# Results Replication:
# Empirical Asset Pricing via Machine Learning

**Zhang Jiaming**

April, 2024

## Abstract

The inherent randomness of the market makes it challenging to isolate and predict the expected return of an asset (risk premium). Gu, Kelly and Xiu (2020) therefore conducted a series of analyses using different machine learning methods to deal with this issue and the results indicate that machine learning has the potential to improve the understanding and highlight the key predictors of empirical asset pricing. Only some selected methods of this paper will be replicated in this report.

## 1 Preliminary and Introduction

This report replicates partially the results from the article Gu, Kelly and Xiu (2020). The original article explores the potential of machine learning methods to predict stock risk premia, at both the level for individual stocks and the aggregate market (by forming portfolios). It compares various machine learning models against traditional statistical methods and highlights the economic benefits of using these advanced techniques by forming portfolios.

Compared to a benchmark OLS model with a few established predictors, machine learning models, particularly trees and neural networks, achieve significantly higher out-of-sample predictive values in the original articles (especially for NN with three hidden layers).

When replicating their results, I only worked on some of that methods with a further truncated time period, due to various constraints like computational power, etc.

## 2 Dataset and Sources

### 2.1 The raw data

For this project, there are two main parts of the data that can be downloaded online. The first part of the data are the monthly level individual equity with nearly 30,000 stock samples (an average of 6,200 stocks per month returns) and 94 stock-level predictive characteristics from CRSP (with stocks listed in the exchanges NYSE, NASDAQ and AMEX). The risk-free interest rate is the Treasury-bill rate. The second part of the data are 8 macroeconomic predictors. This is a panel data structure.

This stocks' panel dataset (named GKX_20201231.csv) downloaded has 101 variables and 4345508 observations and ranging from January 1926 to December 2016 (now expanded to 2020), but the original paper only used the data after 1957-2016.

This truncated dataset (1957-2016) is divided into three periods in the original paper.
1957-1974: training sample set, for training and estimate parameters
1975-1986: validation sample set;
1987-2016: testing sample set, as used to evaluate the predictive results as trained by the previous two sets.

According to their current dataset and also appendix for different algorithms shared online, the similar analysis up to years over 2016 can also be conducted.

The 8 macroeconomic predictors (named PredictorData2023.csv) are from Welch and Goyal (2008), and this dataset is currently up to 2023. The data shape is 1837 rows and 18 columns. As the paper compared up to 2016, only part of the data will be used here in the macro set. Note that we also need to calculate some predictors by ourselves.

In processing these two datasets, one may first merge them as one complete dataset and work on this merged version.

### 2.2 Data merging, splitting and cleaning

Before performing the data analysis, the data cleaning and merging will be necessary. This includes dropping the missing value, creating the industry dummies and understanding the dataset, etc. In the programming session, due to the constraints of the computation power, only a further truncated dataset is worked on. Further, the dataset is splitted into the training and testing set. The testing set shall be naturally later than the training set in time.

## 3 Methodology and the corresponding alorithms

We first recall the general form of the predicted excess return of the asset as

$$r_{i,t+1} = \mathbb{E}_t(r_{i,t+1}) + \epsilon_{i,t+1}$$

with the predicted term by various machine learning or other types of model as

$$\mathbb{E}_t(r_{i,t+1}) = g^*(z_{i,t}).$$

The data possess a "tensor" structure combining of the stock level and macroeconomic level data, that is

$$z_{i,t} = x_t \otimes c_{i,t},$$

and with dimension $P = P_c P_x = 920$, which is the input dimension in the following.

The performance of each model will be assessed by the out of sample $R^2$. The calculation is given by

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

where the predicted values are given by the following different methods.

Following the construction of the paper, the top 1000 and bottom 1000 stocks' $R^2$ in each method will also be presented and form a table at last. This will help us better understand how the models work for good-performing and bad-performing companies in the market, and potentially determine which machine learning methods can fit those companys' risk premia better.

