Replication Work on

Empirical Asset Pricing via Machine Learning

Huang Zhenyu

zhuangcr@connect.ust.hk

Department of Mathematics, School of Science

The Hong Kong University of Science and Technology

Contents

1 Introduction

Machine learning and deep learning algorithms have been utilized in financial fields like banks, hedge funds and other investment corporations for a long time. In this analysis, we are going to replicate what Gu et al. have done in their paper and hoping to see if applying the modern techniques in computer science to a financial context will help us analyze the predictive power of different variables.

2 Data description

The dataset we use comes from Xiu et al., which is available on their homepage. The data is mainly combined with various financial indicators collected by COMPUSTAT and CRSP. The dataset starts from 1926 and ends in 2020, so there are over 4,000,000 observations in it. The number of variables is 101, including 95 predictive characteristics related to stocks and 2 characteristics for calculating the market value. The 8 macroeconomic variables in Goyal and Welch are also included in our work, so in total there are 102 descriptive variables in the dataset.

No.	Acronym	Firm characteristic	Paper's author(s)	Year, Journal	Data Source	Frequency
1	absacc	Absolute accruals	Bandyopadhyay, Huang & Wirjanto	2010, WP	Compustat	Annual
2	acc	Working capital accruals	Sloan	1996, TAR	Compustat	Annual
3	aeavol	Abnormal earnings announcement volume	Lerman, Livnat & Mendenhall	2007, WP	Compustat+CRSP	Quarterly
4	age	# years since first Compustat coverage	Jiang, Lee & Zhang	2005, RAS	Compustat	Annual
5	agr	Asset growth	Cooper, Gulen & Schill	2008, JF	Compustat	Annual
6	baspread	Bid-ask spread	Amihud & Mendelson	1989, JF	CRSP	Monthly
7	beta	Beta	Fama & MacBeth	1973, JPE	CRSP	Monthly
8	betasq	Beta squared	Fama & MacBeth	1973, JPE	CRSP	Monthly
9	bm	Book-to-market	Rosenberg, Reid & Lanstein	1985, JPM	Compustat+CRSP	Annual
10	bm_ia	Industry-adjusted book to market	Asness, Porter & Stevens	2000, WP	Compustat+CRSP	Annual
11	cash	Cash holdings	Palazzo	2012, JFE	Compustat	Quarterly
12	cashdebt	Cash flow to debt	Ou & Penman	1989, JAE	Compustat	Annual
13	cashpr	Cash productivity	Chandrashekar & Rao	2009, WP	Compustat	Annual
14	cfp	Cash flow to price ratio	Desai, Rajgopal & Venkatachalam	2004, TAR	Compustat	Annual
15	cfp_ia	Industry-adjusted cash flow to price ratio	Asness, Porter & Stevens	2000, WP	Compustat	Annual
16	chatoia	Industry-adjusted change in asset turnover	Soliman	2008, TAR	Compustat	Annual
17	chcsho	Change in shares outstanding	Pontiff & Woodgate	2008, JF	Compustat	Annual
18	chempia	Industry-adjusted change in employees	Asness, Porter & Stevens	1994, WP	Compustat	Annual

Figure 1: Description of some variables in dataset given by Gu et al.

2.1 Data Visualization

Figure 2 displays the distribution of data for each variable. The distribution of most variables is symmetric, and the distribution of a small number of variables is skewed, indicating that there might be some outliers in the data. We will discuss it in later section.

One effective way to understand the data is to calculate correlations among the features. Then, we can observe the correlation between the data of various variables according to the heat map diagram and counted the 10 pairs of variables with the highest absolute value of correlation to help our subsequent research. Among them, quick and 'currat', 'pchquick' and 'pchcurrat', 'betasq' and 'beta', 'retvol' and 'maxret' are highly correlated. Figure 3 shows the overall correlation among variables.

2.2 Data preprocessing

Since there are nearly 100 years' data, we believe there are large amount of missing data existing. We calculate the percentage of missing values of each variable every year in the original dataset. It is not surprising that the percentages of blank values are close to 1 for some variables before the 1970s since many descriptive characteristics are not introduced yet in that period. So, we decide not to use the data before 1977 as we don't have much information about the stock characteristics in those years and reasonable imputation would be hard. Detailed view of the missing data can be found in figure 4.

Although we select the years that contain fewer missing values in each variable, we must handle the

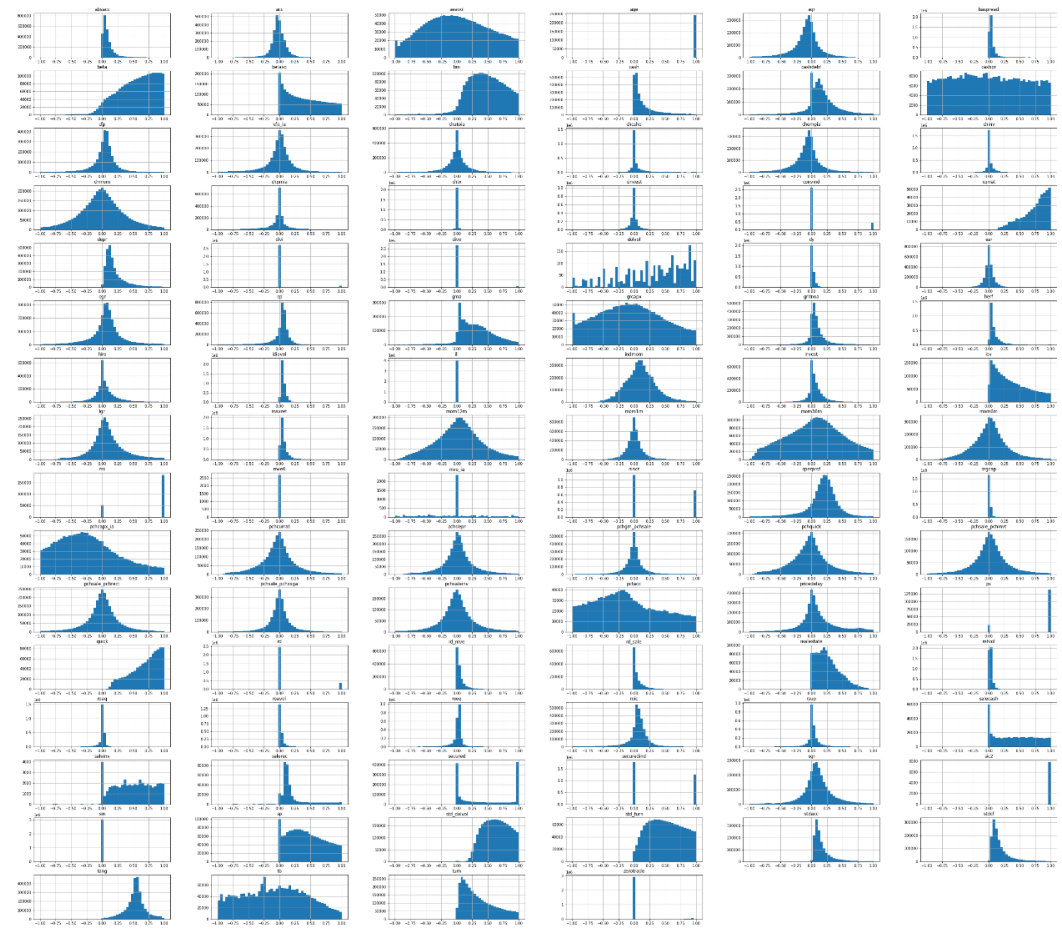


Figure 2: Distribution of all variables

missing values first before modeling. Xiu et al. suggests we should use the cross-sectional median in each month for each stock in imputation. However, there are still some blanks that cannot be filled in as we fail to find any existing value in one month for certain stocks. As a result, we decide to use 0 to represent all these variables. Now we use the data only from 1977 to mid-2019, the length of which is exactly 70% of the data period used in the paper to be replicated.

It is recommended that we need to standardize the variables before model training, especially when we use linear models. Distribution of the variables is shown in figure 2 and we mainly use the Z-score methods to transform all these numerical characteristics. Standard scaler, which transforms the original variables to a set of numbers with 0 mean and unit standard deviation help us in the transformation.

Besides, when we look further into the data, we find some inner connection between variables ‘sic2’, the categorial code and ‘permno’, representing the stocks. We believe the category of each stock remain stable in our dataset, so we just impute the missing sector by each stock and throw away stocks with unknown sector instead of regarding it as 0 or other number, since it may introduce new information that will influence the model and result. We also think that stocks in the same category may have something similar in the stock-specific level of variables, so we finally fill in the blanks by each category in each month. The dataset now contains roughly 3.2 million observations and 103 features.

