Ratio
pchgm_pchsale	Percent change in gross margin munus percent change in sales	Abarbanell and Bushee	1998, TAR	Compustat	Annual	Ratio
pchquick	Percent change in *quick*	Ou and Penman	1989, JAE	Compustat	Annual	Ratio
pchsale_pchinvt	Annual percent change in sales minus annual percent change in inventory	Abarbanell and Bushee	1998, TAR	Compustat	Annual	Ratio
pchsale_pchrect	Annual percent change in sales minus annual percent change in receivables	Abarbanell and Bushee	1998, TAR	Compustat	Annual	Ratio
pchsale_pchxsga	Annual percent change in sales minus annual percent change in SG&A	Abarbanell and Bushee	1998, TAR	Compustat	Annual	Ratio
pchsaleinv	Percent change in *saleinv*	Ou and Penman	1989, JAE	Compustat	Annual	Ratio
pctacc	Same as *acc* except that numerator is divided by the absolute value of annual income before extraordinary items	Hafzalla, Lundholm, and Van Winkle	2011, TAR	Compustat	Annual	Ratio
pricedelay	The proportion of variation in weekly returns for 36 months ending in month t explained by 4 lags of weekly market returns incremental to contemporaneous market return	Hou and Moskowitz	2005, RFS	CRSP	Monthly	Value
ps	Sum of 9 indicator variables to form fundamental health score	Piotroski	2000, JAR	Compustat	Annual	Indicator
quick	(current assets - inventroy) / current liabilities	Ou and Penman	1989, JAE	Compustat	Annual	Ratio
rd	An indicator variable equal to 1 if R&D expense as a percentage of total assets has an increase greater than 5%	Eberhart, Maxwell, and Sideeique	2004, JF	Compustat	Annual	Indicator

(*continued*)

Table 1 – continued from previous page

Acronym	Definition	Author(s)	Year, Journal	Data Source	Frequency	Type
rd_mve	R&D expense divided by end-of-fiscal-year market capitalization	Guo, Lev, and Shi	2006, JBFA	Compustat	Annual	Ratio
rd_sale	R&D expense divided by sales	Guo, Lev, and Shi	2006, JBFA	Compustat	Annual	Ratio
realestate	Buildings and capitalized leases divided by gross PP&E	Tuzel	2010, RFS	Compustat	Annual	Ratio
retvol	Standard deviation of daily returns from month $t-1$	Ang et al.	2006, JF	CRSP	Monthly	Value
roaq	Income before extraordinary items divided by one quarter lagged total assets	Balakrishnan, Bartov, and Faurel	2010, JAE	Compustat	Quarterly	Ratio
roavol	Standard deviation for 16 quarters of income before extraordinary items divided by average total assets	Francis et al.	2004, TAR	Compustat	Quarterly	Ratio
roeq	Earnings before extraordinary items divided by lagged common shareholders' equity	Hou, Xue, and Zhang	2015, RFS	Compustat	Quarterly	Ratio
roic	Annual earnings before interest and taxes minus nonoperating income divided by non-cash enterprise value	Brown and Rowe	2007, WP	Compustat	Annual	Ratio
rsup	Sales from quarter t minus sales from quarter $t-4$ divided by fiscal-quarter-end market capitalization	Kama	2009, JBFA	Compustat	Quarterly	Ratio
salecash	Annual sales divided by cash and cash equivalents	Ou and Penman	1989, JAE	Compustat	Annual	Ratio
saleinv	Annual sales divided by total inventory	Ou and Penman	1989, JAE	Compustat	Annual	Ratio
salerec	Annual sales divided by accounts receivable	Ou and Penman	1989, JAE	Compustat	Annual	Ratio
secured	Total liability scaled secured debt	Valta	2016, JFQA	Compustat	Annual	Value
securedind	An indicator equal to 1 if company has secured debt obligations	Valta	2016, JFQA	Compustat	Annual	Indicator
sfe	Scaled earnings forecast	Elgers, Lo, and Pfeiffer	2001, TAR	I/B/E/S	Quarterly	Value
sgr	Annual percent change in sales	Lakonishok, Shleifer, and Vishny	1994, JF	Compustat	Annual	Ratio

(continued)

Table 1 – continued from previous page

Acronym	Definition	Author(s)	Year, Journal	Data Source	Frequency	Type
sin	An indicator equal to 1 if company's primary industry classification is in smoke or tobacco, beer, or alcohol, or gaming	Hong and Kacperczyk	2009, JFE	Compustat	Annual	Indicator
sp	Annual revenue divided by fiscal year-end market capitalization	Barbee, Mukherji, and Raines	1996, FAJ	Compustat	Annual	Ratio
std_dolvol	Monthly standard deviation of daily dollar trading volume	Chordia, Subrahmanyam, and Anshuman	2001, JFE	CRSP	Monthly	Value
std_turn	Monthly standard deviation of daily share turnover	Chordia, Subrahmanyam, and Anshuman	2001, JFE	CRSP	Monthly	Value
$stdacc$	Standard deviation for 16 quarters of accruals scaled by sales	Bandyopad hyay, Huang, and Wirjanto	2010, WP	Compustat	Quarterly	Value
$stdcf$	Standard deviation for 16quarters of cash flows divided by sales	Hung	2009, JEF	Compustat	Quarterly	Value
sue	Unexpeceted quarterly earning divided by fiscal-quarter-end market cap	Rendelman, Jones, and Latane	1982, JFE	I/B/E/S	Quarterly	Ratio
$tang$	Cash holdings + 0.715 $\times receivables + 0.547 \times inventroy + 0.535 \times (PPE/totalassets)$	Almeida and Campello	2007, RFS	Compustat	Annual	Value
tb	Tax income to book income	Lev and Nissim	2004, TAR	Compustat	Annual	Ratio
$turn$	Average monthly trading volume for most recent 3 months scaled by number of shares outstanding in current month	Datar, Naik, and Radcliffe	1998, JFM	CRSP	Monthly	Value
$zerotrade$	Turnover weighted number of zero trading days for most recent 1 month	Liu	2006, JFE	CRSP	Monthly	Value

[1]The 8 macroeconomic predictors, dp, $bm\_GoyalWelch$, tbl, dfy, $ep\_GoyalWelch$, $ntis$, tms, and $svar$ are computed according to the definition in Goyal and Welch (2008) to compute. These monthly data are available from Amit Goyal's website.

[2]The 102 firm characteristics are computed based on Green et al. (2017), and we use the SAS code from Jeremiah Green's website and extend the data periods to 2017.

Besides, because these data are released publicly with a delay, we make the following assumption to avoid the forward-looking bias. That is, if we want to predict returns at month $t + 1$, we use the most recent monthly-updated firm characteristics at the end of month t, quarterly-updated data at the end of month $t - 4$, and the annual-updated data at the end of month $t - 6$. Take quarterly-updated data as an example, these accounting characteristics are assumed to be known four months after the end of the fiscal year. Thus, if fiscal year of a firm ends in the prior December, its quarterly accounting data can be observable by the end of April of the next year.

We split the 38 years of data into training period of 25 years (1980/1 - 2004/12), the validation period of 6 years (2005/1 - 2010/12), and the out-of-sample period of the remaining 7 years (2011/1 - 2017/12). Recursively refitting for rolling window approach is avoided, because implementing machine learning is computationally intensive.

5.2 Data pre-processing - missing values imputation and normalization

The approach we take to retaining characteristic information is to first winsorize all characteristics at the 1st and 99th percentiles. In Table 2, we can observe that there are lots of missing values in most of firm characteristics and only about 7% of which have no missing values. It makes a strong negative impact on our analysis because those variables with substantial missing values might be important factors for forecasting returns. To reduce the estimation error, we impute the missing values month by month instead of discarding them directly. To impute the missing data, we first normalize the monthly data (except for indicator variables) excluding missing values by an individual feature, and plug the missing values with 0 in each month. By doing so, we complete data imputation and normalization.