One also needs to compare the performance between different pairs of models, and this requires the Diebold and Mariano (DM) test, basically by comparing which method has average squared larger error.

### 3.1 OLS and Adjusted simple linear with Huber robust objective function

The simple linear regression is a relatively less complicated model. The original OLS model is

$$g(z_{i,t};\theta) = z_{i,t}\theta,$$

with OLS objective function as

$$\mathcal{L}(\theta) := \frac{1}{NT}\sum_{i=1}^{N}\sum_{t=1}^{T}(r_{i,t} - g(z_{i,t};\theta))^2.$$

It is a common choice to introduce the Huber robust objective function for dealing with the heavy-tailed observations. The Huber objective function is defined as

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

with

$$H(x,\xi) := \begin{cases} x^2, & |x| \le \xi; \\ 2\xi|x| - \xi^2, & |x| > \xi. \end{cases}$$

### 3.2 PCR and PLS

Principal components regression (PCR) and Partial Least Squares regression (PLS) are two dimension reduction methods. In the original paper, this was realized simply by two steps, the first one is dimension reduction by making some of the variables into a linear combination form, and the second step is to regress on those linear combinations.

To be more precise, both PCR and PLS start from the vectorized form of linear combinations. That is rewrite $r_{i,t+1} = z'_{i,t}\theta + \epsilon_{i,t+1}$ as:

$$R = Z\theta + E,$$

where $R$ is a vector of $r_{i,t+1}$ with size $NT \times 1$, $Z$ is a matrix of predictors $z_{i,t}$ with size $NT \times P$, and $E$ is a vector of residuals $\epsilon_{i,t+1}$ with size $NT \times 1$.

The regression model is:

$$R = (Z\Omega_K)\theta_K + \tilde{E},$$

where $Z\Omega_K$ is obtained by reducing the dimension. $\Omega_K$ is a matrix with columns $w_1, w_2, \cdots, w_K$.

For PCR, one needs to solve for each $j$:

$$w_j = \arg\max_w Var(Zw),$$

s.t. $w'w = 1, Cov(Zw, Zw_l) = 0, l = 1, 2, ..., j-1.$

For PLS, one solves:

$$w_j = \arg\max_w Cov^2(R, Zw),$$

s.t. $w'w = 1$, Cov $(Zw, Zw_l) = 0, l = 1, 2, ..., j-1.$

### 3.3 ENet

Due to the potential overfitting of the linear model when the noise is large, a common trick is to introduce some proper penalty functions on the objective functions. In the paper uses elastic net, that is a combination of LASSO and Ridge. The penalized objective function is defined as

$$\mathcal{L}(\theta;\cdot) = \mathcal{L}(\theta) + \phi(\theta;\cdot),$$

with the ENet case

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

### 3.4 LASSO

This is a special case when the $\rho = 0$ in the previous penalty function. We also include LASSO and compare it with ENet and other methods as well.

### 3.5 NN3

In the original paper, NN1 to NN5 are considered, but the best-performing one is NN3. Therefore it implies that a rather shallow or not too deep layers will already be useful and there is a limit of the model performance in increasing the layers. When replicating their results, the choice will be NN3 (with 3 hidden layers).

The input layers will have 920 neurons, same as the dimension of $P$ as explained in Section 2. Three hidden layers with 32, 16, and 8 neurons are used following the paper, respectively. ReLU function is chosed as the activation function to make the derivative calculation faster in each and all layers, as defined by

$$ReLU(x) = \begin{cases} 0, & x < 0; \\ x, & x \ge 0. \end{cases}$$

In training the NN3, the MSE ($l_2$) and $l_1$ loss function are attempted in pytorch. I also apply Batch normalization to faster train the NN3, with batch size 5000, 10000 and 20000, respectively. It turns out that using $l_1$ loss will make the convergence much faster comparing to $l_2$. However, when changing the batch size (increase or decrease), then this issue seems to be vanishing. For SGD part, Adam SGD method is used according to the online appendix of the original paper and directly by the torch code "torch.optim.Adam()".

