Machine Learning in Financial Data Prediction

MTH9899: DATA SCIENCE II

Zhihao Chen

Xinlu Xiao

Deyun Li

Baruch College

May 26, 2017

Abstract

In this project, we trained and predicted stock price data with 28 input features with machine learning techniques. Various models in three categories are explored: regression, tree and neural network. Considering the time series feature of our data, cross validation is carried out in a sequential scheme. We found that data cleaning and parameter tuning are key to model fitting. Our result shows that neural network would give us the most consistent and robust results.

Contents

1 Data description

Our analysis relies on the proprietary data provided by Seven Eight Capital. The dataset we use consists of 141163 rows of data with 31 features.

Rows

- Each row represents an observation. It is your discretion whether to keep them all, consider a subset, or maybe even enlarge it.

Columns

- Timestamps: the numbers are ordered.
- Id: different numbers represent different financial assets
- 'x': There are 27 cols with names that start with 'x'-those will be the features that you might consider using for your models
- 'y': dependent variable that your model will be predicting
- 'weight': weights associated with each observation

We introduce the following operations to do the **data cleaning**:

- MAD: We use MAD method to detect the outliers.
- Standard Scaler: StandardScaler is a function in the sklearn.preprocessing package, it standardizes features by removing the mean and scaling to unit variance. Actually, it is very similar to normalization.

We've tried three methods to **select features**:

- LassoCV: it is a function in the sklearn.linear_model package, which aims to iteratively fit along a regularization path, and the best model is selected by cross-validation. Lasso method forces weak feature to have zero as coefficients. L1 regularization produces sparse solutions, inherently performing feature selection.
- PCA: it is a function in sklearn.decomposition, which aims to do linear dimensionality reduction using Singular Value Decomposition of the data to project it to a lower dimensional space. The basic idea when using PCA as a tool for feature selection is to select variables according to the magnitude of their coefficients. PCA seeks to replace less correlated variables by uncorrelated linear combinations of the original variables.
- SelectKBest: it is a function in the sklearn.feature_selection, which aims to select features according to the k highest scores, we can pass the score function into this function. And the method we use is "mutual_info_regression", this score function tells the mutual information for a continuous target, measuring the dependency between the variables. It equals to zero if and only if two random variables are independent, and higher values mean higher dependency.

For Lasso CV Regression method, we use PCA, which helps us select around 18 features among 31 features by using different split points. For Neural Network, we use SelectKBest also, which helps us select among 12-16 features. And for Gradient Boosting method, we use LassoCV,

which helps us select around 12 features.

Data Visualization

We give a rough statistical description for the raw data 'y' by showing the distribution and calculating its mean and standard deviation across timestamps.