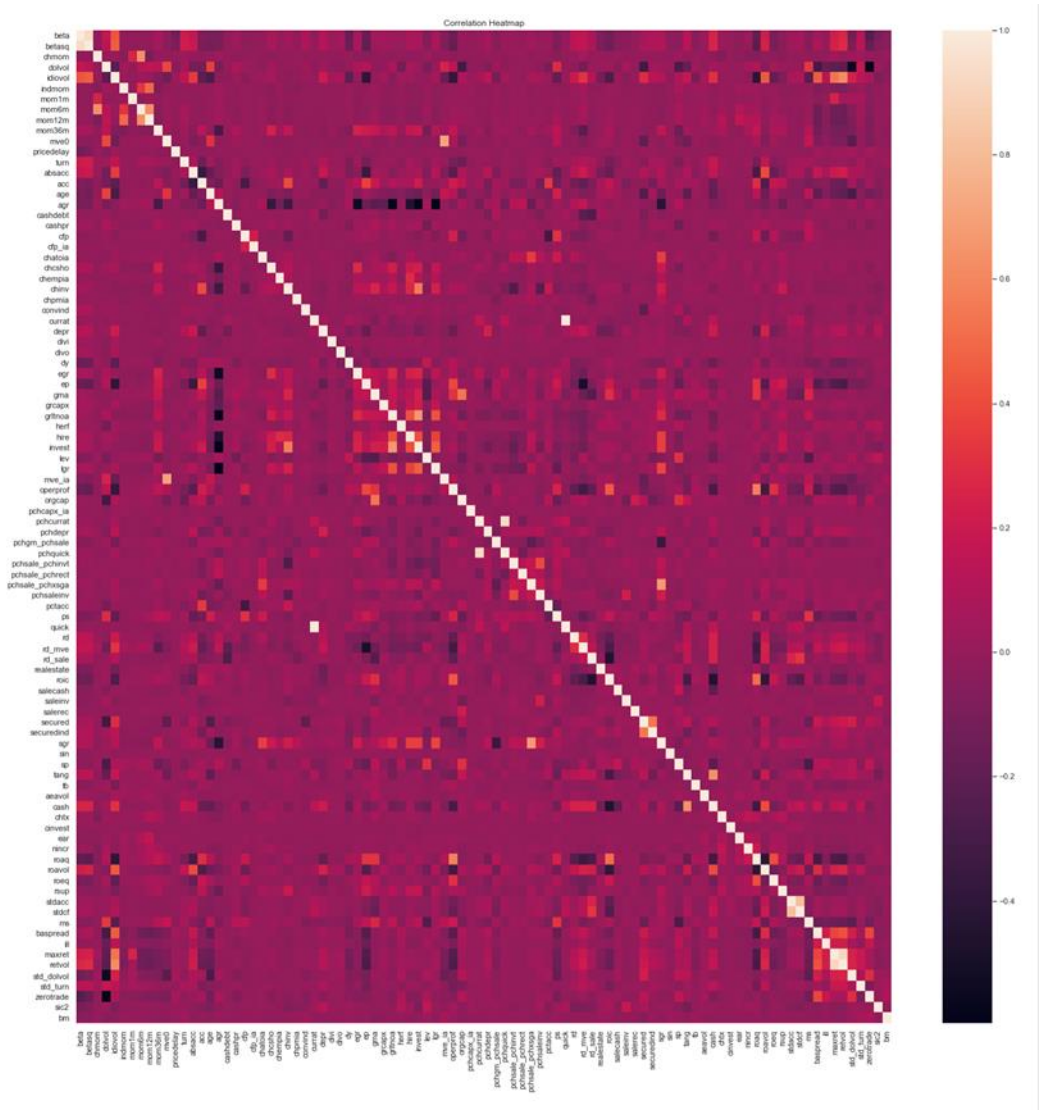


Figure 3: correlation heatmap of all variables

2.3 Feature engineering

In Shihao et al.'s work, the interaction between the 8 macromedia factors and stock-specific factors has been taken into consideration in model training. However, due to the memory and computational limit of our local devices, we fail to generate all these interaction characteristics. Instead, we calculate the interaction between the top one-third variables and all macromedia ones based on the discussion in section 2.2. Here we exclude those indicators whose values are only 0 and 1. Therefore, we reduce the number of stock-specific variables to 30 and finally generate a total of 240 new features.

Moreover, calculating lags and rolling values are effective ways to predict future results in time series data analysis. Lag is expressed in a time unit and corresponds to the amount of data history we allow the model to use when making the prediction. Rolling features creates a rolling window with a specified size and perform calculations on data in this window which of course rolls through data, here we use the rolling mean and median for our further work. We now calculate the lags of the stock return variable for 1 month, 3 months, half year and 1 year, and the rolling mean and median of variable maximum daily return, return volatility, and the stock return with a window of 3 months, 6 months, and

12 months. We now have 58 new features for our prediction.

	name	1957	1958	1959	1960	1961	1962	1963	1964	1965
19	absacc	1.0	1.0	1.000000	1.000000	1.000000	1.000000	0.999756	0.995512	0.982916
20	acc	1.0	1.0	1.000000	1.000000	1.000000	1.000000	0.999756	0.995512	0.982916
21	age	1.0	1.0	0.999456	0.992821	0.983337	0.973756	0.840172	0.729645	0.695645
22	agr	1.0	1.0	1.000000	1.000000	0.997926	0.994510	0.974106	0.833466	0.730241
23	cashdebt	1.0	1.0	0.999456	0.993652	0.986225	0.977270	0.852145	0.749305	0.717127
24	cashpr	1.0	1.0	0.999456	0.992821	0.983337	0.973756	0.858192	0.765788	0.734560
25	cfp	1.0	1.0	1.000000	1.000000	1.000000	0.999671	0.997565	0.992970	0.976067
26	cfp_ia	1.0	1.0	1.000000	1.000000	1.000000	0.999671	0.997565	0.992970	0.976067
27	chatoia	1.0	1.0	1.000000	1.000000	1.000000	0.998627	0.996347	0.975812	0.836712
28	chcsho	1.0	1.0	1.000000	1.000000	0.997926	0.994510	0.974187	0.833942	0.730864
29	chempia	1.0	1.0	1.000000	1.000000	0.997926	0.994510	0.974106	0.833466	0.730241

Figure 4: missing value percentage of some variables in some years

3 Methodology

3.1 Dataset splitting, model training and performance evaluation

We first divide our full dataset into training, validating, and testing samples by general machine learning literature. Following the construction in the paper, we first divide 30% of the data into training sample, 20% into validation and the remaining part into testing set. That is, 13 years for training, 8 years for validation and 21 years for testing at the beginning. We adopt the recursive training by Gu et al. and increase the training data by one year while keeping the period of validation years unchanged. In total we train 21 times for each model. Cross validation is not suitable in our analysis since it may take future information into consideration, but it is unknown until we reach the latest time stamp.

In section 2.3, new features about interaction, lags and rolling window statistics are generated in our work. However, due to our knowledge of machine learning methods, we tend not to use these new features in simple linear methods as we believe linear models cannot handle a larger dimension very well compared to the tree-based methods and neural networks. Linear models may focus more on the relationship between the features and target variable, and the inner effect of the variables might be ignored. Therefore, we just use the original 103 features in all linear models and further add some features in more complex models. Besides, the observation of each stock is also counted in the whole dataset, and we decide to drop the stocks with comparatively low observations to reach a higher training speed. We also believe by keeping stocks that exist in the dataset for a longer time, we can reach a more satisfactory result since those “outliers” may influence the final prediction. We eventually drop around 10% of the data.

According to Shihao et al.’s discussion, the evaluation of our model is judged by out-of-sample R^2 score defined as

$$R^2_{oos} = 1 - \frac{\sum_{(i,t\in\tau_3)}\left(r_{i,t+1} - \hat{r}_{i,t+1}\right)^2}{\sum_{(i,t\in\tau_3)} {r_{i,t+1}}^2}$$

They further consider that using past average return and information to predict future return will surely underestimate the prediction value since noise play a part in the past data. So, they compute the percentage R^2 by comparing the one given by model prediction and the one obtained via treating the prediction as 0. Moreover, since the training sample includes new information in one year, we decide to do prediction in the testing set by year to keep pace with the training method.

The Diebold-Mariano statistic is defined as

$$DM = \frac{\bar{d}}{\sqrt{\gamma_0 + 2\sum_{k=1}^{h-1}\gamma_k}}, for\ h \geq k \geq 1$$

where $\bar{d}$ is the mean of the difference of 2 time-series values and γ_k is the autocorrelation function, or the autocovariance at lag k defined as

$$\gamma_k = \frac{1}{n}\sum_{i=k+1}^{n}(d_i - \bar{d})(d_{i-k} - \bar{d})$$

and it is generally sufficient to set $h = n^{\frac{1}{3}} + 1$. Under the null hypothesis that $\mu = \bar{d} = 0$, DM follows a standard normal distribution

$$DM \sim N(0,1)$$

In practice, Diebold-Mariano test tends to reject the null hypothesis too often for small samples. A better test is the Harvey, Leybourne and Newbold (HLN) test, which is based on the following:

$$HLN = DM\sqrt{\frac{[n + 1 - 2h + h(h-1)]}{n}} \sim T(n-1)$$

In the paper, the annual cross-sectional average of prediction errors from each model instead of comparing errors among individual returns. Following this idea, root mean squared error (RMSE) is adopted to obtain the error series between the true value and prediction of each stock in each year in our approach. Since all the errors are squared in calculating the Diebold-Mariano statistic in each year, the error metric is simply the mean square error.

3.2 Linear models

3.2.1 Simple linear

Linear regression is a simple method that can use a 'straight line' or hyperplane to describe the relationship between response variable and characteristics. It is still popular in real world mainly because it can give clear explanation to the relationship between the response and feature. The loss function, which is generally least squares, controls the estimation error between the ground true value and prediction. But as we mentioned in section 2.2, there seems to be outliers in our dataset, so we adopt Huber loss function to give better prediction. Huber loss function is a kind of robust method that is less sensitive to outliers in data than the squared error loss. It is defined as

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

In our model, we choose the parameter epsilon = 1.1 to train a more robust model, where the default epsilon = 1.35 reach a quantile of 95% of the parameter ξ

Elastic Net is another regularization method that uses the combination of $l1$ and $l2$ penalties to cope with overfitting problems. The elastic net method overcomes the limitations of the LASSO (least absolute shrinkage and selection operator) method. In the case that we have high-dimensional data with few examples, the LASSO selects at most n variables before it saturates. Also, if there is a group of highly correlated variables, then the LASSO tends to select one variable from a group and ignore the others. To overcome these limitations, the elastic net adds a quadratic part to the penalty, which when used alone is ridge regression (known also as Tikhonov regularization). Thus, it adopts the advantages in both Ridge and Lasso regression.

