Paper Replication: Empirical Asset Pricing via Machine Learning

Haolin HE
Department of ISOM
Hong Kong University of Science and Technology
hheaz@connect.ust.hk

Lingchong LIU
Department of ISOM
Hong Kong University of Science and Technology
lliuba@connect.ust.hk

Ruizhao HUANG
Department of ISOM
Hong Kong University of Science and Technology
rhuangbb@connect.ust.hk[*]

Abstract

The study by Gu, Kelly, and Xiu (1) investigates the use of machine learning (ML) techniques for predicting asset risk premiums within the framework of empirical asset pricing. The research demonstrates that ML approaches can substantially improve the accuracy of asset return predictions over traditional regression-based methods. In this project, we focus primarily on replicating the six ML methods employed by (1). Our empirical findings reveal that tree-based models and dimension reduction techniques are the most effective in performance.

1 Introduction

Gu, Kelly, and Xiu (1) conduct a comparative analysis of machine learning methods for measuring asset risk premium. In this report, we replicate the empirical results demonstrated in (1). In particular, we conduct model training, validating, testing using ordinary least squares (OLS), partial least squares (PLS), principal component regression (PCR), elastic net (ENet), generalized linear model (GLM), gradient boosted regression tree (GBRT), and random forest (RF). We further evaluate the performance of each method and trace the most prominent predictors.

[*]The authors contributed equally to the replication task. Specifically, Haolin HE, Lingchong LIU and Ruizhao Huang were responsible for the OLS and RF components, the PCR and PLS components, and the ENET and GBRT components respectively.

2 Methodology

2.1 Simple Linear

We start by assuming a simple linear regression model. Instead of estimating the model using standard least squares objective function, (1) uses the Huber robust objective function:

$$\mathcal{L}(\theta) = \frac{1}{NT} \sum_{i=1}^{N} \sum_{t=1}^{T} H\left(r_{i,t+1} - g\left(z_{i,t}; \theta\right); \xi\right)$$

where the model function $g\left(z_{i,t}; \theta\right)$ is defined as a linear combination of the predictors.

$$g\left(z_{i,t}; \theta\right) = \theta^{\mathsf{T}} z_{i,t}$$

the Huber function is defined as

$$H(x; \xi) = \begin{cases} x^2, & \text{if } |x| \leq \xi \\ 2\xi\,|x| - \xi^2, & \text{otherwise} \end{cases}$$

The Huber loss function is a combination of the squared loss for relatively small errors and the absolute loss for relatively large errors. A parameter ξ is used for determining between small and large errors.

2.2 Penalized Linear: Elastic Net (ENet)

In addition to the simple linear model, (1) also employs the ENet approach, which integrates the regularization techniques of both lasso and ridge penalties within the linear regression framework. This method is designed to combat overfitting by incorporating these penalty terms into the model's loss function, thereby managing the model's complexity and promoting smaller or zero coefficients. Specifically, the ENet loss function is defined as:

$$\mathcal{L}(\theta) = \frac{1}{NT} \sum_{i=1}^{N} \sum_{t=1}^{T} \left(r_{i,t+1} - g\left(z_{i,t}; \theta\right)\right)^2 + \lambda\left((1-\rho) \sum_{j=1}^{P} |\theta_j| + \frac{1}{2}\rho \sum_{j=1}^{P} \theta_j^2\right)$$

Here, λ represents the regularization parameter that dictates the strength of the penalty. A higher value of λ enhances the regularization effect, which can lead to a reduction in the effective degrees of freedom of the model. The parameter ρ adjusts the balance between lasso and ridge penalties, facilitating a flexible adaptation to various data characteristics.

2.3 Dimension Reduction: Principal Components Regression (PCR) and Partial Least Squares (PLS)

In Section 3, it can be observed that while there are numerous predictors available, not all of them contribute significantly to the prediction or explanation of the outcome. Simply including all predictors without discernment may result in unsatisfactory model performance. Therefore, dimension reduction techniques have been proposed to address this issue. According to (1), dimension reduction involves forming linear combinations of predictors to reduce noise, isolate the signal in predictors, and decorrelate highly dependent predictors. Principal components regression (PCR) and partial least squares (PLS) are two well-known techniques.
Our implementations of PCR and PLS are in the linear regression framework. Let N be the number of stocks, T be the number of months in the time period and P be the number of the predictors. The linear regression framework would be

$$R = Z\theta + E$$

where R is a $NT \times 1$ vector denoting the return, Z is a $NT \times P$ matrix denoting the predictors, θ is a $P \times 1$ vector denoting the coefficients to be fitted and E is a $NT \times 1$ vector denoting the residuals. By performing dimension reduction, one could rewrite the frame work to

$$R = (Z\Omega_K)\theta_K + E$$

where $(Z\Omega_K)$ is a $NT \times K$ vector, denoting the transformed regressor after dimension reduction, and θ_K denotes the reduced coefficients.