Table 2: Data descriptive statistics of 8 macroeconomic predictors and 102 firm characteristics

Acronym	N	miss.(%)	Avg	std	min	med	max
Macroeconomic predictors							
dp	456	-	-3.72	0.41	-4.52	-3.85	-2.75
$ep\_GoyalWelch$	456	-	-2.94	0.45	-4.84	-2.94	-1.9
$bm\_GoyalWelch$	456	-	0.4	0.23	0.12	0.32	1.21

$(continued)$

Table 2 – continued from previous page

Acronym	N	miss.(%)	Avg	std	min	med	max
$ntis$	456	-	0	0.02	-0.06	0.01	0.05
tbl	456	-	0.04	0.04	0	0.05	0.16
tms	456	-	0.02	0.01	-0.04	0.02	0.05
dfy	456	-	0.01	0	0.01	0.01	0.03
$svar$	456	-	0	0.01	0	0	0.07
Firm characteristics							
$absacc$	1,775,701	13.34	0.09	0.1	0	0.06	1.14
acc	1,775,701	13.34	-0.03	0.13	-1.12	-0.03	0.52
$aeavol$	1,794,393	12.42	0.87	2.1	-1	0.29	22.52
age	2,048,944	0	11.18	9	1	9	43
agr	1,914,858	6.54	0.16	0.45	-0.71	0.07	6.05
$baspread$	2,012,384	1.78	0.06	0.07	0	0.04	0.93
$beta$	2,028,819	0.98	1.09	0.67	-0.82	1.01	3.97
$betasq$	2,028,819	0.98	1.64	1.9	0	1.03	15.77
bm	2,048,944	0	0.7	0.64	-2.57	0.56	7.64
bm_ia	2,048,944	0	27.32	779.89	-1307.01	0.04	17158.65
$cash$	1,818,998	11.22	0.16	0.2	-0.14	0.07	0.98
$cashdebt$	1,977,499	3.49	-0.04	1.34	-99.68	0.11	2.18
$cashpr$	2,028,268	1.01	-1.04	57.15	-537.25	-0.29	594.9
cfp	1,891,095	7.7	0.05	0.6	-4.43	0.05	24.35
cfp_ia	1,891,095	7.7	14.54	335.7	-296.6	0	7031.6
$chatoia$	1,764,963	13.86	0	0.24	-1.4	0	1.19
$chcsho$	1,914,092	6.58	0.11	0.32	-0.89	0.01	2.65
$chempia$	1,909,832	6.79	-0.12	0.68	-24.52	-0.07	3.8
$chfeps$	1,099,111	46.36	0	0.36	-13.64	0	10.21
$chinv$	1,865,800	8.94	0.01	0.06	-0.29	0	0.36
$chmom$	1,901,565	7.19	0	0.58	-9.09	-0.01	8.65
$chnanalyst$	1,575,363	23.11	-0.02	1.58	-45	0	43
$chpmia$	1,884,184	8.04	0.16	12.58	-548.23	-0.01	116.44
$chtx$	1,794,192	12.43	0	0.01	-0.12	0	0.14
$cinvest$	1,788,933	12.69	-0.02	0.97	-26.2	0	27.64
$convind$	2,048,944	0	0.12	0.33	0	0	1
$currat$	1,981,187	3.31	3.31	4.97	0.14	1.94	58.24
$depr$	1,960,419	4.32	0.28	0.41	-0.98	0.17	5.83
$disp$	912,932	55.44	0.15	0.41	0	0.04	9.97
$divi$	1,914,891	6.54	0.03	0.17	0	0	1
$orgcap$	1,507,419	26.43	0.01	0.01	0	0.01	0.09

(continued)

Table 2 – continued from previous page

Acronym	N	miss.(%)	Avg	std	min	med	max
pchcapx_ia	1,860,751	9.18	7.52	75.74	-237.44	-0.37	1629.63
pchcurrat	1,844,645	9.97	0.07	0.57	-0.89	-0.01	6.7
pchdepr	1,823,344	11.01	0.12	0.55	-0.86	0.03	7.82
pchgm_pchsale	1,888,403	7.84	-0.08	0.95	-12.29	0	4.77
pchquick	1,832,752	10.55	0.09	0.68	-0.92	-0.01	8.92
pchsale_pchinvt	1,508,643	26.37	-0.07	0.89	-11.57	0.02	3.96
pchsale_pchrect	1,835,471	10.42	-0.05	0.63	-7.69	0	3.38
pchsale_pchxsga	1,592,423	22.28	0.02	0.35	-1.45	0	4.68
pchsaleinv	1,489,019	27.33	0.17	1.13	-121.04	0.01	33.24
pctacc	1,775,689	13.34	-0.94	5.95	-64.75	-0.41	68.85
pricedelay	2,028,790	0.98	0.16	1.05	-15.61	0.07	13.47
ps	1,914,891	6.54	4.27	1.73	0	4	9
quick	1,970,271	3.84	2.61	4.41	0.09	1.31	53.93
rd	1,914,891	6.54	0.14	0.35	0	0	1
rd_mve	993,889	51.49	0.06	0.11	-0.03	0.03	2.25
rd_sale	977,364	52.3	0.71	5.69	-218.74	0.03	283.48
realestate	866,165	57.73	0.26	0.19	0	0.23	0.94
retvol	2,012,373	1.78	0.03	0.03	0	0.03	0.27
roaq	1,827,075	10.83	0	0.06	-0.51	0.01	0.14
roavol	1,551,046	24.3	0.03	0.05	0	0.01	0.85
divo	1,914,891	6.54	0.03	0.18	0	0	1
dolvol	1,979,178	3.4	11.33	3.04	-3.06	11.27	19.01
dy	2,043,740	0.25	0.02	0.03	-6.12	0	0.36
ear	1,806,061	11.85	0	0.08	-0.47	0	0.52
egr	1,914,703	6.55	0.14	0.72	-3.74	0.07	9.04
ep	2,048,944	0	-0.04	0.37	-8.22	0.04	0.44
fgr5yr	817,211	60.12	16.4	9.93	-43.5	14.6	85
gma	1,910,367	6.76	0.36	0.35	-0.96	0.31	1.77
grcapx	1,719,178	16.09	0.95	3.62	-12.85	0.12	60.64
grltnoa	1,441,881	29.63	0.09	0.16	-0.61	0.06	1.14
herf	2,048,934	0	0.08	0.09	0.01	0.05	1
hire	1,909,832	6.79	0.09	0.36	-0.77	0.02	4.22
idiovol	2,028,819	0.98	0.07	0.04	0	0.06	0.27
ill	1,953,829	4.64	0	0	0	0	0
indmom	2,048,787	0.01	0.14	0.3	-0.68	0.11	3.48
invest	1,850,781	9.67	0.08	0.18	-0.55	0.04	1.5
IPO	2,048,944	0	0.07	0.26	0	0	1

(continued)

Table 2 – continued from previous page

Acronym	N	miss.(%)	Avg	std	min	med	max
lev	2,043,342	0.27	2.24	4.81	0	0.65	75.63
lgr	1,908,518	6.85	0.26	0.82	-0.76	0.07	10.62
maxret	2,012,428	1.78	0.08	0.08	0	0.05	0.94
mom12m	1,901,565	7.19	0.13	0.61	-0.96	0.05	12.19
mom1m	2,048,944	0	0.01	0.16	-0.7	0	2.2
mom36m	1,600,305	21.9	0.33	0.96	-0.98	0.16	16.07
mom6m	1,989,131	2.92	0.05	0.38	-0.92	0.02	8.13
ms	1,830,211	10.68	3.78	1.68	0	4	8
mve	2,048,944	0	11.88	2.24	2.36	11.76	18.86
mve_ia	2,048,944	0	-202.19	6084.34	-15100.04	-472.12	109419.34
nanalyst	1,600,797	21.87	5.06	6.81	0	2	56
nincr	1,830,211	10.68	1	1.37	0	1	8
operprof	1,910,212	6.77	0.81	1.16	-8.58	0.62	10.51
roeq	1,826,834	10.84	0	0.15	-2.15	0.02	1.43
roic	1,963,488	4.17	-0.12	1.01	-20.32	0.06	0.91
rsup	1,816,448	11.35	0.01	0.2	-3.84	0.01	1.51
salecash	2,032,645	0.8	55.86	172.67	-1230.91	8.85	2505.12
saleinv	1,608,081	21.52	28.38	70.53	-35.44	8.11	979
salerec	1,976,064	3.56	11.52	59.23	-21796	5.82	241.87
secured	1,205,427	41.17	0.58	0.53	0	0.58	4.93
securedind	2,048,944	0	0.47	0.5	0	0	1
sfe	1,083,621	47.11	-0.22	2.66	-100.79	0.04	2.47
sgr	1,888,596	7.83	0.2	0.62	-0.87	0.09	8.73
sin	2,048,944	0	0.01	0.09	0	0	1
sp	2,043,482	0.27	2.04	3.26	-4.13	0.98	40.52
std_dolvol	1,950,573	4.8	0.88	0.42	0	0.82	2.86
std_turn	1,955,627	4.55	4.19	6.6	0	2.14	137.96
stdacc	1,290,891	37	4.44	31.54	0	0.13	714.21
stdcf	1,290,891	37	10.18	71.88	0	0.15	1665.7
sue	1,818,201	11.26	0	0.12	-5.3	0	1.76
tang	1,965,850	4.06	0.54	0.16	0	0.54	0.98
tb	1,804,442	11.93	-0.11	1.72	-27.7	-0.07	11.35
turn	1,979,639	3.38	1.1	1.44	0	0.6	66.48
zerotrade	1,953,855	4.64	1.4	3.4	0	0	19.95

To observe whether the multicollinearity problem exists, we measures the variance inflation factor (VIF) and the cross-correlations among our set of 102 firm characteristics and 8 macroeconomic factors. Multicollinearity will increase the standard errors in estimated parameters. Some characteristics are mechanically or economically related, for example, $roaq$ and $roeq$. If the absolute correlation between two variables excesses 0.2, the collinearitey problem affects the estimation.