3.2.2 Principle component methods

Despite tuning loss function, we also try principal component regression (PCR) and partial least squares (PLS) which are useful in dimension reduction. PCR combines principal component analysis (PCA) that can best describes the dataset with a smaller number of features and one linear regression. It helps to avoid overfitting, but the dimension reduction and regression are not working simultaneously. Different from PCR, PLS is a supervised learning method whose purpose is to find directions that can explain the independent variables as well as the dependent variables. It is a regression modeling method of transforming multiple dependent variables to multiple independent variables so that it is easy to explain the regression coefficients of independent variables as well as identifying the noise.

3.3 Tree based methods

3.3.1 Bagging and boosting

Random Forest Regression is a supervised learning algorithm that uses ensemble learning method for regression. Ensemble learning method is a technique that combines predictions from multiple machine learning algorithms to make a more accurate prediction than a single model. In the training part, bootstrap sampling is utilized to collect multiple different sub-training datasets from the original one to train multiple different decision trees, the purpose of which is to increase the robustness and stability of the final model prediction results (i.e., reduce the variance) with multiple different sub-models. In a decision tree, each internal node represents a judgment on an attribute, each branch represents an output of a judgment result, and finally the mean of each leaf node represents a regression result. Random Forest regression contains multiple decision trees, and the output is determined by the mean of each the individual tree. It is capable of handling high-dimensional data and does not need much parameter tuning and feature selection. It also performs well in dealing with outliers and overfitting issues. In regression task, it is suggested that the number of features in each splitting to be around one-third of the total number of features.

Gradient Boosting Regression Tree (GBRT) is adapted from gradient descent method and aims at improving learning methods by combining many weaker learners, which are created sequentially, in attempt to produce a strong learner. In GBRT, each tree learns from the residuals of all previous trees with the calculation of negative gradient to update estimation. Since we have a huge number of observations in the training set, we adopt stochastic gradient method. In each iteration, the learners update the full model by taking random sample from the full amount of data without replacement. It not only reduces the amount of calculation, but also effectively prevents over-fitting, achieving two birds with one stone.

3.3.2 Trees with histogram algorithm

XGBoost is an optimized distributed gradient boosting library designed to be highly efficient, flexible, and portable. It implements machine learning algorithms under the Gradient Boosting framework. LightGBM is also a gradient boosting framework that uses tree-based learning algorithms. It is designed to be distributed and efficient with the following advantages: faster training speed and higher efficiency, lower memory usage, better accuracy, support of parallel, distributed, and GPU learning and capable of handling large-scale data.

Both XGBoost and LightGBM supports histogram algorithm to achieve faster training speed. The basic idea of the histogram algorithm is to discretize the continuous floating-point values into k integers and construct a histogram with a width of k at the same time. When looping the data, each discretized value is regarded as an index to accumulate statistics in the histogram. After looping the data once, the histogram accumulates the required statistics, and then loop again to find the optimal value according to the discrete value of the histogram, which is just the splitting point.

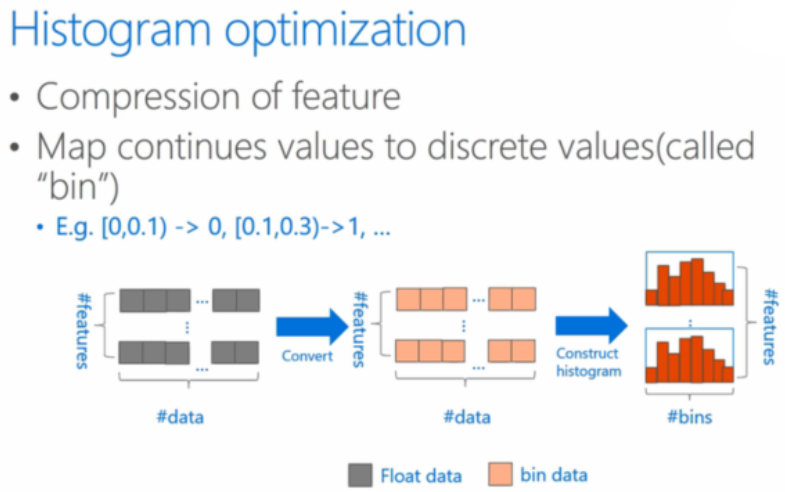


Figure 5: A simple description of histogram algorithm

Moreover, these 2 methods can support NA values directly, and we can avoid bringing in extra information and just leave those blanks as they are.

3.4 Neural network

Neural network consists of three parts: input layer, hidden layer, and output layer. The number of hidden layers may be 0 for simple task but hundreds or thousands in complicated methods and models incorporate more flexible predictive associations by adding hidden layers. The nodes of each layer in the model are called "neurons". The neurons located in the input layer correspond to the characteristics of the training data. The neurons in the hidden layer and the output layer are expressed by the activation function, which is ReLU according to the paper. As the depth of the network increases and the number of nodes at each layer increases, the expressive ability of the network can be strengthened. The higher the complexity of the network, the stronger the expressive ability and the more complex models can be expressed. We can also see that the learning of the network is the learning of the connection weights and thresholds between each node in the network, that is, to find the optimal connection weights and thresholds so that the model can reach the optimal (generally local) solution.

In our work, we mainly use 2 hidden layers to achieve the balance of training speed and optimal value of loss function. Each layer has 64 neurons, since the recommended number of neurons is 2 to the n-th power.

4 Result analysis

In our replication work, we use linear regression model, including simple linear regression, elastic net, principal component regression (with and without Huber loss) and partial least squares (PLS), as well as ensemble trees (RandomForest, Gradient Boosting, XGBoost, and LightGBM) and neural network (MLP), a total of 12 different models to find the specific relationship between the return and the characteristics.

Due to the discussion in section 3.1, the prediction work is conducted in the following parts. First, the future return is predicted by only the given characteristics in the original dataset, and then we add the lag and rolling features as well as the interaction ones in trees and neural network to see if there is any improvement on the prediction result.

4.1 Out-of-sample R^2 score

We will look at the R^2 score first. Scores of all machine learning models are given by the mean value of all the 21 training epochs, and in each epoch, the result is given by averaging all the annual R^2. Detailed result is shown in Table 1. Unfortunately, we only manage to get some positive R^2 scores that are slightly over 0 in some training periods, which means the estimation is highly far away from the ground true value. Performance of top and bottom stocks ranked by their market values is also compared in our work. Figure 6 below gives the average score among different models. We can see that when only considering the ordinary variables with the total number around 100, the complex models outperform the simple linear methods except ordinary Elastic Net and, Partial Least Square as the average out-of-sample scores of the 2 models are all greater than 0. Using PCA for dimension reduction does not give better performance on linear models in our test, which does not correspond to what we have assumed before modelling. Moreover, when we look at the average value of each training epoch, we find that for most of the time, the first 3 training results are the worst among all scores, and we tend to believe the average performance might be lowered down by these poor prediction results.

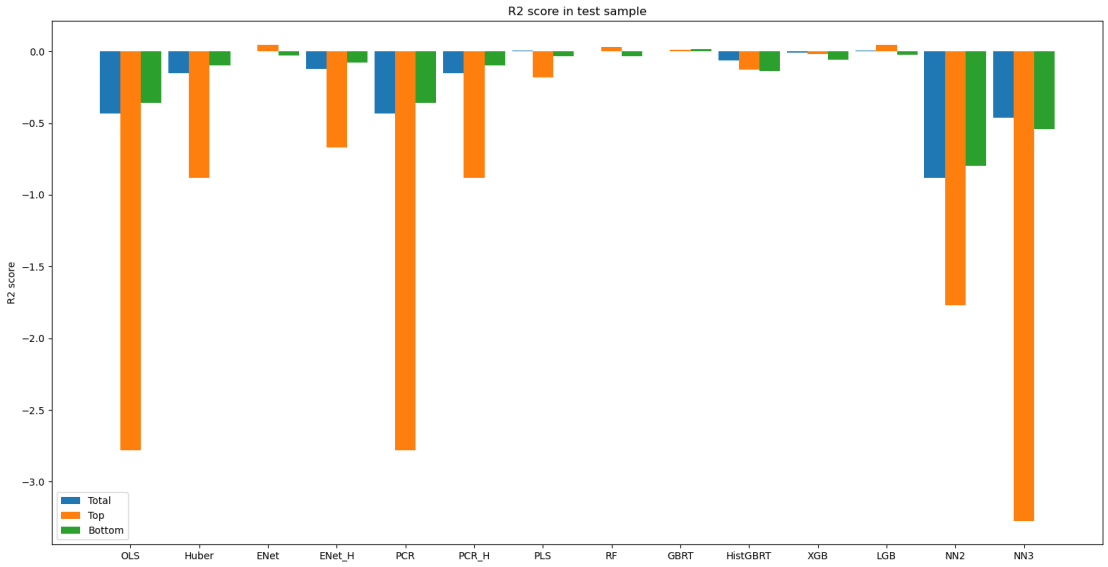


Figure 6: overall out-of-sample R^2 score

Linear	OLS	Huber	ENet	ENet+H	PCR	PCR+H	PLS
All	-0.4353	-0.1501	0.0012	-0.1219	-0.4353	-0.1502	0.0060
Top	-2.7829	-0.8838	0.0432	-0.6719	-2.7829	-0.8843	-0.1841
Bottom	-0.3594	-0.0962	-0.0265	-0.0759	-0.3594	-0.0962	-0.0337
Tree/NN	RF	GBRT	HistGBRT	XGB	LGB	NN2	NN3
All	-0.0006	0.0015	-0.0163	-0.0110	0.0034	-0.8844	-0.4627
Top	0.0298	0.0101	-0.0277	-0.0204	0.0404	-1.7708	-3.2727
Bottom	-0.0361	0.0149	-0.0519	-0.0564	-0.0107	-0.8003	-0.5436