2.3.1 PCR

PCR comprises of two modules: Principal components Analysis (PCA) and linear regression. PCA searches for linear combinations of normalized original predictors and passes them to the linear regression module as regressors. The regressors provided by PCA maximize variance and are orthogonal to each other. The technique could be described the optimization problem proposed in (1):

$$w_j = \arg\max_w \operatorname{Var}(Zw), \quad \text{s.t. } w'w = 1, \quad \operatorname{Cov}(Zw, Zw_l) = 0, \quad l = 1, 2, \ldots, j-1.$$

2.3.2 PLS

PLS employ a similar structure as PCR. Instead of maximizing the variance of the linear combination, it maximizes the covariance between the response variable and the linear combination. The technique could be described the optimization problem proposed in (1):

$$w_j = \arg\max_w \operatorname{Cov}^2(R, Zw), \quad \text{s.t.} \quad w'w = 1, \quad \operatorname{Cov}(Zw, Zw_l) = 0, \quad l = 1, 2, \ldots, j-1.$$

2.4 Tree-Based Methods: Gradient Boosted Regression Trees (GBRT) and Random Forests (RF)

2.4.1 GBRT

The first tree-based method (1) considered is GBRT, which is a powerful and flexible machine learning technique that builds on the principles of boosting to optimize predictive accuracy. GBRT combines multiple weak predictive models into a stronger ensemble predictor through a stage-wise fashion where subsequent trees are trained to correct the residuals left by earlier trees. The contribution of each tree $\mathcal{T}$, having K leaves or terminal nodes and a depth of L, can be formally described by the function:

$$g(z_{i,t}; \theta, K, L) = \sum_{k=1}^{K} \theta_k \mathbf{1}_{\{z_{i,t} \in C_k(L)\}}$$

where θ_k represents the constant value assigned to the k^{th} partition. This value is typically the mean outcome within that partition. $\mathbf{1}_{\{z_{i,t} \in C_k(L)\}}$ is an indicator function that equals 1 if the feature vector $z_{i,t}$ falls within the k^{th} partition, $C_k(L)$, and 0 otherwise.

2.4.2 RF

A RF combines forecasts from various trees. The major difference of RF from GBRT is that when reducing the correlation among trees, RF adopts a "dropout" method, which considers only a randomly selected subset of predictors when splitting. The method lowers the average correlation among predictions compared to GBRT. The details of the RF procedure, as stated in (1), could be found in Algorithm 1.

Algorithm 1 Random Forest

```
for $b$ from 1 to $B$ do
    Generate Bootstrap samples $\{z_{i_t}, r_{i,t+1}, (i,t) \in Bootstrap(b)\}$ from the original dataset and
    grow regression trees for each sample. At each step of splitting, sue only a random subset of
    features. The resulting tree grown from the bth sample can be written as:
        $\hat{g}_b\left(z_{i,t}; \hat{\theta}_b, L\right) = \sum_{k=1}^{2^L} \theta_b^{(k)} \mathbf{1}\{z_{i,t} \in C_k(L)\}$
end for
Result: The random forest is given by the average of the outputs of all the B trees.
    $\hat{g}(z_{i,t}; L, B) = \frac{1}{B} \sum_{b=1}^{B} \hat{g}_b\left(z_{i,t}; \hat{\theta}_b, L\right)$
```

2.5 Performance Evaluation and Variable Importance

2.5.1 Out-of-Sample R^2_{oos}

To assess the predictive performance of each model, we adopt the evaluation metric used in (1). Specifically, Specifically, after refitting each model 30 times, we aggregate all the test samples and compute the R^2_{oos} as follows:

$$R^2_{\text{oos}} = 1 - \frac{\sum_{(i,t)\in\mathcal{T}_3} \left(r_{i,t+1} - \widehat{r}_{i,t+1}\right)^2}{\sum_{(i,t)\in\mathcal{T}_3} r^2_{i,t+1}}$$

where $\mathcal{T}_3$ indicates that fits are only assessed on the testing subsample, whose data never enter into model estimation or tuning. This metric quantifies the proportion of "variation" in the observed outcomes that is predictable from the model inputs, offering a measure of model effectiveness in capturing the underlying data patterns.