Table 3 provides the summary statistics of VIFs and absolute cross-correlation among firm characteristics. There are 8 firm characteristics whose VIFs are larger than 10 (about 9%) and most of absolute cross-correlations (about 93%) are below 0.2. This indicates that the multicollinearity problem exists in our data. The simplest way to deal with multicollinearity problem is to remove variables highly correlated with others. However, because we would like to if machine learning is able to deal with the multicollinearity problem compared with the traditional methods, we include all of characteristics and macroeconomic factors throughout our analysis.

In fact, data normalization is a common requirement for many machine learning methods. The reason is that if a certain variable has a larger magnitude of variance than others or outliers exist in data set, they might dominate the objective function. Our model general form in equation (3.2) is a pooled model, and aggregates all information to produce the best parameters. It ignores that firm characteristics and macroeconomic predictors change over time. The average and standard deviation values of each factor also changes in time. If we use the original data without normalization, the estimation error is likely to increase. For example, tree-based model will automatically determine the splitting features and values during the training process. However, the values of firm characteristics and macroeconomic predictors in validation or testing periods change a lot and thus differ from the values in the training period. It will make our model unavailable to predict future stock returns. Our data pre-process approach allows us to eliminate the variations in the characteristics.

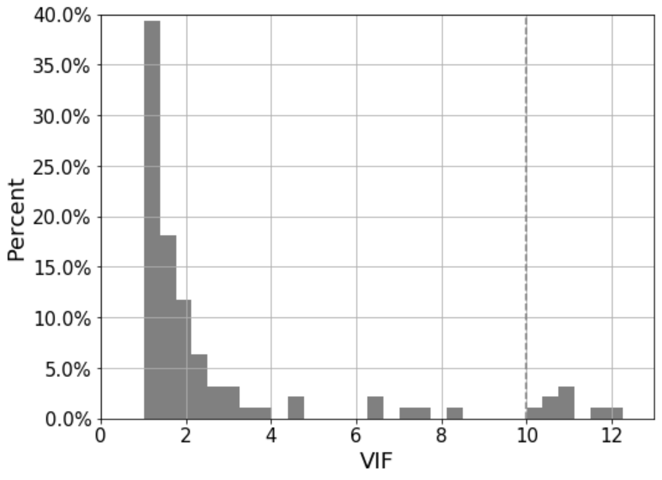
Figure 3 shows the box plots of autocorrelation function (ACF) at lags 1, 2, and 6 for each firm characteristic. It clearly shows that most of average ACF of characteristics excesses 0.25, and 80% of them are significantly different from 0 at the 5% level with Ljung-Box test. This indicates that serial correlations exists in most of the characteristics. Serial correlation will also cause estimation errors. However, solving serial correlation is out of the scope of our studies.

Table 3: Descriptive statistics on VIFs and cross-correlations among firm characteristics

A. Statistics of VIFs and cross-correlation

	Avg.	std	kurt	min	25%	50%	75%	max
VIFs	2.81	2.9	3.13	1.02	1.21	1.63	2.42	12.26
Absolute cross-correlations	0.06	0.09	20.42	0.00	0.01	0.03	0.08	0.95

B. Distribution of VIFs



c. Distribution of the absolute cross-correlations among characteristics.

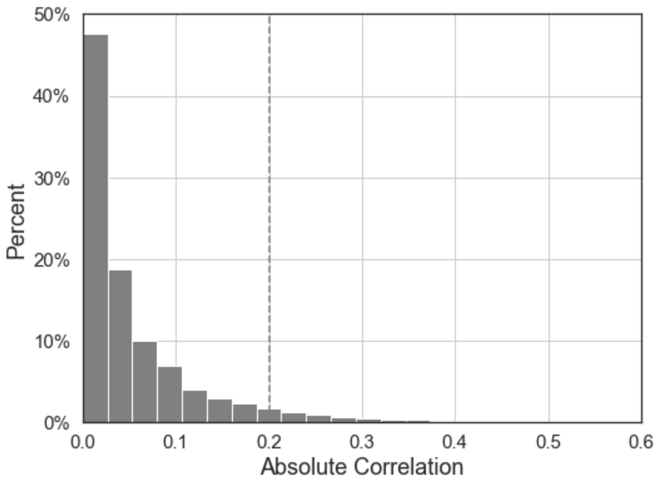
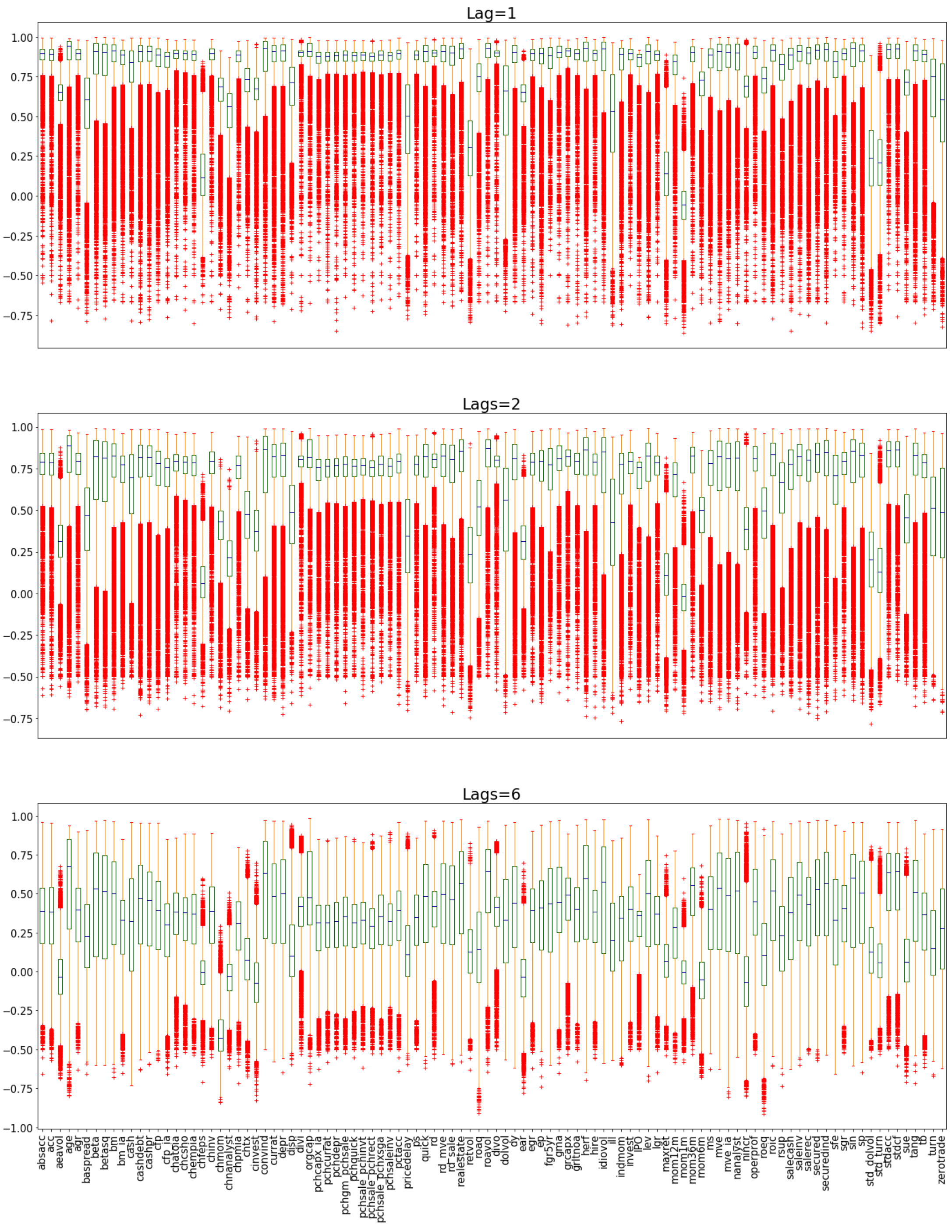


Figure 3: Firm Characteristics ACF Box Plots at Lags=1, 2, and 6



Note: The box plots show that the autocorrelation function (ACF) of each firm characteristics of lags 1, 2, and 6. They clearly show that strong serial correlation effect exist in most of predictors, and most of ACF of characteristics are significantly different from 0 at the 5% level with the Ljung-Box test.