Table 1: Average out-of-sample R^2 score of all machine learning models

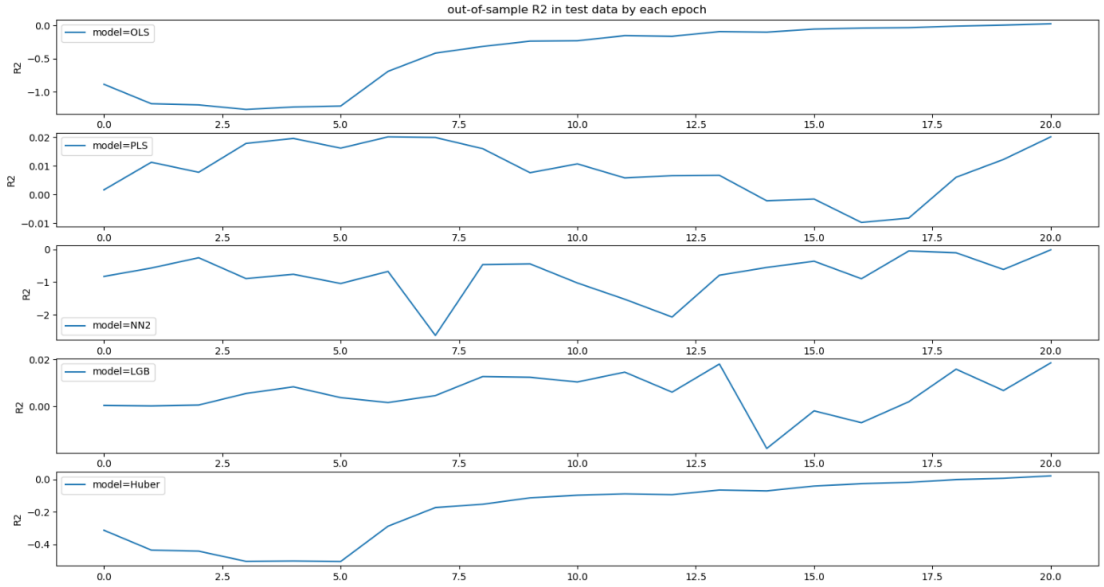


Figure 7: out-of-sample R^2 score by each training epoch for 5 models

We also calculate the final R^2 by preserving the 90% of the dataset, adding interaction, lags and rolling windows into the data and using the original one without dealing with the missing values. To reduce the training time, we only compare the results by adding features on 2 linear models, ElasticNet and Partial Least Square, tree methods XGBoost and LightGBM and neural network with 2 and 3 layers, respectively. Due to the limitation of our local device, we cannot merge all these newly generated features into one dataset, and we only test the performance separately. Table 2 and figure 8 describe the average out-of-sample R^2 after the total observations of the dataset are reduced to 90%.

Linear	OLS	Huber	ENet	ENet+H	PCR	PCR+H	PLS
All	-0.4511	-0.1864	0.0017	-0.1326	-0.4511	-0.1864	0.0088
Top	-3.8377	-1.1498	0.0469	-0.7505	-3.8377	-1.1504	-0.1404
Bottom	-0.3552	-0.1039	-0.0269	-0.0796	-0.3552	-0.1039	-0.0299
Tree/NN	RF	GBRT	HistGBRT	XGB	LGB	NN2	NN3
All	-0.0031	0.0032	-0.0619	-0.0190	0.0055	-0.7869	-0.6187
Top	0.0234	0.0152	-0.1284	-0.0148	0.0454	-6.5580	-4.3651
Bottom	-0.0410	-0.0006	-0.1359	-0.0646	-0.0220	-0.5258	-0.5099

Table 2: Average out-of-sample R^2 score of all machine learning models using 90% of the data

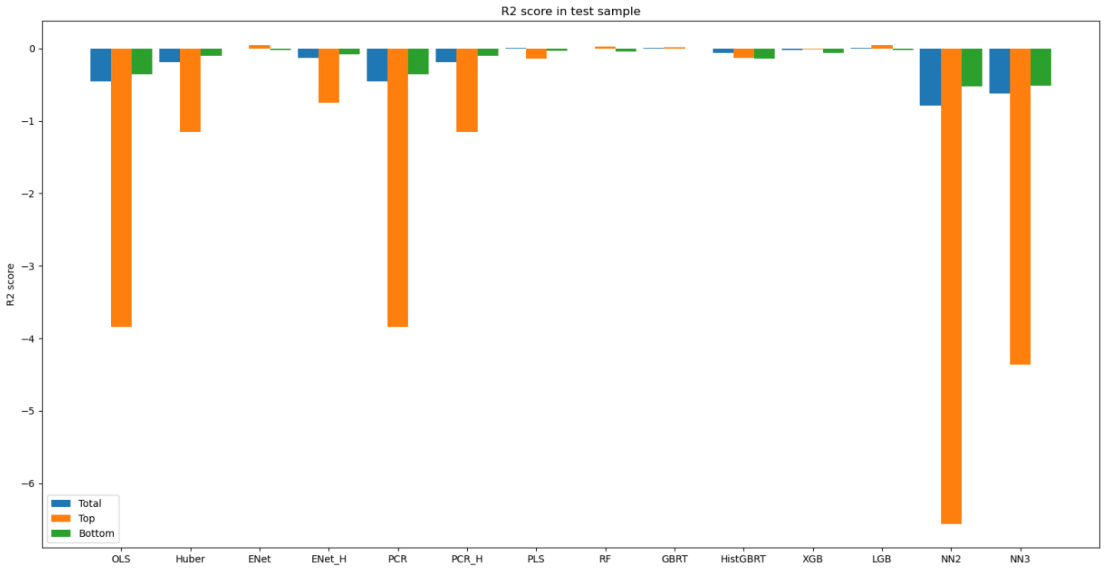


Figure 8: overall out-of-sample R^2 score after preserving 90% of the data

Unfortunately, the result is completely opposite to what we have thought of. The R^2 scores computed by the new dataset are worse than that obtained in the original one for most models especially when we do prediction on the return of stocks with highest market values.

Now we would like to find out if the newly added features help improve the accuracy of our prediction. Table 3 below shows the performance of 2 linear models that perform best among all the linear ones.

	ENet	PLS	ENet+ roll&lag	PLS+ roll&lag	Enet+ interaction	PLS+ interaction
All	0.0017	0.0088	0.0012	0.0048	0.0012	0.0157
Top	0.0469	-0.1404	0.4486	-0.1956	0.0449	0.072
Bottom	-0.0269	-0.0299	-0.0249	-0.0083	-0.0249	-0.0175

Table 3: Average out-of-sample R^2 score of Elastic Net and Partial Least Square

The newly added features fail to improve the prediction result on Elastic Net, but the scores of PLS rise a little. Then we will look at whether the performances of XGBoost and LightGBM are affected.

NA imputed	XGBoost	LGBM	XGBoost+ roll&lag	LGBM + roll&lag	XGBoost+ interaction	LGBM + interaction
All	-0.0110	0.0055	-0.017	0.0046	-0.105	0.0088
Top	-0.0204	0.0454	-0.0148	0.0432	0.0023	0.0556
Bottom	-0.0564	-0.0220	-0.0509	-0.0102	-0.0469	-0.0155
NA contained	XGBoost	LGBM	XGBoost+ roll&lag	LGBM + roll&lag	XGBoost+ interaction	LGBM + interaction
All	-0.0175	0.0049	-0.0231	0.006	-0.0168	0.0072
Top	0.0015	0.0508	-0.0155	0.0563	0.0031	0.0601
Bottom	-0.0463	-0.0146	-0.062	-0.0145	-0.0217	-0.0086

Table 3: Average out-of-sample R^2 score of XGBoost and LightGBM

It is hard to tell whether the rolling windows and lags help the prediction or not since the changes are quite small. But the interaction features do help improve the performance, corresponding to our initial thinking that trees can learn better from the relationship between variables. Also, when we look at the 2 datasets that whether they contain NA or not, the difference between them is quite small. We think the method of imputation might be reasonable by only looking at the data inside one month to avoid using future information.

4.2 Feature Importance

For feature importance, we simply use the absolute value of each coefficient in the linear model to represent the importance of each variable. As for tree-based methods, we simply drop one feature by another to compare the R^2 score so that we can determine the significance, and it has been done in scikit-learn package. We then show the top 25 important features generated by each model. Figure 11 below displays the normalized feature importance of Gradient Boosting. We can see that the generated macroeconomic features have higher scores in the importance ranking. The overall feature importance of all models is given in figure 12. The macroeconomic features always have higher importance in model prediction, especially the 'svar', 'dfy', 'd/p' and 'tbl'. PCA seems to regard the variable 'age' as the most siginificant indicator for prediction, but the R^2 scores for PCR and PCR with Huber loss do not improve compared to the ordinary ones.

We also calculate the importance of the created features. Figure 13 and 14 shows the top-25 normalized feature importance of the interaction factors only and all the factors used in the dataset, respectively. Figures 15 and 16 together display whether the rolling window and lag factors are important by LGBM. We can see that the interaction between macromedia features and stock-specific factors like momentum, bid-ask spread, Beta and trading volume is considered the best variables for prediction, and they enter the top vital features for prediction. The rolling mean of the stock return in the past 3 months and the lag of it in one month seem to be the best time-series features, but they hardly beat the macroeconomic variables in the significance ranking. It helps to explain why there is barely no improvement in prediction when we add the time-series features into thee dataset.