2.5.2 Variable Importance

In addition to evaluating the predictive power of our models, we also assess their interpretability through Variable Importance (VI) analysis. This approach helps us understand the contribution of individual predictors to the model's performance.
For this project, we do not employ Neural Networks (NN), and therefore, we adopt the first notion of VI in (1). Specifically, this method involves estimating the reduction in the panel predictive R^2 that occurs when all values of a given predictor j are set to zero, while the estimates for all other model components remain fixed within the training sample. This procedure highlights the significance of each predictor by measuring how much the predictive accuracy declines when the predictor is effectively "removed" from the model. Such insights are crucial for identifying key drivers of predictive power and for making informed decisions about model simplification or feature engineering.

3 Experiments

3.1 Experiment Setting

The data retrieved from (1) and (2) were used in this study. Specifically, (1) provided the 94 stock-level predictors and the industry category variable, while the 8 macroeconomic predictors were computed from the raw data provided by (2). In (1), the experiments included interaction terms between the stock-level predictors and the macroeconomic predictors, resulting in a total of $94\times(8+1)+74 = 920$ variables, where 74 is the number of industry category dummy variables. Due to hardware limitations (specifically, RAM) and computation power, we were unable to replicate the experiments at this scale. Instead, we masked the interaction variables and preserved the remaining $94 + 8 + 74 = 176$ variables.
We follow the same recursive performance evaluation scheme as in (1). The initial training set is the data from 1957 to 1974, initial validation set is the data from 1975 to 1986 and initial test set is the data in 1987. The model will be refitted for every 12 months in the whole test set from 1987 to 2016 and in each iteration, the size of in-loop training set will increased by 12 months, while the validation set and the testing set will roll forward by 12 months, keeping the same size. Details could be found in Algorithm 2.
One possible drawback for such a scheme might be the gap between the training set and the testing set. Data of the 12 years that between the training set and the testing set is used for validation, which means the information of such a period only enter the model via hyper-parameter. However, as the validation set is more closed to the testing set in time dimension, the information among the 12-year validation set might be more valuable than the information among the 18-year training set. We suggest in future study, one could try to only use 1-year data for validation, and 29-year data for training, instead of using 12-year data for validation and 18-year data for training.

Table 1 gives the tuning parameters for each method. Due to limitation of memory storage, some of the hyperparameters are shrunken compared to (1).

Table 1: Tuning Parameters For All Methods

	OLS-3	PLS	PCR	ENet	RF	GBRT
Huber	✓	-	-	✓	-	✓

Others		$K \in \{1, 3, 5\}$	$K \in \{30, 50, 70\}$	$\rho = 0.5$
				$\lambda \in (10^{-4}, 10^{-1})$
					Depth $= 1 \sim 3$
					# Trees $= 50$
					# Features $= 30$
						Depth $= 1$
						# Trees $= 500$
						lr = 0.01

Table 2: Monthly out-of-sample Stock-level Prediction Performance R^2_{oos} (Percentage)

	OLS	OLS-3	PCR	PLS	ENet	RF	GBRT
All	-1.14	0.10	1.26	1.28	0.21	0.14	0.82
Top-1000	-0.98	1.07	3.60	2.83	1.34	1.92	3.62
Bottom-1000	-0.14	0.20	0.06	-0.90	-0.47	-0.56	-0.34