Figure 1 Distribution of y

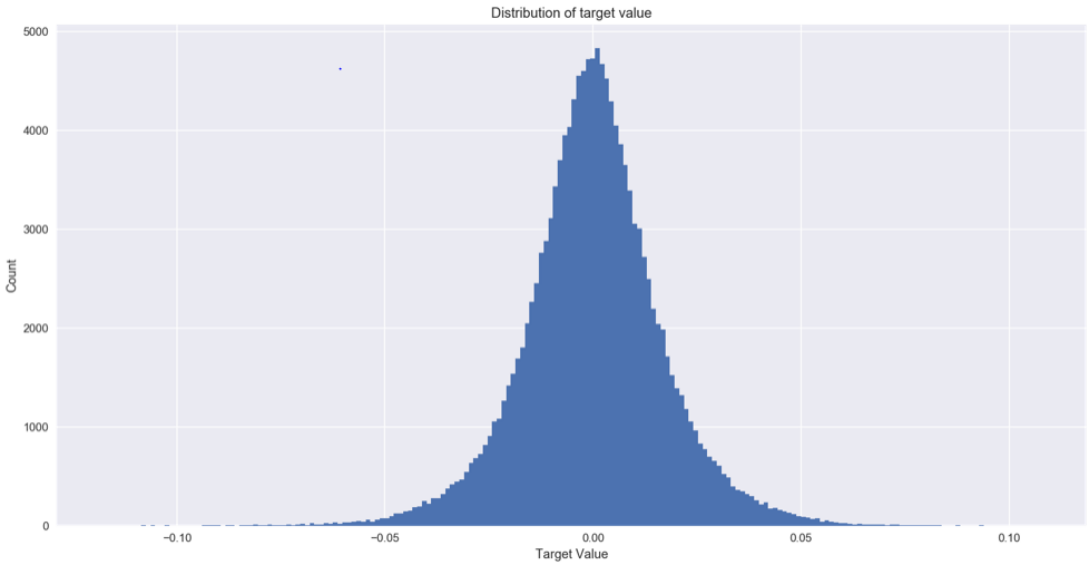
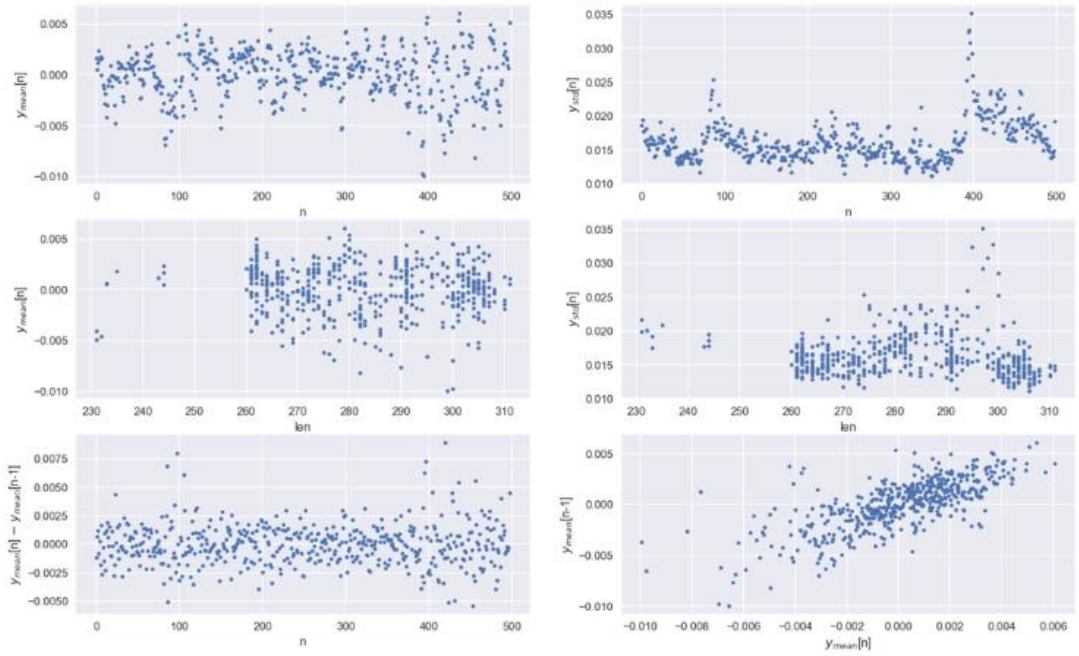


Figure 2 shows the mean and standard deviation across timestamps and for difference length of the timestamps, and also the last two figures in the figure2 shows the difference between each two consecutive y and the correlation of y_t and y_{t-1}.