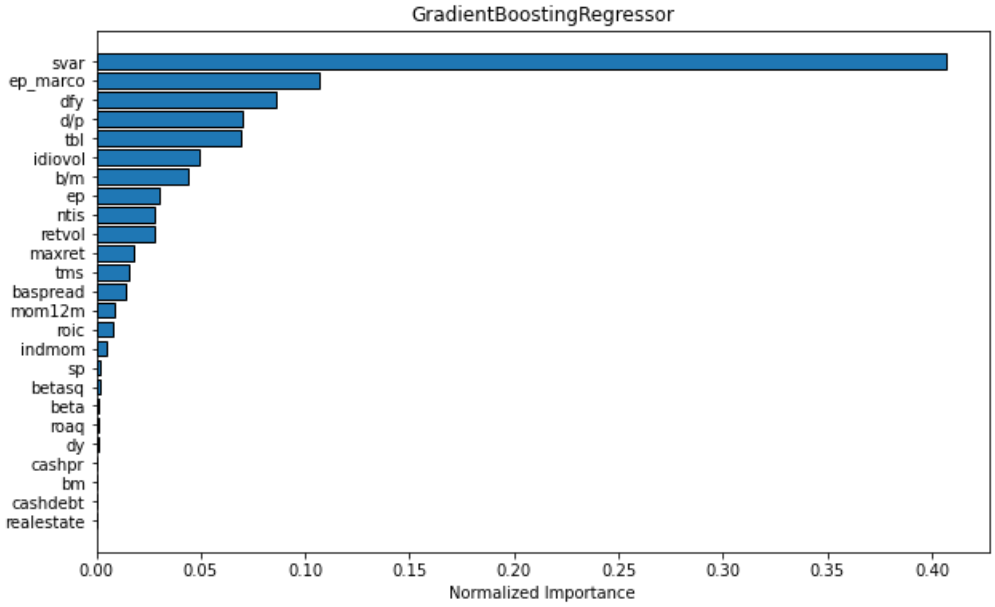


Figure 11: top 25 important features after normalization in Gradient Boosting Tree

Figure 12: Overall feature importance of all models

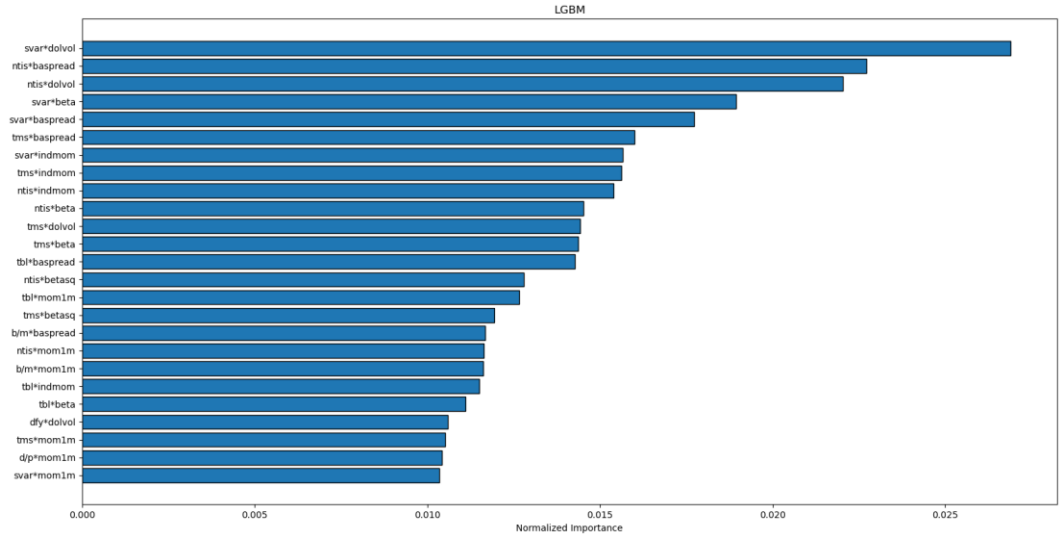


Figure 13: top 25 interaction features after normalization in LGBM

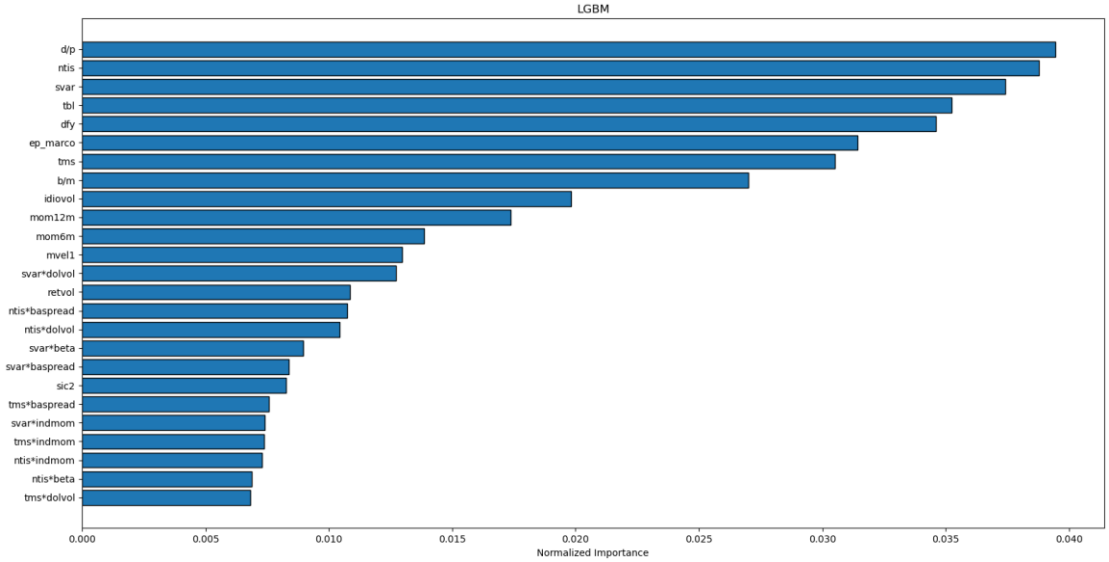


Figure 14: top 25 features including interactions after normalization in LGBM

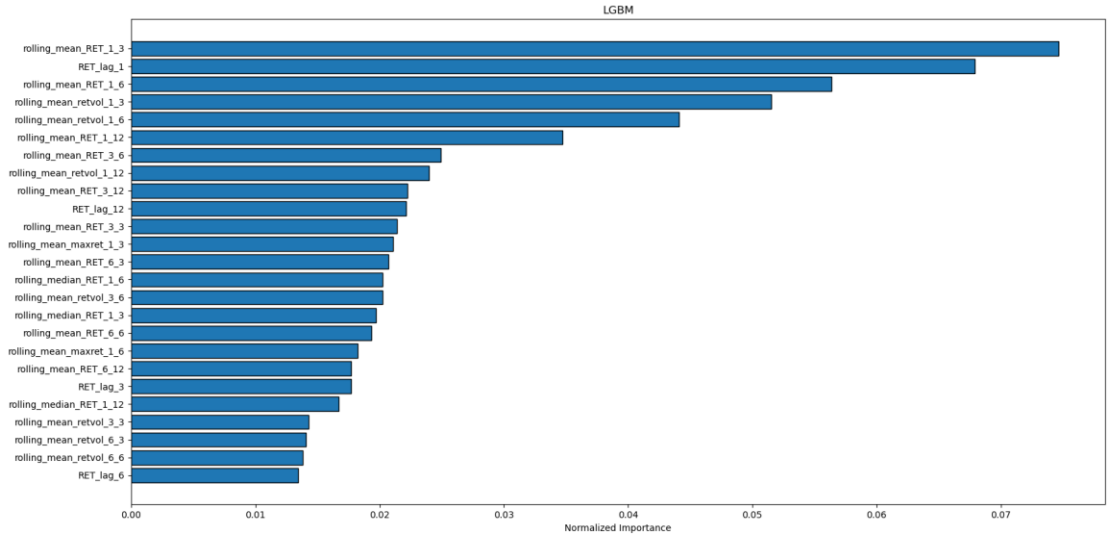


Figure 15: top 25 features of rolling windows and lags after normalization in LGBM

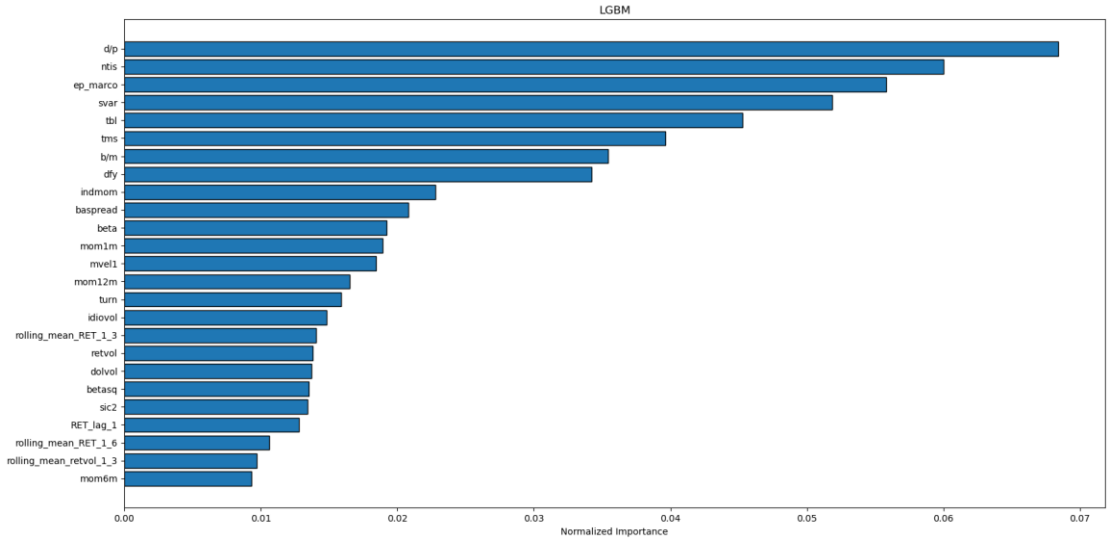


Figure 16: top 25 features including the rolls and lags after normalization in LGBM