To be more precise, the mathematical model of the NN with in total $L$ layers can be described by the following iterative definition:

$$x_k^{(l)} := ReLU(x^{(l-1)'}\theta_k^{(l-1)}),$$

and the output layer is

$$g(z, \theta) = x^{(L-1)'}\theta_k^{(L-1)}.$$

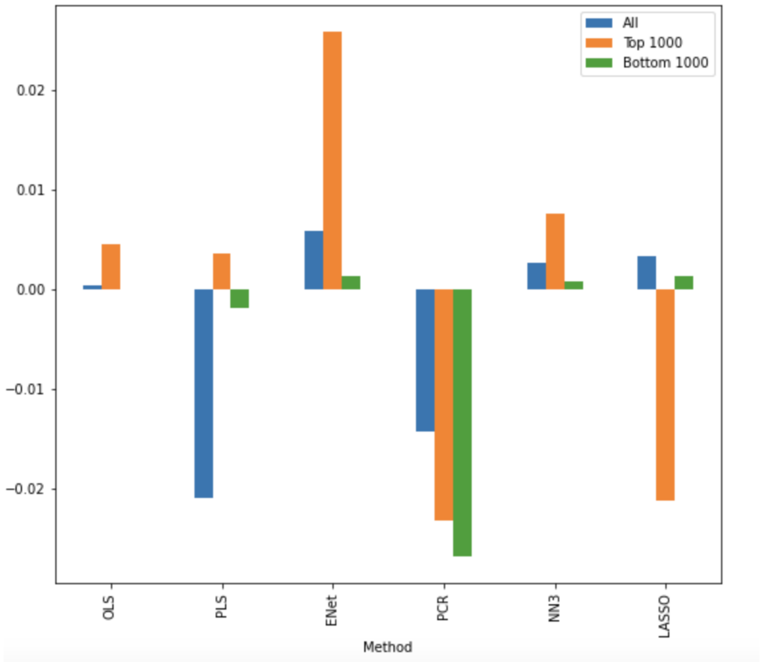
In our case, L=5 in total, with three hidden layers plus the input and output layers.

When coding, NN1-NN5 are also attempted, but they will not be included in the performance analysis.

## 4 Results and Performance

Due to limited computation power, I only try to use the data as training set and testing set. The data from 2008-2015 is used as the training set, and the year 2016 data is used as the testing set.

### 4.1 $R^2_{oos}$ comparison



In this figure, I include the overall $R^2_{oos}$ as well as $R^2_{oos}$ for top and bottom 1000 stocks using different models. ENet and NN3 seem to have the best performance, but the explanatory power is not strong comparing to the original paper. PCR and PLS are very effective. Some models even work differently for top and bottom stocks. The reason is only 8 years data are used as training, which is clearly not enough. The results are presented in the following table. In general, the explanatory power of all the included models have a large room to increase. It may also indicate that when the training set is not big enough, some simple method like ENet may perform better than the complicated nonlinear method.

| $R^2_{oos}$ | | | |
|---|---|---|---|
| Models | overall $R^2_{oos}$ | top 1000 $R^2_{oos}$ | bottom 1000 $R^2_{oos}$ |
| OLS | 0.0004313 | 0.0045692 | 4.89822962e-054 |
| PLS | -0.0208765 | 0.0036662 | -0.0018202 |
| ENet | 0.0058293 | 0.0259246 | 0.0013079 |
| PCR | -0.0142888 | -0.0231877 | -0.0268121 |
| NN3 | 0.0027359 | 0.0076275 | 0.0007627 |
| LASSO | 0.0033831 | -0.0211763 | 0.0013768 |