5.3 The out-of-sample R^2 of cross-section stock returns

We calculate the out-of-sample R^2 to assess predictive performance for individual excess stock return forecasts.

$$R^2_{oos} = 1 - \frac{\sum_{i \in I, t \in \mathcal{T}} (r_{i,t+1} - \hat{r}_{i,t+1})^2}{\sum_{i \in I, t \in \mathcal{T}} r^2_{i,t+1}}$$

where I is the set of all firms and τ indicates the testing period. The denominator of the ratio of R^2_{oos} is the sum of squared returns and remains constant for all methods, whereas the numerator varies across methods. Specifically, the numerator of the ratio of R^2_{oos} measures the amount of squared prediction errors. A model of a higher R^2_{OOS} is preferred because it produces less prediction errors. As a result, the R^2_{OOS} is convenience measure that allows us to compare the performance of each method.

Table 4 compares a total of twelve models:OLS, Fama-MacBeth regression (FM), OLS with Huber loss function (Huber), elastic net (ENet), random forest (RF), gradient boosted regression trees (GBRT), XGBoost (XGB), and neural network architectures with one to five layers (denoted as NN1, . . . , NN5, respectively).

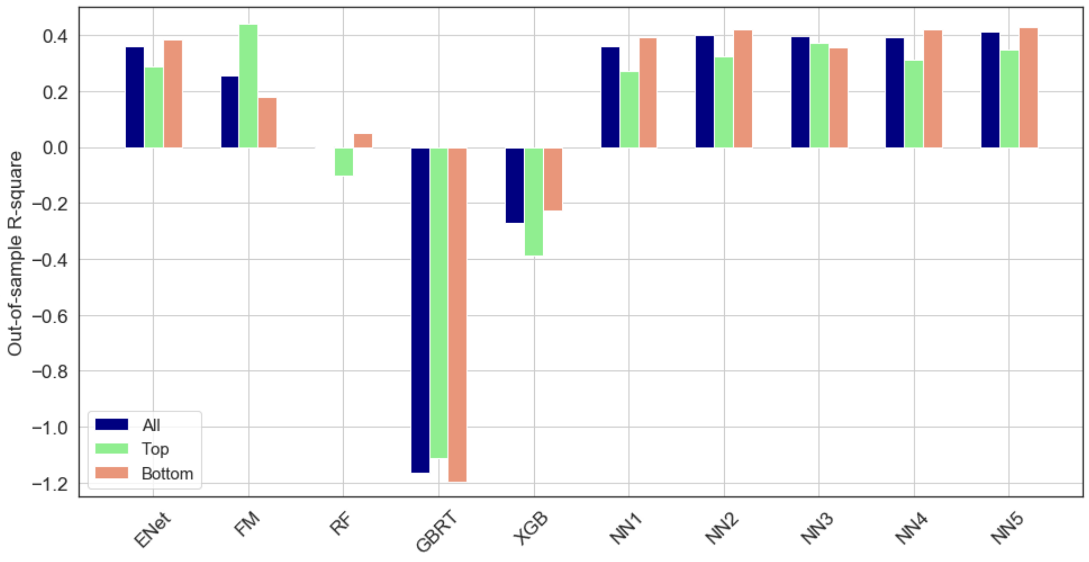
From the first row of Table 4, the OLS model produces an R^2_{oos} of -9.41%. This indicates that OLS fails to predict returns. It is actually as expected, because OLS yields highly susceptible fit in the in-sample period. Then, if we use Huber loss function to avoid the influence of the extreme outcome values in OLS, R^2_{oos} improves from -9.41% to -3.11%. Moreover, R^2_{oos} raises to 0.36% and 0.26% for ENet and FM, respectively. Indeed, ENet chooses a smaller set of variables but produces higher R^2_{oos}. This indicates that including all the predictors does not quarantine better prediction. .

Tree-Based methods (random forest, boosted trees, and XGB) seem to fail to forecast returns. R^2_{oos} of tree-based models are negative: 0.00%, -1.16%, and -0.27%. This is possibly because that the prediction outcomes are computed by the average $r_{i,t+1}$ of observation corresponding to leaves (terminal nodes) in training periods. Note that we only consider shallow trees. The number of leaves is possibly not large enough to predict returns. However, as will be shown later, they can distinguish the high-level and low-level future returns, even though they can't predict the return values directly.

Neural networks outperform all methods in predicting stock returns: R^2_{oos} is 0.36% for NN1

Table 4: Monthly Out-of-sample Stock-level Prediction Performance (Percentage R^2)

	OLS	Huber	ENet	FM	RF	GBRT	XGB	NN1	NN2	NN3	NN4	NN5
All	-9.41	-3.11	0.36	0.26	0.00	-1.16	-0.27	0.36	0.40	0.40	0.39	0.41
Top	-9.39	-2.75	0.29	0.44	-0.10	-1.11	-0.39	0.27	0.32	0.37	0.31	0.35
Bottom	-9.40	-3.16	0.39	0.18	0.05	-1.20	-0.23	0.39	0.42	0.36	0.42	0.43



Note: In this table, we report monthly R^2_{oos} for the return forecasts using OLS, OLS with Huber loss function (Huber), elastic net (ENet), Fama-MacBeth regression (FM), random forest (RF), gradient boosting regression tree (GBRT), XGBoost (XGB), and neural networks with one to five layers (NN1-NN5). We also report these R^2_{oos} within subsamples that include only the top 400 stocks or bottom 400 stocks by market values. The lower panel provides a visual comparison of the R^2_{oos} statistcs in the table (omitting OLS and Huber due to their large negative values.)

and reaches its peaks at 0.41% for NN5. However, we can observe that the marginal benefits from the deeper neural network become smaller. The results show that there is limitation for a deep neural network to improve the forecasts.

The second and third rows of Table 4 report the R^2_{oos} for large and small stocks (the top and bottom 400 by market values in each month). Note that we use the parameters estimated from the full data to forecasts returns for the two subsamples. We observe that FM dominates all of models among large stocks, with R^2_{oos} 0.44%.

5.4 The Bottom-Up Equal Weight Portfolio

Now, we evaluate the forecasting performance of traditional methods and machine learning techniques in forming a portfolio. There are a number of benefits to analyzing portfolio-level forecasts. First of all, portfolio forecasts provide an alternative to evaluate the prediction and robustness for each method. Second, most of investors practically tend to hold multiple assets to reduce market risk, so they usually construct portfolio or buy mutual fund in practice. Third, portfolio performance (like Sharpe ratio) is sensitive to the dependence among stock returns. Indeed, machine learning methods can build up a good stock-level forecasts model, but it is not guaranteed their predicted returns help to construct a portfolio with high returns. As a result, we construct a bottom-up equal-weight portfolio according the predicted returns of each stock. In adddtion, equal-weight portfolio forecasts can eliminate the prediction biases from each method.