We also conduct the Diebold-Mariano Test for all R^2 scores from our models. Since the R^2 scores are not satisfactory, we find that there are some outliers in the final Diebold-Mariano statistic, which means the DM values in some specific years are extremely large compared to that in other periods. We decide to omit these values to obtain the final comparison result. Figure 15 below shows the DM values of all machine learning methods used. We use the models whose names are shown in the rows as the benchmark and use all methods in the columns to make comparisons. The DM value is calculated by the average annual error of each machine learning model, and thus negative values means that models in the column outperform the models in the rows. We can see that the all the neural networks perform poorly among all methods, and PLS seems to be the best model overall if we only consider simple features, verifying what we have seen by comparing the out-of-sample R^2 score.

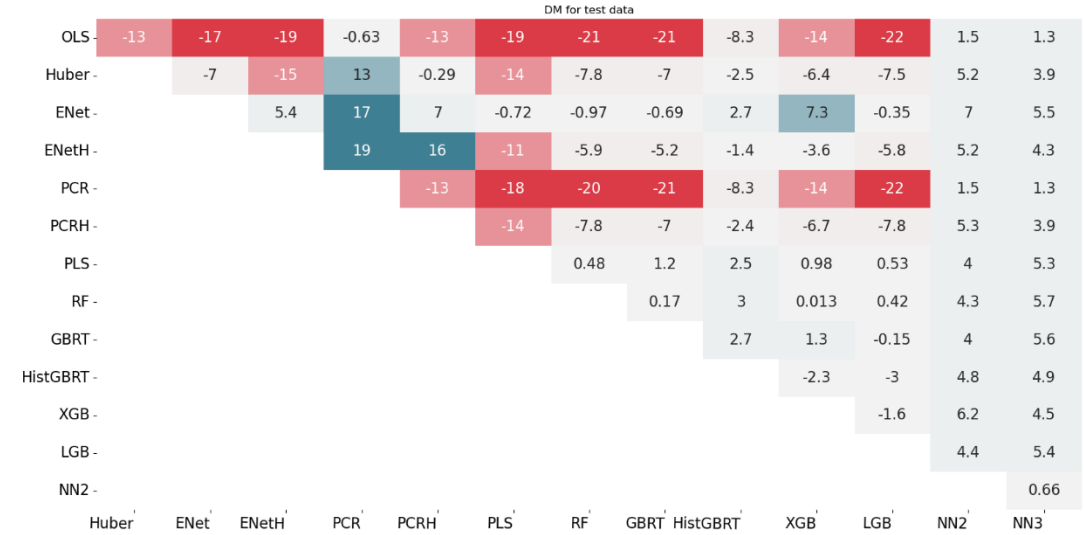


Figure 17: DM result for test dataset

5 Conclusion, future work, and acknowledgement

We conduct this simple replication work with various methods and obtain some results given by our machine learning methods. However, some models themselves cannot correctly predict the response

values such as the ordinary Elastic Net, whose prediction is always a constant number and thus the feature importance of all variables is 0. The result is always a constant value for each individual stock no matter how we tune the hyperparameters. We may consider following steps to improve our model.

First, we can adopt business knowledge and practice into our model so we can understand what is crucial to the final judgment instead of only looking at the information table.

Second, we may try to find new ways of dealing with data. For the missing data, we may look deeper into the inner relationship like what we have done for category and stock names. For example, the variable age, which is defined as the number of years since first COMPUSTAT coverage, meaning that it may be linearly related to year. In Shihao et al.'s work, variable age does not have the least importance among all variables, but in our test, the R^2 score does not change much when we remove it from our dataset. Besides, we can try time-series imputation methods or deep learning imputation methods.

Third, we may introduce RNN or LSTM to predict the returns. Multi-layer perceptron seems to have poor performance in this task, but we tend to believe the more complex and flexible neural networks may study the interaction between the characteristics and response better compared to simple linear ones. Due to the limitation of devices, we cannot adopt them here.

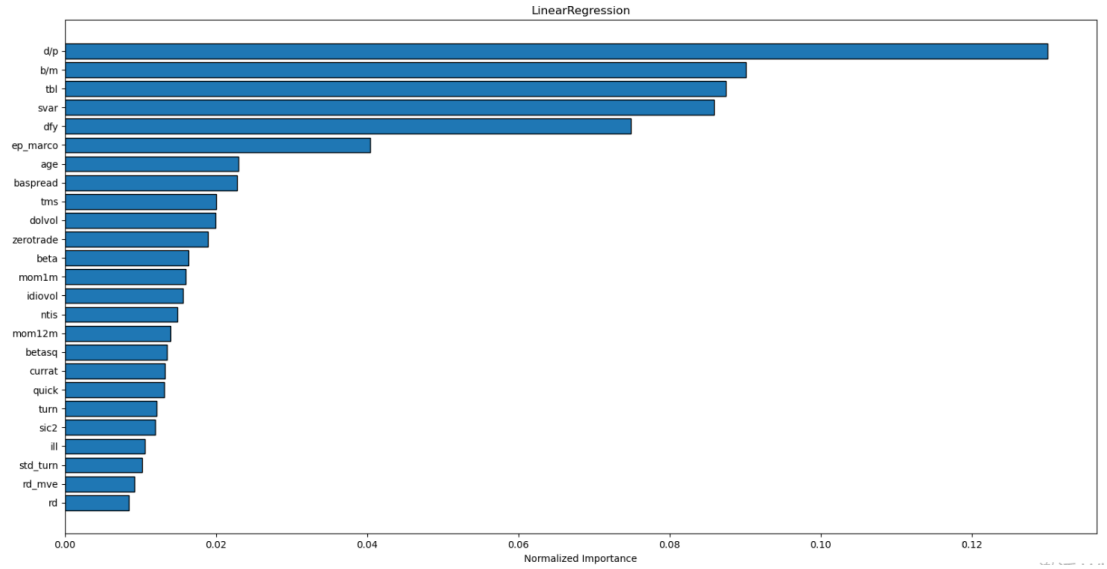
Thanks to Prof. Yao and Prof. You who give me the opportunity to participate in the independent project. Besides, I want to give my appreciation to my classmate Luo Jiahao and Yang Yannan who help me a lot during the task. I also want to express my thanks to students who have taken the course MAFM6010Z, from whose work I have learned a lot. Finally, thanks to those who have shared their experience online!