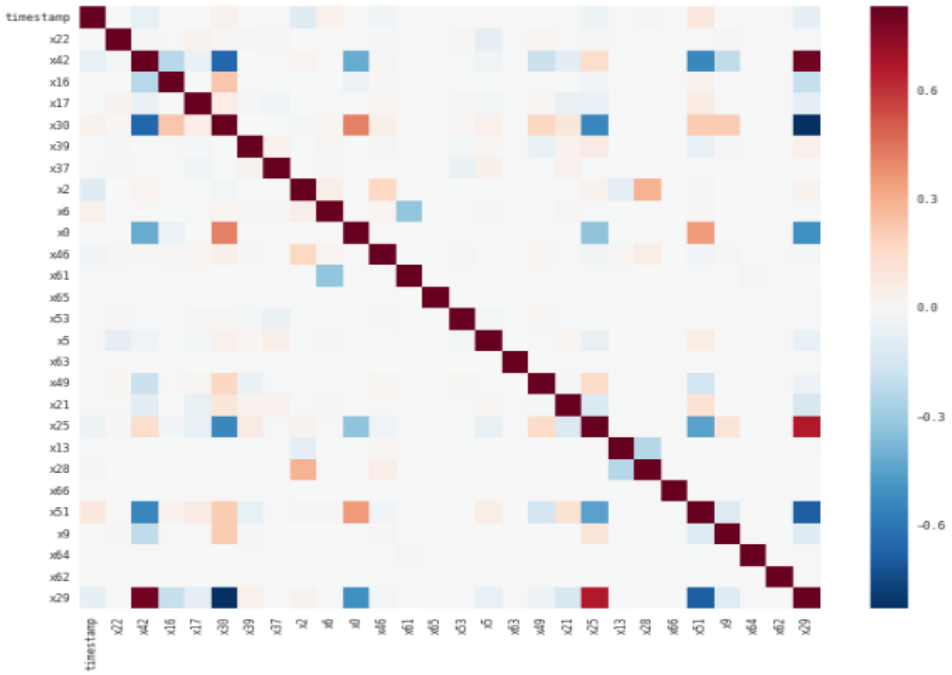
Figure 2 y statistics



2 Methodology

2.1 Regression Methods

Starting from the basic point, we tried fitting the data with regression models. First we plot the correlation matrix graph for all input features.



From the heatmap above, we see that there is little collinearity among our input variables. Thus we neglect collinearity effect in the whole regression process

We have tried four regression methods: LASSO CV, Ridge CV, Stochastic Gradient Descent and Support Vector Regression.

LASSO is a regression analysis method that performs both variable selection and regularization. It aims to minimize

$$\min_{\beta_0,\beta}\left\{\frac{1}{N}\sum_{i=1}^{N}(y_i-\beta_0-x_i^T\beta)^2\right\} \text{ subject to } \sum_{j=1}^{p}|\beta_j|\le t.$$

Ridge tries to alleviate multicollinearity amongst regression predictors by adding a penalty to traditional sum of squared errors. It aims to minimize

$$\sum_{i=1}^{n}(y_i-\sum_{j=1}^{p}x_{ij}\beta_j)^2+\lambda\sum_{j=1}^{p}\beta_j^2$$

In this paper, we use Lasso and Ridge with cross validation to reduce overfitting. Considering that our data is ordered in time and we cannot predict using future data,

the split mechanic for cross validation is also time series split which is a variation of *k*-fold returning first k folds as train set and the *(k+1)*th fold as test set.

Stochastic Gradient Descent (SGD) optimizes the coefficients of input predictors by iteratively minimizing the error of the model on the training data. The sum of the squared errors are calculated for all pairs of input and output. The coefficients are updated towards minimizing the sum of squares iteratively until a minimum sum squared error is achieved or no further improvement is possible.

In this paper, we use SGD with squared loss and L1 penalty.

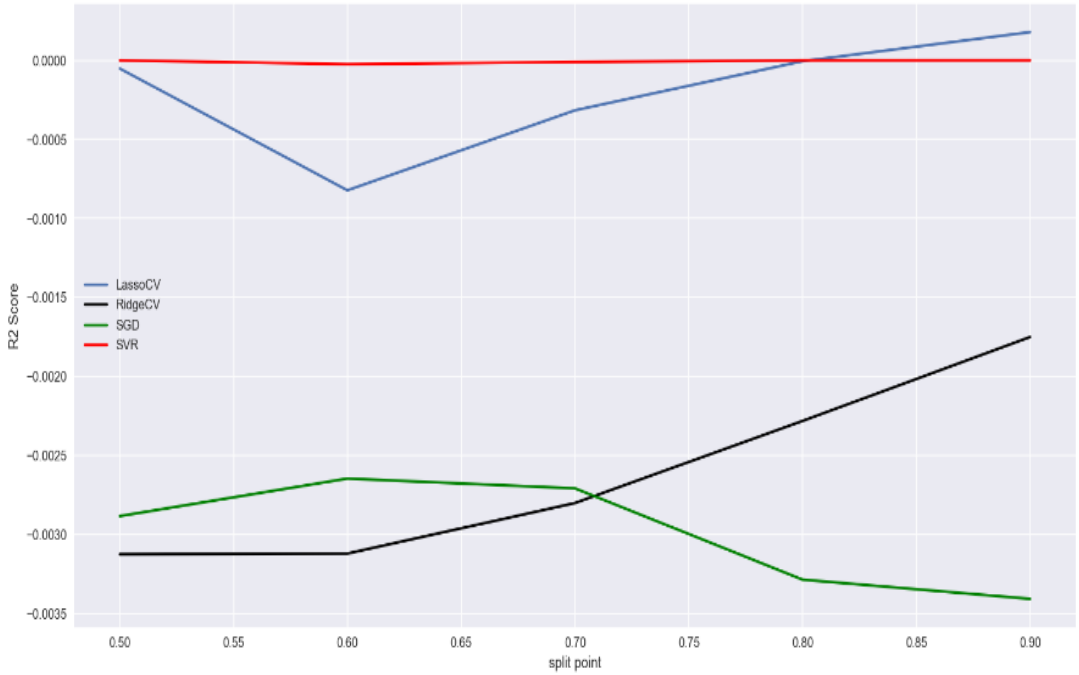
Support Vector Regression (SVR) aims to find a function *f(x)* that deviates from *y* by a value no greater than ε for each training point *x*, and at the same time is as flat as possible.