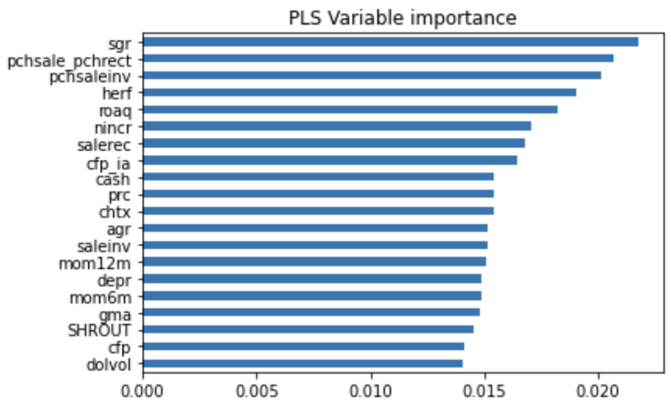
The DM test table is also included as the following:

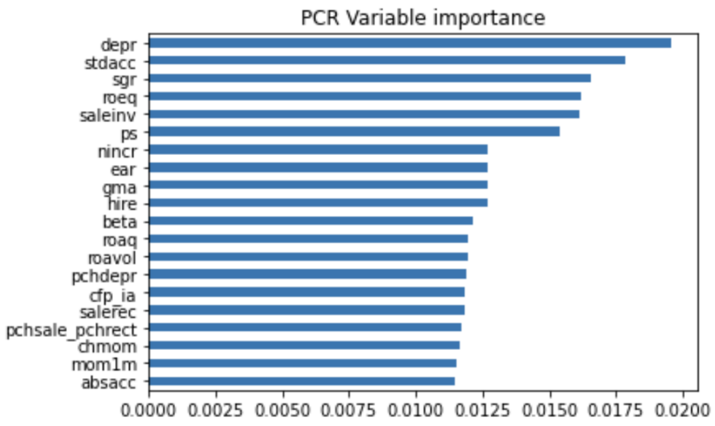
| Models | OLS | PLS | ENet | PCR | NN3 | LASSO |
|---|---|---|---|---|---|---|
| OLS | 0 | 0.30533266 | 0.20043865 | -0.98444982 | -0.53356513 | -0.12866752 |
| PLS | 0 | 0 | -0.54413629 | -0.99140457 | -0.64758319 | -0.37754218 |
| ENet | 0 | 0 | 0 | -1.01735616 | -0.60795009 | -0.26199831 |
| PCR | 0 | 0 | 0 | 0 | -0.14160073 | 0.61347735 |
| NN3 | 0 | 0 | 0 | 0 | 0 | 0.42214179 |
| LASSO | 0 | 0 | 0 | 0 | 0 | 0 |

From this pairwise comparing table, there is no clear advantage of NN3 and ENet observed comparing to other methods.

### 4.2 Variable importance

When conducting variable importance analysis, one has to make some of the variables zero and reconduct the machine learning procedure. This makes the whole computation workload large, and therefore I further reduce the year number of the training data due to time constraint.





The two Variable importance plots of PCR and PLS models are included. It seems that for those two models the important variables are quite different.

## 5 Conclusion

In section 4, we observe that all the methods in the replication do not perform that well comparing to the original paper. This is due to the limited training set to tune the parameters, as well as data cleaning skills as well. I find it quite difficult to understand and organize the data. Also, if the coding is more effective, then perhaps we can make the whole process more effective. Since I am a beginner in coding, during this process I learn a lot about building up the code using python.

Larger training data, better parameter adjusting and programming skills are needed to improve the effectiveness of those machine learning models.

## References

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

[2] Welch, I., Goyal, A. (2008). A comprehensive look at the empirical performance of equity premium prediction. *The Review of Financial Studies*, 21(4), 1455-1508.

[3] Diebold, F. X. (2015). Comparing predictive accuracy, twenty years later: A personal perspective on the use and abuse of Diebold–Mariano tests. *Journal of Business & Economic Statistics*, 33(1), 1-1.