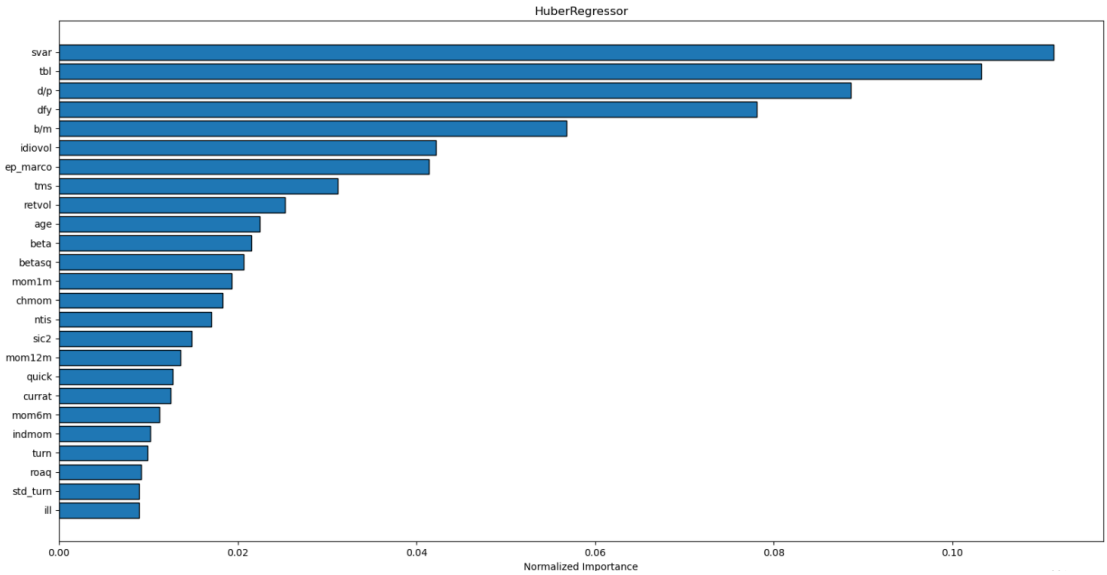
Reference

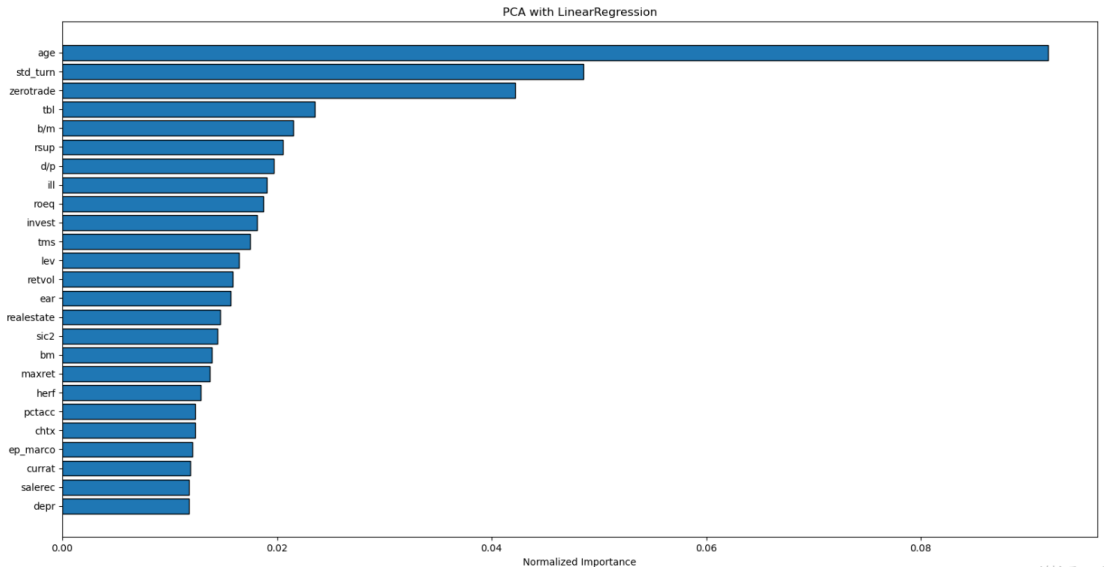
[1] Echo Sun, Diebold Mariano Test Package, https://github.com/echosun1996/DieboldMarianoTest
[2] Kaan Yolsever, Empirical Asset Pricing via Deep Learning Algorithms, https://github.com/yolsever/ML-in-equity-prediction
[3] Homepage for Artificial Intelligence in Machine Learning, https://yao-lab.github.io/aifin/2021/

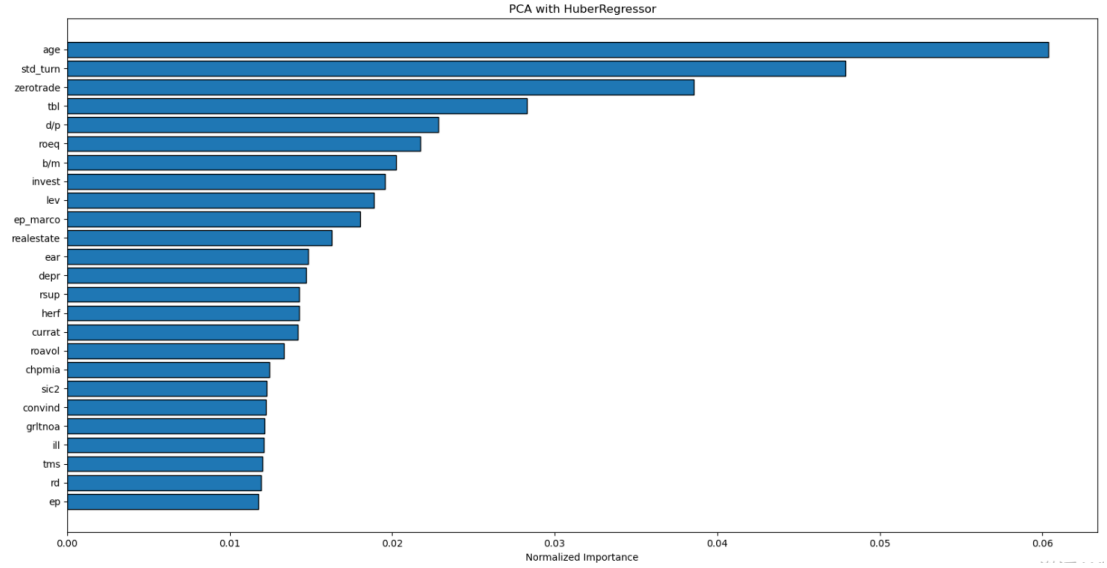
Appendix

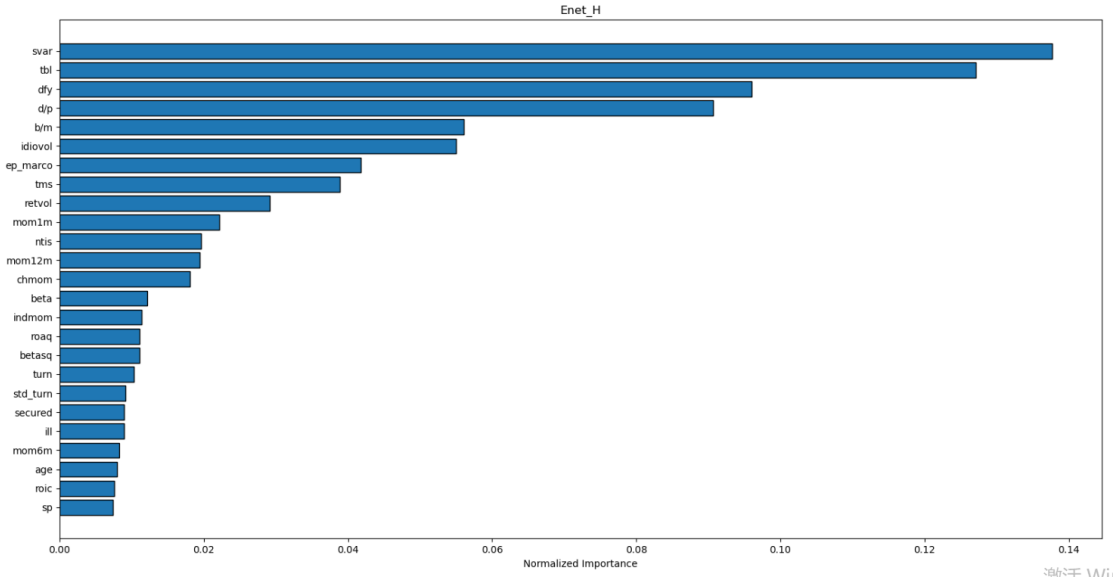
Figures of Feature Importance for all Machine Learning Models