To construct the bottom-up equal-weight portfolio, we calculate one-month-ahead predicted returns for each method in the end of each month. Then, we rank the stocks by their forecast returns and sort them into 10 deciles. Next, we construct equal-weight portfolio based on those deciles in last step. Finally, we can get 10 portfolios from the stocks with the highest expected returns (decile 10) to the stocks with lowest expected returns. For notional ease, we call the decile 10 portfolio as high (H) portfolio and decile 1 portfolio as low (L) portfolio. Given the weight of stock i in portfolio p (denoted $\frac{1}{N_{p,t}}$, $N_{p,t}$ is the number of stocks in the portfolio p at time t.) and a model-based out-of sample forecast for stock i (denoted $\hat{r}_{i,t+1}$), we construct the portfolio return forecast as

$$\hat{r}_{p,t+1} = \sum_{i=1}^{n} \frac{1}{N_{p,t}} \times \hat{r}_{i,t+1}$$

Table 5 reports the monthly portfolio-level of predictability R^2 over 7-year testing periods. From top to bottom, the first row presents the R^2_{oos} of samples among portfolio with lowest predicted return, and the last row shows R^2_{oos} among portfolio with highest predicted return. OLS, Huber, and GBRT show that their predictability (-1.99%, -1.28%, and -0.15%) are poor when they predict higher portfolio-level returns, but their predictability become much better when predicting low portfolio-level returns (-0.39%, -0.03%, and .05%). Also, we can observe the extreme predicted returns really mislead model results with increasing the loss in objective function, since the R^2_{oos} improves a lot from OLS to Huber. In contrast, machine learning methods (ENet, XGB, and neural network) perform better when predicting higher portfolio-level returns, and most of their R^2_{oos} are higher 0.10% than linear models. However, their predictability reverse when predicting lower portfolio-level returns, producing negative R^2_{oos}. In Table 5, we can know that machine learning are good at catching the trend with higher forecast returns, but their predictability in lower forecast returns may not outperform traditional linear models.

Table 5: Monthly Portfolio-level Out-of-Sample Predictive Performance (Percentage R^2)

	OLS	Huber	ENet	FM	RF	GBRT	XGB	NN1	NN2	NN3	NN4	NN5
Low (L)	-0.39	-0.03	-0.04	0.00	-0.03	-0.05	-0.11	-0.02	-0.02	0.00	-0.03	-0.03
2	-0.75	-0.07	0.02	0.01	0.03	-0.09	-0.07	0.01	0.01	0.02	0.01	0.00
3	-0.96	-0.18	0.02	0.05	0.07	-0.14	-0.03	0.03	0.03	0.03	0.03	0.01
4	-0.98	-0.25	0.05	0.06	0.00	-0.19	-0.01	0.06	0.06	0.06	0.04	0.03
5	-1.13	-0.38	0.07	0.07	0.01	-0.18	0.00	0.06	0.07	0.09	0.06	0.07
6	-1.17	-0.48	0.04	0.06	-0.03	-0.18	0.01	0.06	0.09	0.09	0.06	0.09
7	-1.22	-0.62	0.07	0.06	0.00	-0.17	-0.01	0.07	0.07	0.08	0.05	0.09
8	-1.47	-0.70	0.06	0.09	-0.02	-0.20	-0.05	0.05	0.05	0.05	0.07	0.13
9	-1.39	-0.90	0.05	0.02	-0.02	-0.15	-0.07	0.04	0.04	0.07	0.10	0.12
High (H)	-1.99	-1.28	0.12	-0.07	0.00	-0.15	0.11	0.09	0.10	0.01	0.16	0.15

Note: In this table, we report the R^2_{oos} for 10-decile portfolios using OLS, OLS with Huber loss function (Huber), elastic net (ENet), Fama-MacBeth regression (FM), random forest (RF), gradient boosting regression tree (GBRT), XGBoost (XGB), and neural networks with one to five layers (NN1-NN5). The 10 equal-weight portfolios of each models are constructed based on 10 deciles, which sort the stocks into 10 decile from high to low stock-level return forecasts. Low (L) is the equal-weight portfolio constructed by stocks with lowest returns forecasts. High (H) is the equal-weight portfolio constructed by stocks with highest returns forecasts.

The following measures help to evaluate the performance of a portfolio in various aspects. Let P_t denote the portfolio value at time t. The initial portfolio value is denoted as P_0. The last value is

$$P_T = 100 \times \prod_{t=1}^{T} e^{r_t^p},$$

where r_t^p is the realized portfolio-level return at time t, $t = 1, \cdots, T$. The compound annual growth rate (CAGR) is

$$\text{CAGR} = \frac{1}{(T/12)}(logP_T - logP_0).$$

The annualized standard deviation (SD) is

$$\text{SD} = \sqrt{E[(r_t^p - E(r_t^p))^2] \times 12}.$$

The Sharpe ratio (SR) is

$$\text{SR} = \frac{CAGR - r_f}{SD},$$

where r_f is the annualized risk-free rate. The turnover rate (TO) is

$$\text{TO} = \frac{1}{T}\sum_{t=2}^{T}(\sum_{i=1}^{N_T} |w_{i,t} - w_{i,t-1}|),$$

where $w_{i,t}$ is the weight of i-th stock at time t, for $i = 1, \cdots, N_T$. Table 6 reports the backtesting results by setting the initial value of a portfolio as 100. The equal-weight portfolio and the S&P 500 index ETF (SPY) are served as two benchmark portfolios.

The performance of these portfolios appear to be close to the performance of each method in terms R^2_{oos} in Section 5.3. First of all, for all methods except RF, the GAGR of the group portfolios increases monotonically as the group rank increases. This means that each method is able to predict portfolio-level returns. Another way to measure the model performance in building a group portfolio is to compare the difference in last value or CAGR between the high and low portfolios. When the difference is larger, the model can better predict stocks with higher or lower returns. The XGBoost and NN4 dominate all the other methods because they have the largest different in the last values (about 320).

Table 6: Performance of Bottom-Up Portfolios of Each Models

	Last Value	Pred.	CAGR	SD	SR	TO	Last Value	Pred.	CAGR	SD	SR	TO
			Equal-Weight						SPY			
Benchmark	225.60	-	0.12	0.14	0.81	0.02	244.87	-	0.13	0.11	1.18	-
			OLS						FM			
Low (L)	92	0.42	-0.01	0.21	-0.06	0.57	77	-0.01	-0.04	0.21	-0.17	0.61
2	158	0.55	0.07	0.16	0.40	1.09	162	0.00	0.07	0.16	0.43	1.12
3	198	0.60	0.10	0.14	0.68	1.28	226	0.01	0.12	0.14	0.81	1.29
4	243	0.64	0.13	0.14	0.90	1.36	236	0.01	0.12	0.14	0.89	1.37
5	248	0.67	0.13	0.14	0.95	1.38	247	0.02	0.13	0.13	0.96	1.40
6	255	0.69	0.13	0.14	0.96	1.37	255	0.02	0.13	0.14	0.95	1.39
7	291	0.72	0.15	0.14	1.12	1.34	267	0.02	0.14	0.14	1.02	1.35
8	277	0.76	0.15	0.14	1.07	1.25	323	0.02	0.17	0.14	1.18	1.28
9	319	0.80	0.17	0.14	1.16	1.09	281	0.03	0.15	0.14	1.05	1.10
High (H)	300	0.94	0.16	0.15	1.04	0.66	338	0.04	0.17	0.14	1.20	0.67
H-L	326	0.52	0.17	0.11	1.50	-	436	0.05	0.21	0.12	1.69	-
			Huber						ENet			
Low (L)	67	-0.07	-0.06	0.26	-0.23	0.43	108	0.15	0.01	0.19	0.06	0.24
2	153	0.21	0.06	0.21	0.30	0.90	200	0.16	0.10	0.17	0.58	0.51
3	189	0.33	0.09	0.17	0.55	1.10	205	0.16	0.10	0.15	0.66	0.71
4	248	0.40	0.13	0.15	0.86	1.23	232	0.16	0.12	0.14	0.83	0.82
5	260	0.45	0.14	0.14	0.97	1.30	260	0.16	0.14	0.14	0.96	0.85
6	290	0.49	0.15	0.13	1.19	1.34	228	0.16	0.12	0.15	0.80	0.84
7	288	0.52	0.15	0.12	1.23	1.34	271	0.16	0.14	0.14	0.98	0.79
8	314	0.56	0.16	0.12	1.35	1.28	250	0.17	0.13	0.15	0.90	0.70
9	316	0.59	0.16	0.12	1.42	1.14	227	0.17	0.12	0.13	0.92	0.60
High (H)	334	0.65	0.17	0.11	1.59	0.69	367	0.17	0.19	0.14	1.28	0.33
H-L	501	0.73	0.23	0.20	1.17	-	339	0.03	0.17	0.12	1.50	-
			RF						GBRT			
Low (L)	169	0.20	0.07	0.17	0.45	0.48	94	0.11	-0.01	0.26	-0.04	0.47
2	237	0.21	0.12	0.15	0.82	0.81	145	0.24	0.05	0.20	0.27	1.03
3	287	0.21	0.15	0.14	1.04	0.87	189	0.30	0.09	0.16	0.57	1.18
4	219	0.21	0.11	0.15	0.75	0.88	220	0.33	0.11	0.14	0.82	1.21
5	227	0.22	0.12	0.14	0.84	0.94	253	0.36	0.13	0.13	1.04	1.25
6	199	0.23	0.10	0.15	0.66	0.97	261	0.37	0.14	0.13	1.06	1.26
7	224	0.23	0.12	0.16	0.73	0.87	289	0.38	0.15	0.12	1.23	1.26
8	217	0.24	0.11	0.15	0.75	0.73	280	0.40	0.15	0.13	1.15	1.22
9	234	0.26	0.12	0.15	0.84	0.62	327	0.41	0.17	0.13	1.28	1.09
High (H)	263	0.28	0.14	0.16	0.89	0.34	349	0.43	0.18	0.13	1.39	0.69
H-L	156	0.08	0.06	0.11	0.60	-	373	0.31	0.19	0.19	0.97	-
			XGBoost						NN1			
Low (L)	89	0.23	-0.02	0.21	-0.08	0.35	117	0.09	-0.01	0.20	0.11	0.29
2	170	0.26	0.08	0.17	0.44	0.75	186	0.14	0.08	0.17	0.51	0.63
3	222	0.27	0.11	0.15	0.74	0.96	213	0.15	0.11	0.15	0.70	0.81
4	245	0.27	0.13	0.15	0.86	1.07	252	0.16	0.13	0.15	0.90	0.89
5	255	0.28	0.13	0.15	0.90	1.10	262	0.17	0.13	0.15	0.93	0.90
6	269	0.28	0.14	0.14	1.03	1.09	243	0.17	0.15	0.14	0.92	0.86
7	261	0.29	0.14	0.14	0.96	1.04	259	0.18	0.14	0.14	1.00	0.79
8	238	0.29	0.12	0.14	0.88	0.94	235	0.18	0.13	0.14	0.85	0.68
9	243	0.30	0.13	0.12	1.01	0.78	232	0.19	0.12	0.13	0.91	0.60
High (H)	407	0.33	0.20	0.13	1.55	0.40	326	0.23	0.18	0.15	1.10	0.31
H-L	458	0.10	0.22	0.13	1.68	-	278	0.14	0.19	0.14	1.04	-