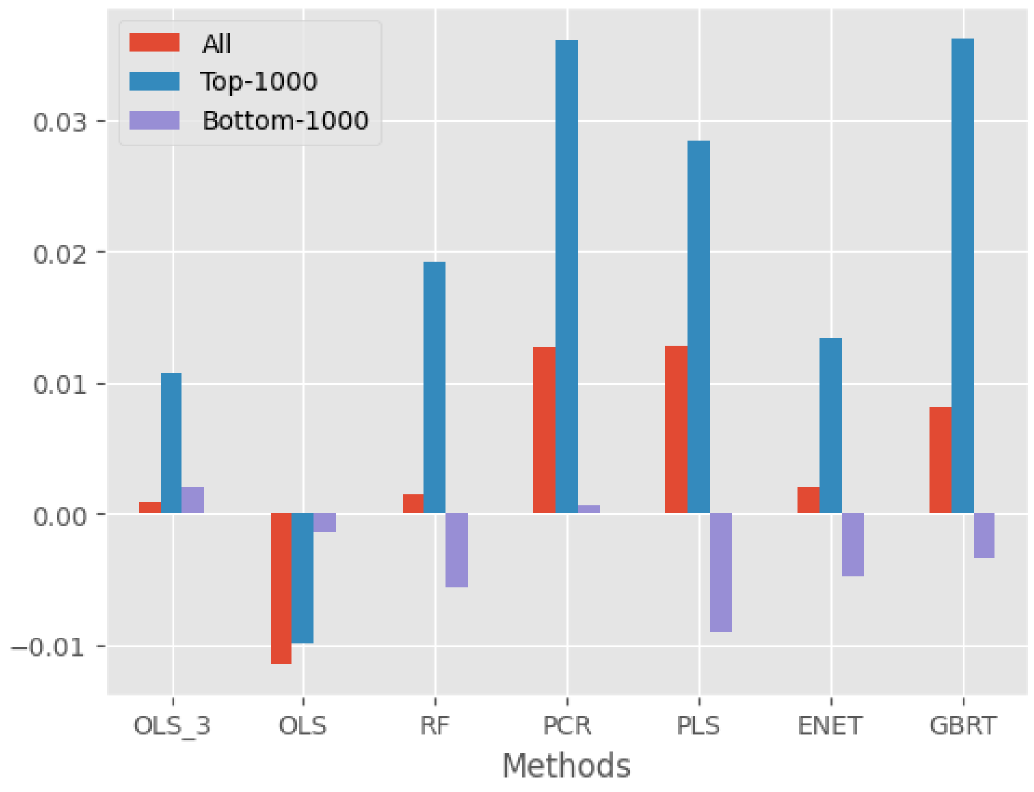

Figure 1: Monthly out-of-sample Stock-level Prediction Performance R^2_{oos}

Algorithm 2 Recursive Performance Evaluation Scheme

```
 1: INPUT:
        - Processed predictors $X$, corresponding response variable $Y$
        - Initial starting and ending dates of training set, validation set and testing set, respectively
        - The search space $\mathcal{Q}$ for tuning hyper-paramter $q$
 2: for $i$ in $\{0, \dots, 29\}$ do
 3:     Get the corresponding training set, validation set and testing set        ▷ Slice the data
 4:     for $q$ in $\mathcal{Q}$ do
 5:         Fit the model with training set and hyper-parameter $q$
 6:         Compute $R^2_{oos}$ on the validation set
 7:     end for
 8:     Pick the model with greatest $R^2_{oos}$
 9:     Compute $R^2_{oos}$ on the testing set
10:     The ending dates of training set move forward by 12 months
11:     The starting and ending dates of training set move forward by 12 months
12: end for
```

3.2 Out-of-Sample R^2_{oos}

By definition, a naive zero prediction would result in $R^2_{oos} = 0$, which could be regarded as a benchmark. From Table 3.2, and Figure 3.2, one could observe that OLS under-performs naive zero prediction, which is expected, as the data is noisy and the number of predictors is large.
We compute R^2_{oos} under 3 setting of the sub-sample: all the stocks, the stocks corresponding to top-1000 market value, the stocks corresponding to bottom-1000 market value. Thanks to dimension reduction, PCR and PLS outperforms the rest of models if we include all the stocks. It is worth noting that all the models (except OLS) performs better if we only include the top-1000 stock, and perform worse if we only include the bottom 1000 stock. One possible explanation is that those companies with higher market value will be more stable in fundamental, hence there will be less noise in data, which will make it easier to capture the signal.

3.3 Variable Importance

We investigate the variable importance in the experiments as described in section 2.5.2 and the results are presented in Figure 3.3. We follow the same scheme as in (1): The variable importance within each model is normalized so that the sum would be 1, then select the variable importance of the top-20 most influential variables. Due to hardware limitations and computation power, we cannot compute the variable importance in each training-testing iteration and use the average, instead, we only compute the variable importance in the last iteration (corresponding to the largest training set).
Figure 3.3 reveal a few worth-noting phenomena. Firstly, when compared to PCR, PLS, and GBRT, ENet and RF exhibit more concentrated variable importance. This suggests that they may be more sensitive to certain variables, which could result in inadequate robustness. This may explain why ENet and RF under-perform when compared to PCR, PLS, and GBRT. Secondly, sparsity can be observed in both ENet and GBRT. In ENet, sparsity may be attributed to the L-1 penalty, while in GBRT, sparsity may be due to the shallow depth of the trees.

4 Conclusion

In conclusion, this report presents a replication of (1), which investigates the application of machine learning methods for predicting asset risk premiums. Our findings align with the original research, demonstrating that tree-based models and dimension reduction techniques, such as PCR, PLS and GBRT, outperform other methods in terms of accuracy. We also identify potential improvements for future studies, including adjusting the training and validation set sizes to better capture information. Our analysis further reveals that models perform better when considering top-1000 market value stocks, as these companies exhibit greater stability and less noise in the data. Additionally, we examine variable importance and observe that ENet and RF models exhibit more concentrated variable importance, possibly leading to inadequate robustness. Sparsity is also noted in both ENet and GBRT, which may be attributed to the L-1 penalty and shallow tree depth, respectively.