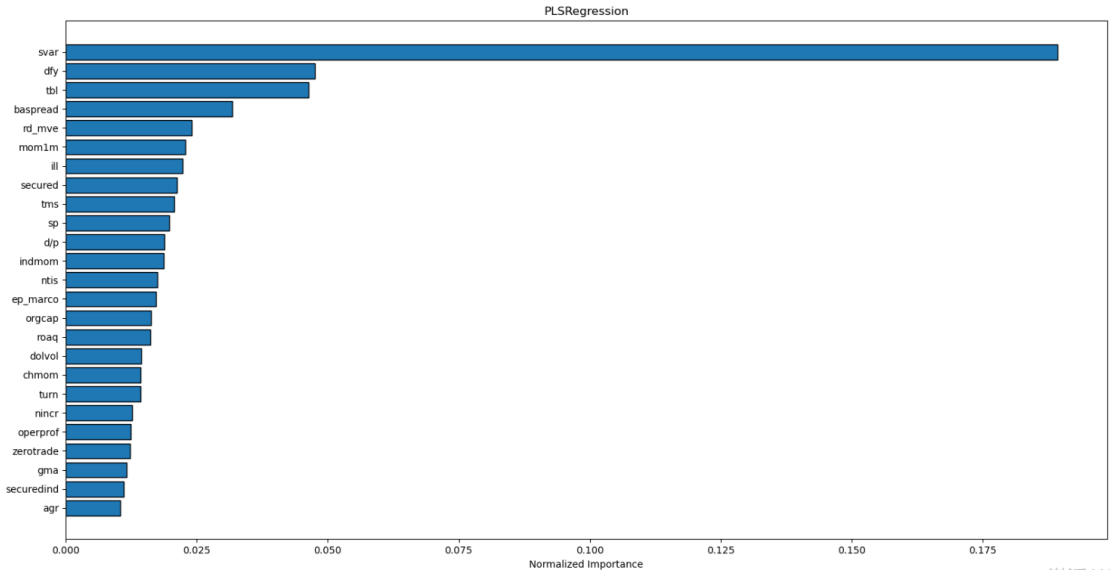


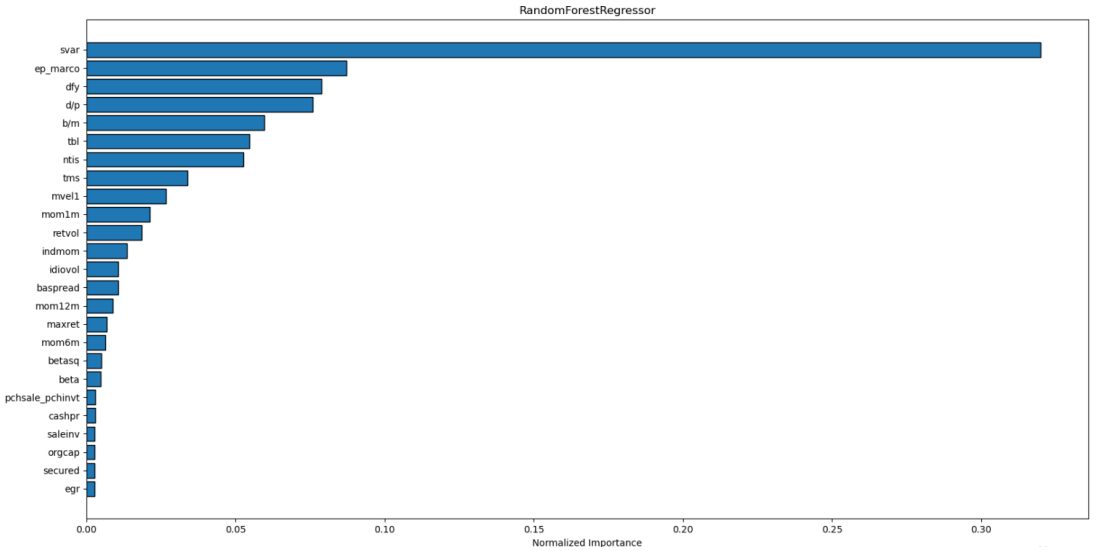


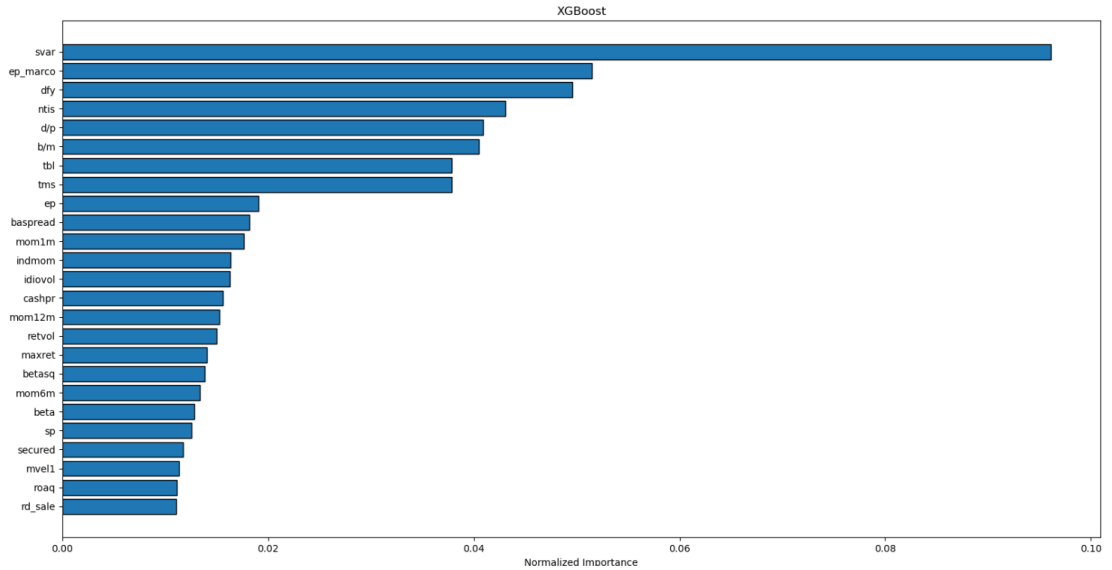

PCA with LinearRegression
age
std_turn
zerotrade
tbl
b/m
rsup
d/p
ill
roeq
invest
tms
lev
retvol
ear
realestate
sic2
bm
maxret
herf
pctacc
chtx
ep_marco
currat
salerec
depr
0.00
0.02
0.04
0.06
0.08
Normalized Importance

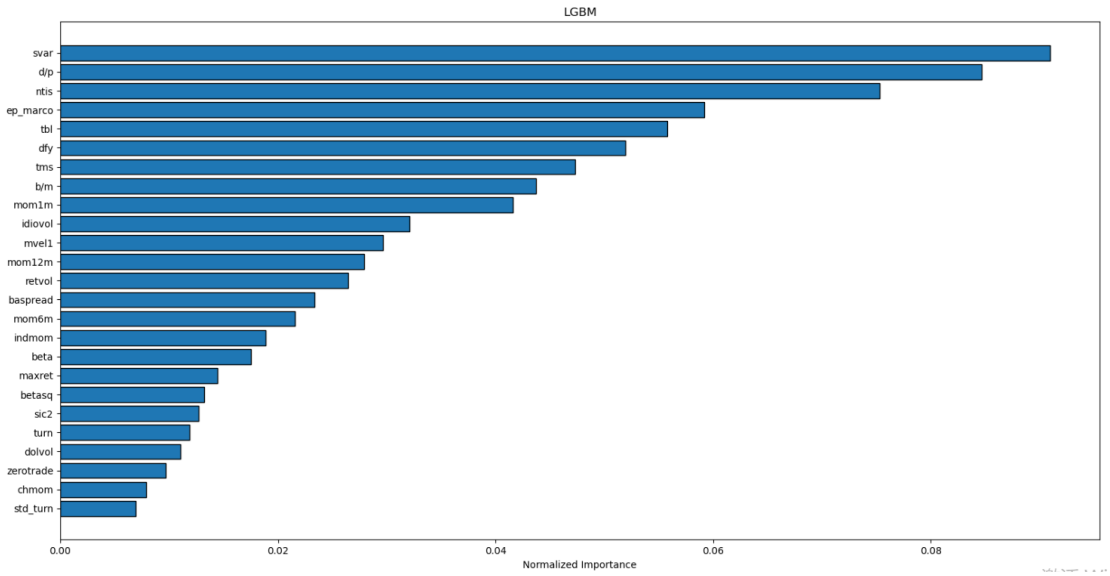

PCA with HuberRegressor
age
std_turn
zerotrade
tbl
d/p
roeq
b/m
invest
lev
ep_marco
realestate
ear
depr
rsup
herf
currat
roavol
chpmia
sic2
convind
grltnoa
ill
tms
rd
ep
0.00
0.01
0.02
0.03
0.04
0.05
0.06
Normalized Importance

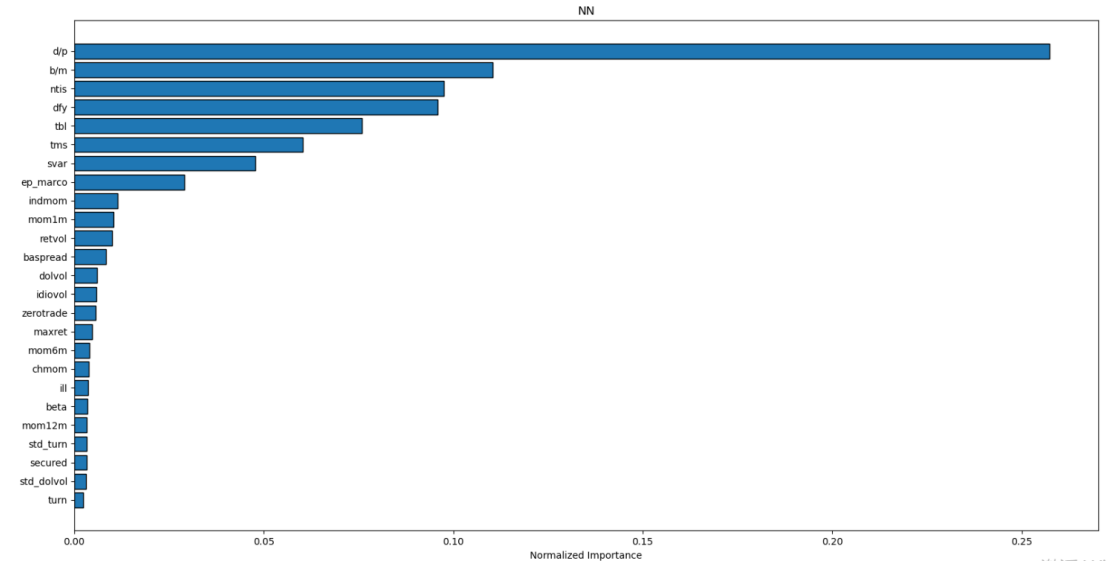

Enet_H
svar
tbl
dfy
d/p
b/m
idiovol
ep_marco
tms
retvol
mom1m
ntis
mom12m
chmom
beta
indmom
roaq
betasq
turn
std_turn
secured
ill
mom6m
age
roic
sp
0.00
0.02
0.04
0.06
0.08
0.10
0.12
0.14
Normalized Importance


PLSRegression
svar
dfy
tbl
baspread
rd_mve
mom1m
ill
secured
tms
sp
d/p
indmom
ntis
ep_marco
orgcap
roaq
dolvol
chmom
turn
nincr
operprof
zerotrade
gma
securedind
agr
0.000
0.025
0.050
0.075
0.100
0.125
0.150
0.175
Normalized Importance


RandomForestRegressor
svar
ep_marco
dfy
d/p
b/m
tbl
ntis
tms
mvel1
mom1m
retvol
indmom
idiovol
baspread
mom12m
maxret
mom6m
betasq
beta
pchsale_pchinvt
cashpr
saleinv
orgcap
secured
egr
0.00
0.05
0.10
0.15
0.20
0.25
0.30
Normalized Importance


XGBoost
svar
ep_marco
dfy
ntis
d/p
b/m
tbl
tms
ep
baspread
mom1m
indmom
idiovol
cashpr
mom12m
retvol
maxret
betasq
mom6m
beta
sp
secured
mvel1
roaq
rd_sale
0.00
0.02
0.04
0.06
0.08
0.10
Normalized Importance


LGBM
svar
d/p
ntis
ep_marco
tbl
dfy
tms
b/m
mom1m
idiovol
mvel1
mom12m
retvol
baspread
mom6m
indmom
beta
maxret
betasq
sic2
turn
dolvol
zerotrade
chmom
std_turn
0.00
0.02
0.04
0.06
0.08
Normalized Importance


NN
d/p
b/m
ntis
dfy
tbl
tms
svar
ep_marco
indmom
mom1m
retvol
baspread
dolvol
idiovol
zerotrade
maxret
mom6m
chmom
ill
beta
mom12m
std_turn
secured
std_dolvol
turn
0.00
0.05
0.10
0.15
0.20
0.25
Normalized Importance