(continued)

Table 6 – continued from previous page

	Last Value	Pred.	CAGR	SD	SR	TO	Last Value	Pred.	CAGR	SD	SR	TO
			NN2						NN3			
Low (L)	95	0.13	0.02	0.21	-0.03	0.29	85	0.11	-0.02	0.22	-0.11	0.44
2	179	0.15	0.09	0.18	0.45	0.66	173	0.14	0.09	0.17	0.47	0.90
3	215	0.16	0.11	0.15	0.73	0.84	205	0.16	0.11	0.15	0.69	1.09
4	245	0.16	0.13	0.14	0.88	0.90	238	0.16	0.12	0.14	0.89	1.16
5	255	0.17	0.14	0.14	0.97	0.93	281	0.17	0.14	0.13	1.10	1.19
6	282	0.17	0.13	0.13	1.10	0.91	267	0.17	0.13	0.13	1.08	1.17
7	260	0.17	0.14	0.14	0.96	0.83	280	0.17	0.13	0.14	1.08	1.11
8	248	0.18	0.12	0.14	0.90	0.71	253	0.17	0.13	0.14	0.95	1.01
9	230	0.19	0.12	0.12	0.96	0.62	289	0.19	0.14	0.13	1.12	0.85
High (H)	359	0.21	0.17	0.16	1.17	0.33	314	0.21	0.20	0.16	0.99	0.45
H-L	378	0.08	0.15	0.14	1.39	-	371	0.08	0.22	0.13	1.49	-
			NN4						NN5			
Low (L)	89	0.11	-0.02	0.21	-0.08	0.27	88	0.03	-0.03	0.22	-0.08	0.37
2	181	0.14	0.07	0.18	0.48	0.58	159	0.12	0.09	0.20	0.34	0.81
3	212	0.16	0.09	0.16	0.66	0.75	183	0.14	0.12	0.17	0.51	1.03
4	225	0.16	0.11	0.15	0.78	0.81	212	0.15	0.12	0.16	0.67	1.14
5	258	0.17	0.14	0.15	0.92	0.83	263	0.16	0.13	0.14	0.98	1.22
6	255	0.17	0.15	0.15	0.90	0.81	277	0.18	0.13	0.13	1.12	1.24
7	241	0.17	0.14	0.14	0.91	0.76	267	0.19	0.14	0.13	1.10	1.20
8	254	0.17	0.17	0.13	1.00	0.69	319	0.21	0.13	0.12	1.34	1.09
9	276	0.18	0.15	0.12	1.21	0.61	278	0.23	0.15	0.11	1.31	0.91
High (H)	405	0.19	0.19	0.13	1.50	0.32	368	0.27	0.18	0.13	1.44	0.53
H-L	457	0.08	0.20	0.14	1.50	-	418	0.24	0.21	0.14	1.48	-

Note: In this table, we report the performance of prediction-sorted portfolios over out-of-sample testing periods. All stocks are sorted into deciles based on their predicted returns for the next month. Within each decile of stocks, we use equal-weight to construct portfolio. We report last value (Last value), the predicted monthly portfolio-level returns (Pred.), compound annual growth rate (CAGR), annualized standard deviation (SD), Sharpe ratio (SR), and turnover rate (TO). Benchmarks are equal-weight portfolio with all of stocks (EW_all) and S&P 500 index ETF (SPY).

When only observing the last value for the high portfolio, the XGBoost and NN4 dominate all the other methods. Their last values are 407 and 405, respectively. However, machine learning techniques fail to gain lower last values for the low portfolio than traditional methods: The least two last values using the traditional methods are 67 and 77 using the OLS with Huber's objective and FM regression, respectively. However, the least three last values for the low portfolio by machine learning technique are 85, 88, and 89 by the NN3, NN5, and XGBoost, respectively. The observations are consistent with the results summarized in Table 5. It is found that machine learning techniques perform well in predicting stocks returns with higher returns ex post, but they perform comparatively with the traditional methods in predicting stock returns with lower returns ex post.

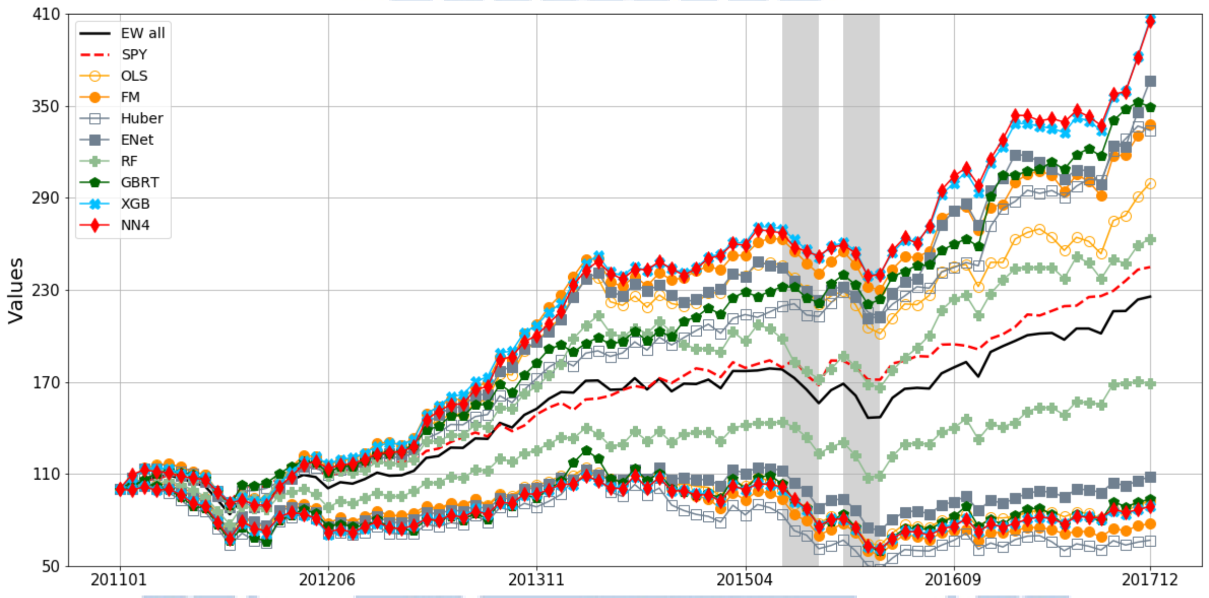
The Sharpe ratio calculates the additional amount of return per unit of increase in risk, and hence is a good measure to evaluate the performance of a portfolio in practice. A long-short portfolio consists a long position of the high portfolio and a short position of the low portfolio. When holding a long-short, FM and XGBoost yield the highest Sharpe ratios at 1.69 and 1.68, respectively. These Sharp ratios are substantially larger than the benchmark portfolios: Equal-weight portfolio and SPX, which yield Sharp ratios at 0.81 and 1.02, respectively. When FM produce worse R^2_{oos} as shown in Section 5.3, FM produces the most competitive Sharp ratio as the XGBoost. It is possibly because FM is able to identify stocks with both high and low returns ex post.

