**Don't overfit report**

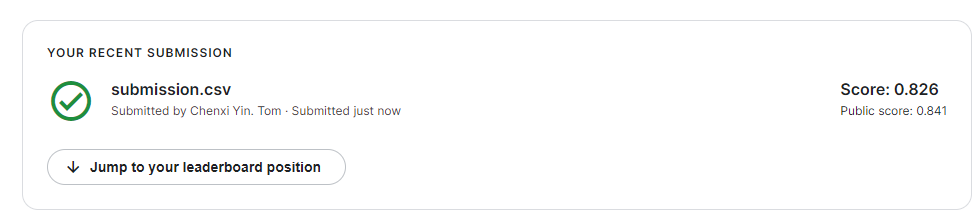
**Comp 4432 machine learning**

**Professor: Dr. Korris Chung**

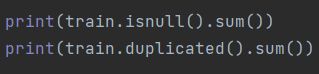
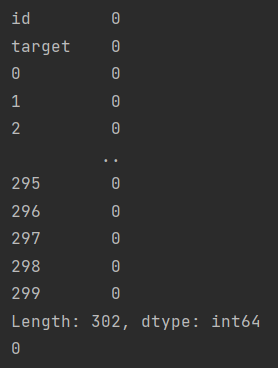
**Yin Chenxi**

**19089951d**

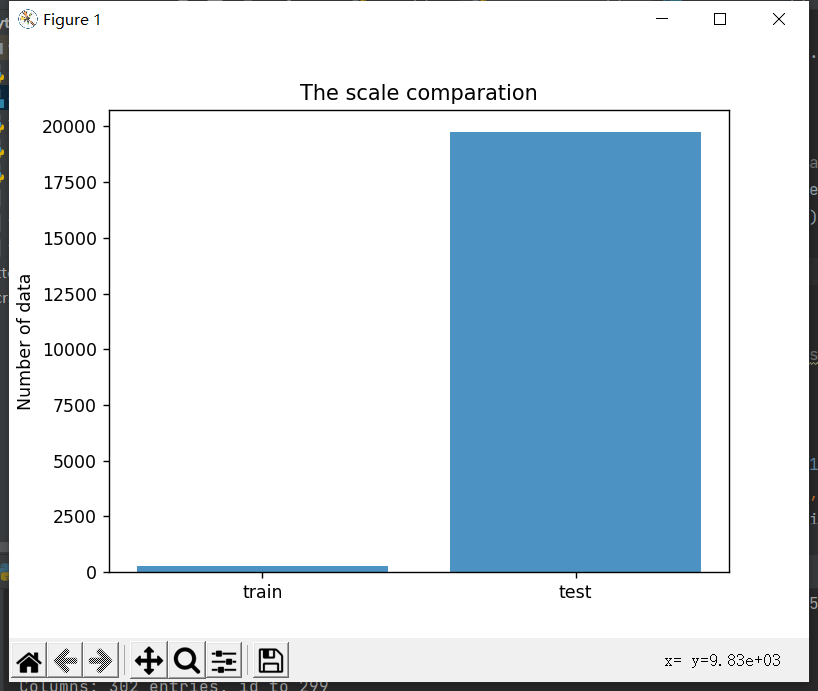
**Team name: Chengxi Yin.tom**

**Final score**

1. **Observe the data**
2. **Check whether there is empty data and duplicated data.**

****

1. **Compare the scale of the test data set and train data set.**



The number of training data is too small. There is a high probability to overfit.