In this paper, we use SVR with polynomial kernel with degree 3.

Before fitting data into the regression models, we have implemented feature selection with SelectKBest, LassoCV and PCA.

Results show that PCA works better than SelectKBest, while LassoCV selection does not help much to our model fitting.

With PCA feature selection, here's the graph of r2_score vesus train/test data split point for all 4 regression models.



Fixed parameters: number of features selected is 18, splits number for cross validation is 19.

Split point = len(train_set) / len(train_set + test_set)

According to the graph, LassoCV and SVR generate better result than SGD and RidgeCV. With split point 0.9 (train / test = 9 / 1), LassoCV has positive 2.8bps for out-of-sample weighted R2.

Parameter tuning is also a crucial part in training models. We have tested different parameters and use r2_score as the benchmark for performance valuation.

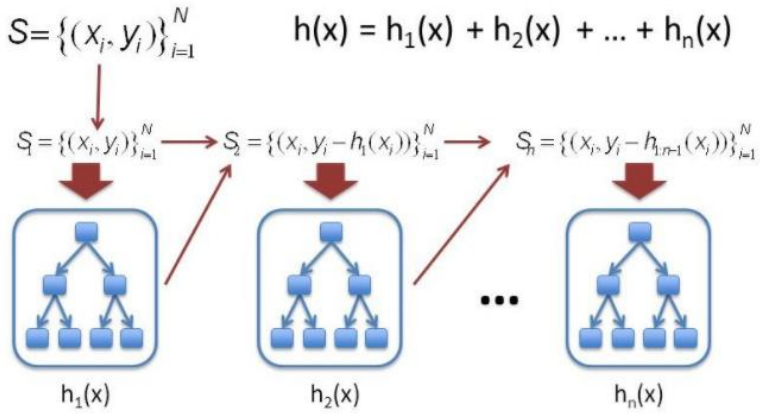
Parameters tuned include number of features selected, number of splits for cross validation in LassoCV. And we found for Lasso CV combined with PCA feature selection, it is optimal to select 18 features and adopt 9 splits cross-validation.

2.2 Gradient Boosting

Basic introduction

Gradient boosting is a machine learning technique for regression and classification problems, which produces as prediction model in the form of an ensemble of weak prediction models, typically decision trees. It builds the model in a stage-wise fashion like other boosting methods do, and it generalizes them by allowing optimization of an arbitrary differentiable loss function.

Figure 3 Simple version explanation of Gradient Boosting



Below shows the algorithm of gradient boosting.

Figure 4 Gradient Boosting Algorithm

Algorithm 1: Gradient_Boost

1 $F_0(\mathbf{x}) = \arg\min_\rho \sum_{i=1}^{N} L(y_i, \rho)$

2 For $m = 1$ to M do:

3 $\tilde{y}_i = -\left[\frac{\partial L(y_i, F(\mathbf{x}_i))}{\partial F(\mathbf{x}_i)}\right]_{F(\mathbf{x})=F_{m-1}(\mathbf{x})}, \; i = 1, N$

4 $\mathbf{a}_m = \arg\min_{\mathbf{a},\beta} \sum_{i=1}^{N} [\tilde{y}_i - \beta h(\mathbf{x}_i; \mathbf{a})]^2$

5 $\rho_m = \arg\min_\rho \sum_{i=1}^{N} L(y_i, F_{m-1}(\mathbf{x}_i) + \rho h(\mathbf{x}_i; \mathbf{a}_m))$

6 $F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + \rho_m h(\mathbf{x}; \mathbf{a}_m)$

7 endFor

end Algorithm

Model implementation

In the model implementation of the Gradient Boosting, we first split data by using different split point into training set and test set. Then we use scaler to normalize the training set of X and test set of X. We first tuned the parameters to select a best set of parameters that give highest r square.

Then we've tried to use feature selection method including LassoCV, PCA and also SelectKBest, comparing the R square of the result, LassoCV turned out to be a good feature selection method for gradient boosting. The comparison result shows as below.

We use lasso to select feature from the training set, and refresh the training set and test set.