It is complicated to calculate transaction costs directly. For simplicity, we calculate the turnover rate as a proxy for the transaction cost. High and low portfolios in each method are smaller than other portfolios. Next, turnover rates in high and low portfolios of XGBoost and NN4 are smaller than those of traditional methods. It is surprising that, in contrast to the traditional methods such as OLS, FM and Huber, XGBoost and NN4 produce comparatively larger Sharp ratios, yet they also yield lower turnover rates.

Figure 4 visualizes Table 6 by plotting the values of the high and low portfolios over time, and include the equal-weight portfolio and S&P 500 index for comparison. The lines above equal-weight portfolio (solid black line) are the values of the high portfolios. On the contrary, the lines below the equal-weight portfolio are the values of the low portfolios. Also, it is intuitive that a method performs well if difference between the values of the high and low portfolio is large. On the other hand, our model performs poorly if the difference is small and if the values of the high and low portfolios are close to the equal-weight portfolio. For the high portfolios, it is observed that XGBoost and NN4 dominate the other methods, because their last values are 40 higher than the best traditional method, ENet. For the low portfolio, all of values are close to each other. The values of portfolio constructed by traditional methods are always lower than those by machine learning techniques. Random forest fails to predict portfolio-level returns, and its performs worst among all methods. As for the tree-based method, the boosting method (including GBRT and XGBoost) outperforms greatly the bagging method (random forest).

In addition, we report the maximum drawdown of each strategy in Table 7 and define maximum drawdown as

Figure 4: Cumulative Returns of Machine Learning Portfolios



Note: This figure visualizes cumulative returns of portfolios sorted on out-of-sample returns forecasts from all of models and benchmarks, including equal-weight portfolio with all of stocks (EW all), S&P 500 index (SPX), OLS, Fama-MacBeth regression (FM), OLS with Huber loss function (Huber), elastic net (ENet), random forest (RF), gradient boosting regression tree (GBRT), XGBoost (XGB), and neural networks with four layers (NN4). The black solid and the red dash lines are the benchmarks, equal-weight portfolio with all of stocks and S&P 500 index, respectively. The lines above benchmarks visualize the high portfolios, and the lines below benchmarks visualize the low portfolios. The gray parts are the periods with larger drawdow. (2015/7-2015/10 and 2015/12-2016/03.)

$$\text{MaxDD} = \max_{0 \le t_1 \le t_2 \le T} (P_{t_1} - P_{t_2})$$

where P_t is the value from period 0 through t. In the high portfolios, the highest maximum drawdown is XGBoost (16.17), and lowest maximum drawdown is Huber (-10.1). However, Huber has lower drawdown than other methods in all of bottom-up portfolios, it has highest maximum drawdown in long-and-short portfolio (-40.3). Besides, we can observe that the drawdown of the machine learning methods (XGB and NN4) are not too high in long-and short portfolios (-22 and -19.1, respectively). This indicates that machine learning methods provide us to obtain more economics gains, Sharpe ratio, without taking more risk.

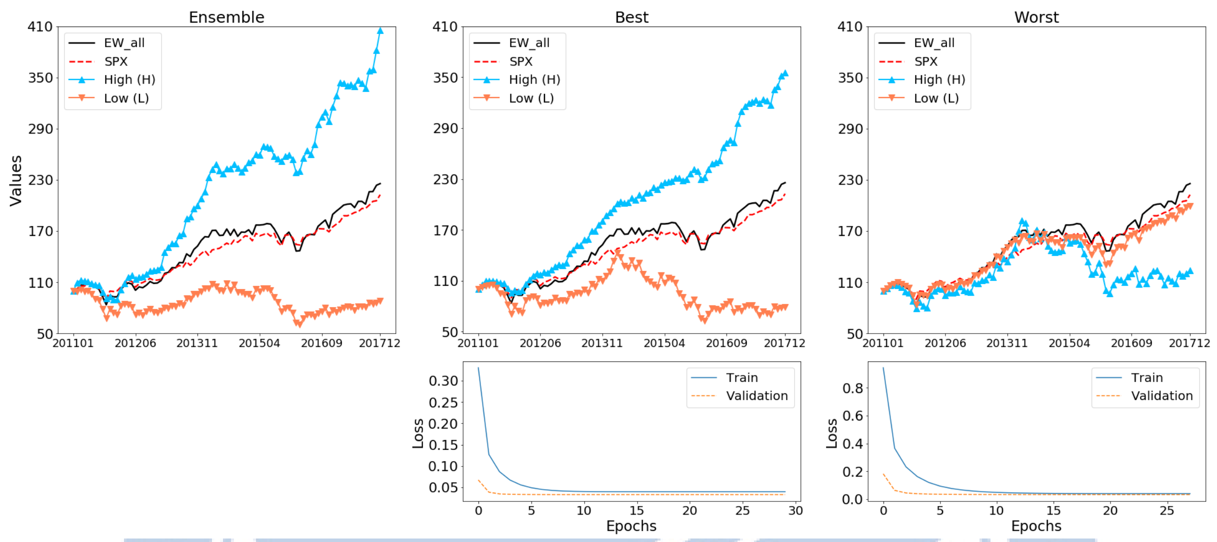
Table 7: Maximum Drawdown of each method

	OLS	FM	Huber	ENet	RF	GBRT	XGB	NN4
Low (L)	-11.8	-12.1	-11.1	-11.7	-14.1	-13.1	-11.1	-12.2
2	-14.9	-15.2	-18.1	-16.5	-14.4	-18.1	-16.9	-18.6
3	-15.5	-14.5	-17.4	-18.8	-18.2	-15.9	-17.6	-17.4
4	-15.5	-14.3	-16.5	-16.8	-14.8	-14.6	-16.3	-14.4
5	-13.8	-13.9	-12.6	-14.8	-12.4	-12.5	-15.4	-14
6	-12.6	-13.8	-12.5	-10.8	-14.1	-11.5	-13.3	-13.7
7	-15.1	-12.6	-13.2	-14.4	-16	-11.8	-12.8	-10.9
8	-13.4	-14.4	-10.7	-12.6	-15.3	-10.8	-9.7	-10.5
9	-15	-12.3	-12.7	-10.7	-13.6	-12.6	-10.5	-10.9
High (H)	-15.3	-15.4	-10.1	-14.7	-15.5	-12.8	-16.2	-15
H-L	-21.1	-24.9	-40.3	-13	-12	-31.2	-22	-19.1

5.5 Difficulties in training a Neural Network

Section 3.6 mentions that we use an ensemble approach in training a neural network to reduce the variations in weights inferring caused by the random initial weights and the stochastic optimization process. Figure 5 shows the cumulative portfolio returns performance in the testing period, and learning histories in the training and validation periods. We repeat to train a neural network of the same architecture (NN4) for thirty times to provide two extreme cases to illustrate the difficulty in training a neural network. Here, the ensemble portfolio (Ensemble) is the portfolio based on the average returns predicted in each trial. The best case and and worst case represent two extreme results among the thirty trials. In the best case, the resulting portfolio produces the largest difference between the last values of the high and low portfolios. Howevert, in the worse case, the resulting portfolio produces the least difference between the last values of the high and low portfolios. In short, the performances of the resulting portfolio returns differ greatly in the best and the worst case.

Figure 5: Neural Network Portfolios and Their Training Histories



Note: The top panels show the cumulative portfolio-level returns performance of high and low portfolios of neural network with four layer (NN4) in testing periods, and the bottom panels shows the learning histories in training and validation periods. In training history, we report the loss (MSE) of each epoch for both of training and validation periods. We use the ensemble approach to reduce the prediction variance by averaging the return forecasts from 30 neural network models with the same architecture. The left top panel gives the results of ensemble approach. We define best and worst are the results with largest and smallest difference in last values among 30 records. The middle panels give the portfolios performance of best record and its training history. The right panels give the portfolios performance of worst record and its training history.