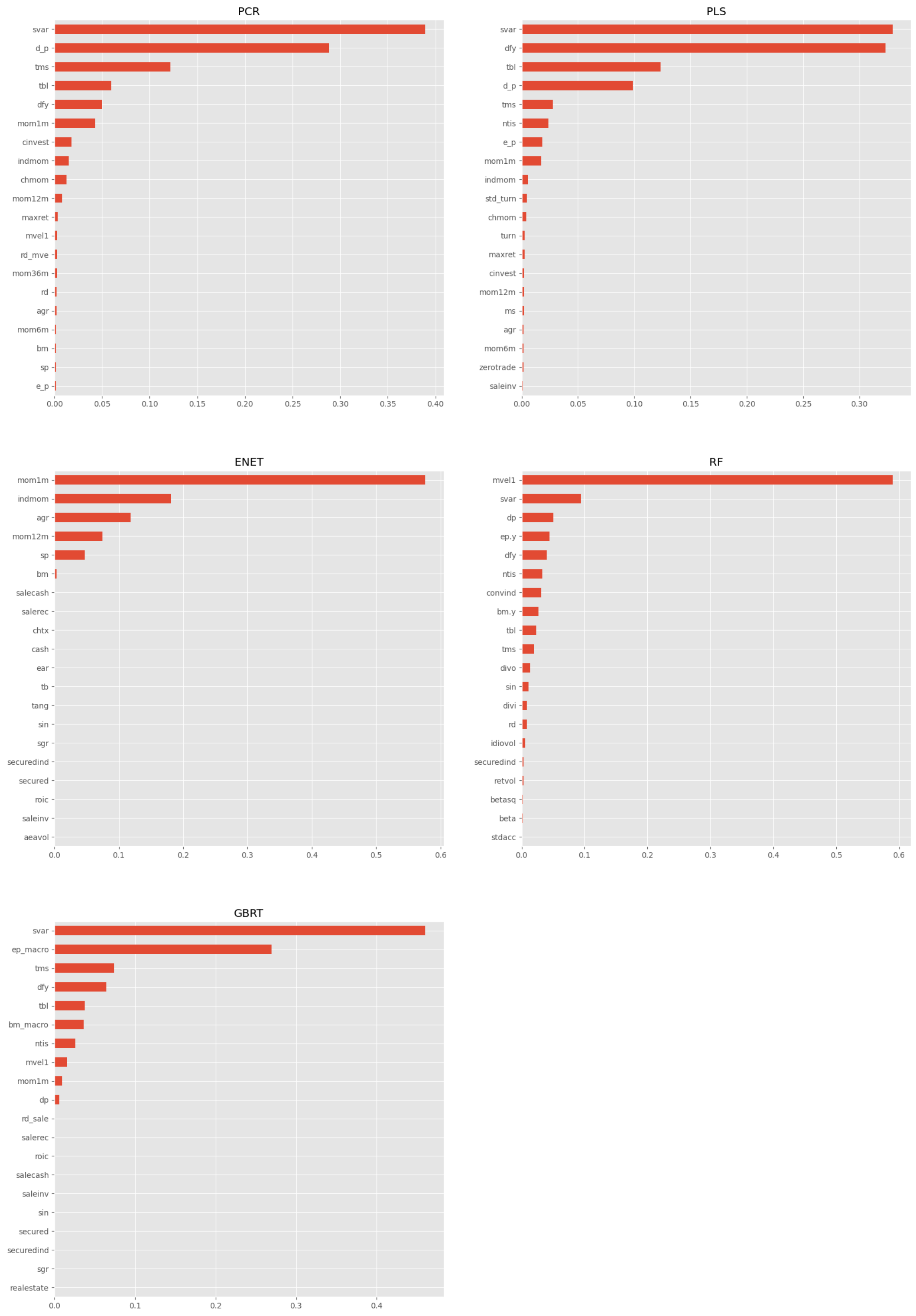

Figure 2: Top 20 Variable Importance

References

[1] S. Gu, B. Kelly, and D. Xiu, “Empirical Asset Pricing via Machine Learning,” *The Review of Financial Studies*, vol. 33, pp. 2223–2273, 02 2020.

[2] I. Welch and A. Goyal, “A Comprehensive Look at The Empirical Performance of Equity Premium Prediction,” *The Review of Financial Studies*, vol. 21, pp. 1455–1508, 03 2007.