Figure 5 Parameter tune result

	max_depth	random_state	min_sample_split	n_estimators	learning_rate	subsample	r2
set1	1	0	5	100	1	1	-0.022914253
set2	1	0	5	100	0.1	1	-0.006173866
set3	1	0	5	100	0.1	0.8	-0.007086942
set4	1	0	5	100	0.08	0.8	-0.006245245
set5	1	0	5	50	0.08	0.8	-0.005253096
set6	1	0	5	50	0.1	0.8	-0.005594153

By this figure we can conclude that for a fixed data split, the below set of parameters could give better R square.

Figure 6 Best parameter set

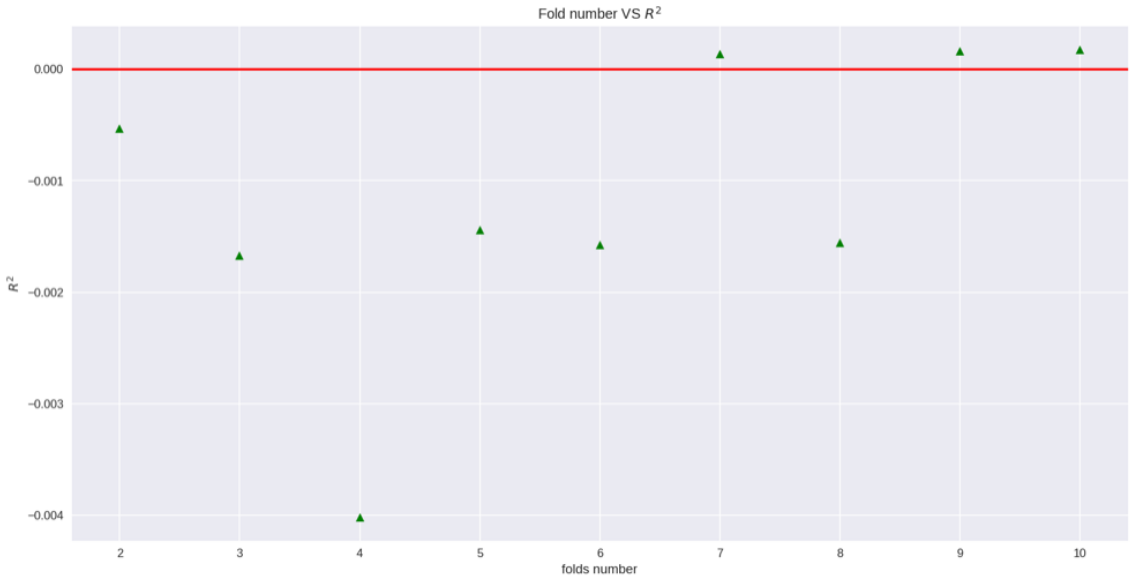
	max_depth	random_state	min_sample_split	n_estimators	learning_rate	subsample
Gradient Boosting	1	0	5	50	0.08	0.8

From the Figure7, fixed the folds number (which represents the number of folds we split, among which the last fold is the test data, and the remaining is the training data), comparing the three methods, Lasso almost give the best R square. For LassoCV, the best split point is fold number 7,9,10. The Figure 8 shows the change of the R square with respect to the number of folds.

Figure 7 Feature selection method

folds number	Lasso		PCA		Kbest	
	R square	feature #	R square	feature #	R square	feature #
2	-0.00056	14	-0.00077	12	-0.02768	12
3	-0.00164	7	-0.00303	12	-0.05338	12
4	-0.00362	8	-0.00497	12	-0.03159	12
5	-0.00139	14	-0.00228	12	-0.02313	12
6	-0.0016	13	-0.0032	12	-0.02555	12
7	0.000134	11	-0.00219	12	-0.06301	12
8	-0.00156	12	-0.00261	12	-0.17792	12
9	0.000161	11	-0.00204	12	-0.03975	12
10	0.000174	11	-0.00168	12	-0.07674	12