Note that the learning history plot provides information to investigate the speed of convergence in loss over epochs and to check whether the training procedure has converged and if the neural network overfits the data. From the learning histories given in the bottom panels, the loss (MSE) of each epoch for both of training and validation periods are about of the same scale. Note that the losses of the two case have converged to zero after 10 epochs in both the best and worse cases. In the best case, the predicted returns allows to predict stocks with higher or lower returns. Unfortunately, this is not the case in the worse case. Indeed, in the worse case, the neural network fails to predict returns well, and this leads to similar last values of the high and low portfolios: the time series plot of high portfolio overlaps with that of the equal-weight portfolio.

In contrast, the ensemble method improves the predictability substantially, in the sense that the resulting high and low portfolio are the highest and lowest among those in the thirty trials. All the above observations indicate that training a single neural network is difficult and the inferred weights could be unstable.

6 Conclusion

Asset pricing models have been a long-standing critical issue both in theory and practice, and there have been numerous traditional methods proposed to explain the stock returns. Machine learning gains a great amount of attention and has been applied to the filed of finance in various aspects recently. One major reason is that machine learning approach is able to capture information using a large set of predictors and avoid the problem of collinearity, which leads to a failure of the traditional methods.

In this thesis, we empirically investigate asset pricing models and compare the performance of traditional linear methods with machine learning techniques. For traditional methods, we consider the OLS, Fama-MacBeth regression, OLS with Huber loss function. Also, for machine learning techniques, we consider the elastic net regression, random forest, gradient boosting regression tree, XGBoost, and neural networks.

To evaluate these methods, we first compare the predictability for stock-level returns of each method by using 102 firm characteristics and 8 macroeconomic factors, which have been shown to be significant in prior studies. Then, we sort the stocks based on their predicted returns from the lowest to the highest, and divide them according to the predicted returns into ten groups. For each group, we form a portfolio with equal weight in each stock. As a result, we are able to construct 10 portfolios, where stocks with the top 10% predicted returns forms the high portfolio and stocks with the bottom 10% stock returns forms the low portfolio.

Our empirical analysis suggests benefits of applying machine learning methods in the filed of empirical asset pricing. First of all, our analysis shows that machine learning methods can improve the predictability in stock returns. Machine learning methods can also estimate risk premia with less errors. Moreover, adding penalty and regularization terms, such as ENet and XGBoost, substantially improves the predicted results of each model. Interestingly, for neural networks, we find that the marginal improvement decreases as the neural network gets ”deeper” since their R^2_{oos} only changes a little.

Second, the success of machine learning methods in returns prediction allows investors to form a portfolio for practical purposes. Indeed, the difference among portfolios constructed based on the predicted stock returns using traditional and machine learning methods becomes even larger. In more details, machine learning methods outperform traditional methods in distinguishing the stocks with higher predicted returns.

Our machine learning approach provides ways to capture the useful information through using a large set of predictors and estimate risk premia with less approximation errors. Our results suggest that machine learning has been an important role in empirical asset pricing fields.

7 Future Work

In the following, we address issues and challenges in implementing machine learning, which deserve careful future research and investigations.

In terms of which explanatory variables to used, we point out three research directions. To start, identifying which predictors provide more useful information in predicting returns is of considerable importance. Green et al. (2017) an Gu et al. (2020) overcome this challenge by using traditional linear models and machine learning methods. Whether the more advanced machine learning methods are able to identify the critical predictors is of considerable importance for empirical asset pricing.

Second, it appears to be useful include interactions between predictors. As shown in Gu et al. (2020), including the interaction effect between 94 firm characteristics and 8 macroeconomic predictors (a total of $94 \times (8+1)$) results in a significant improvement in both the out-of-sample R^2 and economic gains.

Third, the serial correlation effect for the predictors should be considered. It is observed in Figure 3 that serial correlation effect exists in most of predictors, because most ACF at lags 1, 2, and 6, are significantly different from zeros. How to include predictors prior to time t, such as the predictive regressions (Stambaugh, 1999), is worthy of future studies. It is also plausible to include the more recent developed deep neural networks to learn long-term dependencies, such as the recurrent neural networks (RNN), and the Long short-term memory (LSTM).

In terms of numerical challenges, implementing machine learning successfully is usually computationally demanding, because issues such as hyperparameters tuning, optimization, and regularization algorithms, need to be carefully taken into account.

For the tree-based methods, it is shown that they are competitive to neural networks and are able to identify the which stocks will have relatively high or low returns in next month. This results in portfolios based on the tree-based methods yields high Sharp ratios. However, the tree-based methods seem to be influenced by the extreme values and thus fail to predict stock returns directly. It is unclear at this stage how to covercome this issue.

Finally, the estimation for a neural work is computationally demanding. Specifically, even when the loss function appears to be stable while fitting one neural network, parameters based on each estimation result vary drastically among other neural networks of the same architecture and lead to unstable performances of the portfolios. At this stage, we overcome this problem by using the ensemble method, i.e., to estimate many neural networks and take an average of the predicted returns from these neural networks as the final predicted return. However, it would be interesting and challenging to propose a method to more efficiently and accurately estimate a neural network.

Reference

Campbell R. Harvey, Yan Liu, H. Z. (2016). … and the cross-section of expected returns. *The Review of Financial Studies 29*, 5–68.

Chen, T. and C. Guestrin (2016). Xgboost: A scalable tree boosting system. *in Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, New York, USA*.

Cochrane, J. H. (2011). Presidential address: Discount rate. *The Journal of Finance 66*(4), 1047–1108.

Fama, E. F. and K. R. French (1993). Common risk factors in the returns on stocks and bonds. *Journal of Financial Economics 33*(1), 3–56.

Fama, E. F. and K. R. French (1996). Multifactor explanations of asset pricing anomalies. *The Journal of Finance 51*(1), 55–84.

Fama, E. F. and J. D. MacBeth (1973). Risk, return, and equilibrium: Empricial tests. *The Journal of Political Economy 81*(3), 607–636.

Feng, G., N. G. Polson, and J. Xu (2019). Deep learning in asset pricing. *Technical report, University of Chicago*.

Fengy, G., S. Giglio, and D. Xiu (2020). Taming the factor zoo: A test of new factors. *The Journal of Finance 75*(3), 1327–1370.

Goyal, A. and I. Welch (2008). A comprehensive look at the empirical performance of equity premium prediction. *The Review of Economic Studies 21*(4), 1455–1508.

Green, J., J. R. M. Hand, and X. F. Zhang (2017). The characteristics that provide independent information about average U.S. monthly stock returns. *The Review of Finance Studies 30*(12), 4389–4436.

Gu, S., B. Kelly, and D. Xiu (2020). Empirical asset pricing via machine learning. *The Review of Financial Studies 33*(5), 2223–2273.

Gu, S., B. T. Kelly, and D. Xiu (2019). Autoencoder asset pricing models. *Journal of Econometrics, forthcoming*.

Hastie, T., R. Tibshirani, and J. Friedman (2009). *The Elements of Statistical Learning*. New York: Springer.

Heaton, J., N. Polson, and J. Witte (2016). Deep learning for finance: Deep portfolios. *Applied Stochastic Models in Business and Industry 33*(1), 3–12.

Hou, K., C. Xue, and L. Zhang (2015). Digesting anomalies: An investment approach. *The Review of Financial Studies 28*(3), 650–705.

Jeremiah, G., H. J. RM, and Z. X. Frank (2013). The supraview of return predictive signals. *Review of Accounting Studies 18*(3), 692–730.

Kelly, B., S. Pruitt, and Y. Su (2019). Characteristics are covariances: A unified model of risk and return. *Journal of Financial Economics 134*(3), 501–524.

Lewellen, J. (2015). Cross-section of expected stock return. *Critical Finance Review* (4), 1–44.

Mclean, R. D. and J. Pontiff (2016). Does academic research destroy stock return predictability? *The Journal of Finance 71*(1), 5–32.

Roll, R. and S. A. Ross (1980). An empirical investigation of the arbitrage pricing theory. *The Journal of Finance 35*(5), 1073–1103.

Sharpe, W. F. (1964). Capital asset prices: A theory of market equilibrium under conditions of risk. *The Journal of Finance 19*(3), 425–442.

Stambaugh, R. F. (1999). Predictive regressions. *Journal of Financial Economics 54*(3), 375–421.

Tsang, K. H. and H. Y. Wong (2019). Deep-learning solution to portfolio selection with serially-dependent returns. *SIAM Journal on Financial Mathematics 11*(2), 593–619.