Figure 8 Fold number vs R square



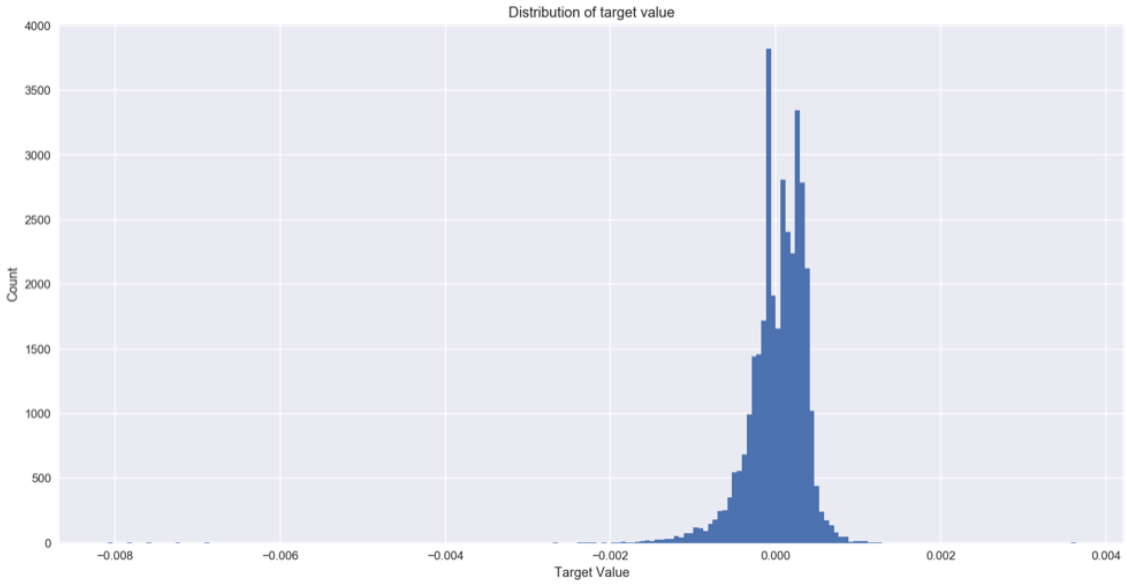
The model based on gradient boosting by using the following model setting gives the best R square as follows:

Figure 9 Best parameter setting for sample data

max_depth	random_state	min_sample_split	n_estimators	learning_rate	subsample	K	R square
1	0	5	50	0.08	0.8	10	0.000174

Below figure is the final model prediction by using the model fitted by the old data, and pass the given new X, and predict the y_predict.

Figure 10 Final model prediction



2.3 Neural Network

Main Packages

We mainly used Keras with Theano Backend, for building neural network model and callback functions. We also used Sklearn for model selection and data preprocessing.

Main Steps

1. We did data cleaning first by MAD method. We choose 4MAD in order to keep enough data points for the following steps. We drop the data points outside the range of $\pm 4\text{MAD}$ (There are 4W+ data points left if 3MAD, 10W+ data points left if we use 4MAD). We later realized dropping outliers will decrease our fitting efficiency, instead we should clip the data.
2. Then we implement feature reduction: we first use StandardScaler to scale the inputs and apply LassoCV to the whole dataset. The dimension is reduced to be 18.
3. After that, we build our neural network model with Keras package, using Adam as the optimizer. We set the input layer size to be 18 and add a hidden layer with k nodes, and tanh, sigmoid, relu as the alternatives of activation function. We also set batch size and epoch as parameters to tune.

We tune the parameters with time series cross validation, which is a rolling based cross validation. To be more specific, we are actually dividing the dataset into 5 layers and 4 folds, and do the folloing:

- fold 1 : training [1], test [2]
- fold 2 : training [1 2], test [3]
- fold 3 : training [1 2 3], test [4]
- fold 4 : training [1 2 3 4], test [5]

We use r2_score of sklearn package as the benchmark to tune all the above parameters.

4. After that, we noticed that the r2 are very unlikely to be positive, and we highly doubt that there might be some structural change in the dataset, the front part of which might not be useful for predicting the future anymore. Therefore, we give our best try to also set size of training test to be a variable. We did sample Division first. Noticing that the holdout dataset size is 34713, we also reserve the most recent 34713 data points as the test data. Then we take n data points as training dataset before those test data. We choose n to be from 1-2 times test data number(i.e. from 34713-69426) and test which n gives us the best positive R^2. For each n, we do Lasso again to the X_train and Y_train, to further shrink the feature space. The feature space is typically shrinked to be between 11 to 15. We build the input layer for the resulting feature space for each n and use the optimized parameters in step 3.

Surprisingly, we find there are more stable positive R2 when we adopt this method. We also find the train size with the largest R2 is n = 55413, with R2 = 15.5 bps

5. Therefore, we use the optimal **n = 55413** to train our model to predict the holdout data

3 Conclusion

We combine the 3 methods to predict the out-sample data notice that neural network does better job than gradient boost and linear regression. So we give greater weight to the neural network. We assign weights of the three models to be 0.1, 0.1, 0.8.