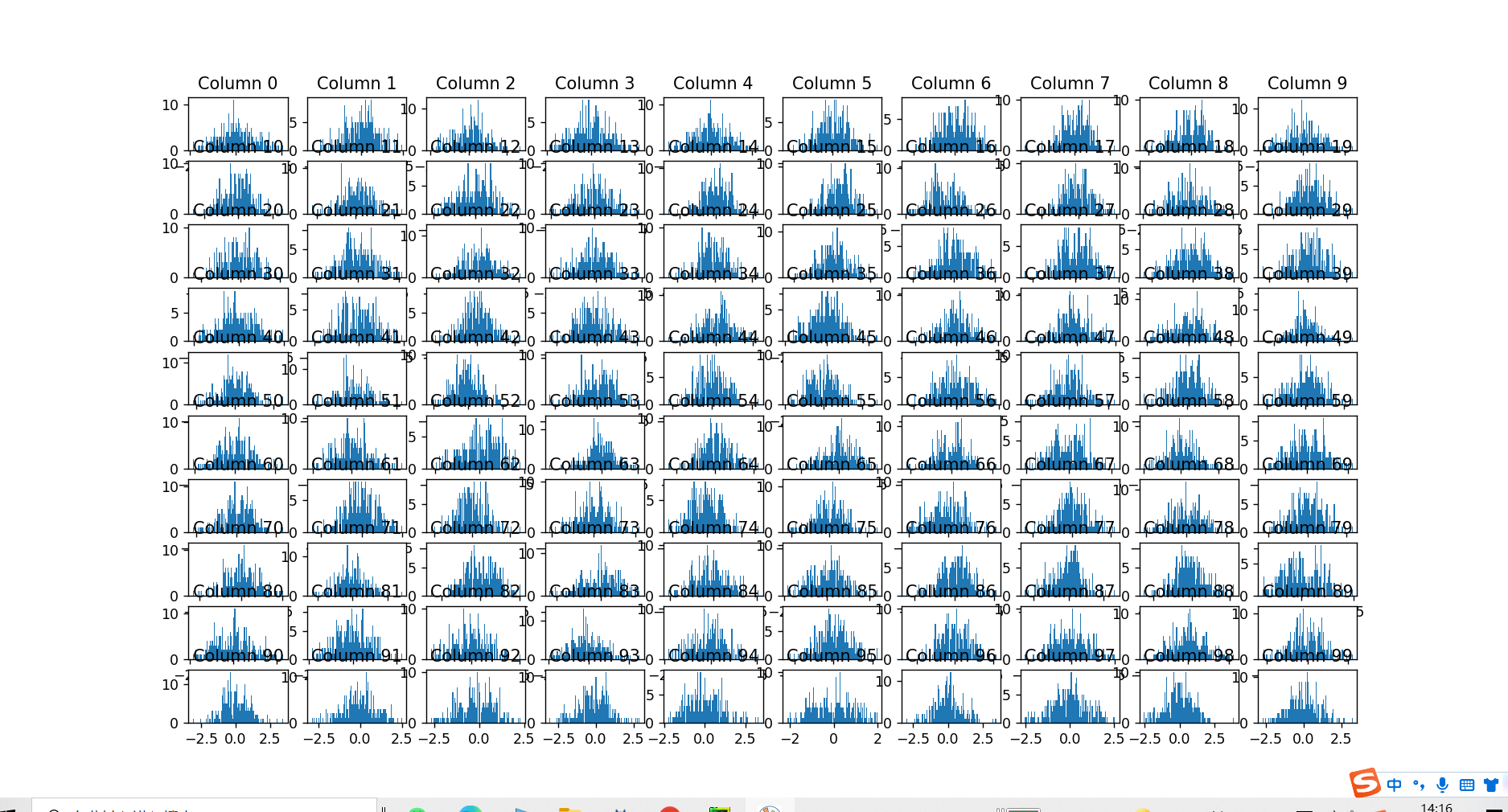
1. **Observe the data distribution(actually is done after the first attempt)**

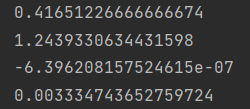
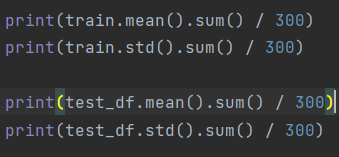
I went back to re-observing the data after trying a simple logistic regression combined with REF, because the predictions were poor.

After seeing the characteristics of the data, I was too eager to select the characteristics, which was my problem and caused a lot of waste of time. I should better observe the data before selecting features.

After looking at the data from multiple angles, I found a surprising implementation where the distribution of the seemingly messy data was regular.

from the hist of each column it shows that data follows a gaussian shape or normal distribution around 0 mean and std =1





Observe the data’s mean and std.

1. **Preparation of the training.**
2. **Add some noise to the training data.**

****

1. **Check the data collinear**

At first I do not know to check the “collinear”

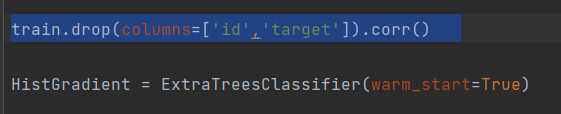
After I read some articles, I know the importance of this.

The corr() function calculates the pairwise correlation of numeric columns, excluding NA or null values. Note that it does not include non-numeric feature columns, such as categorical features. The variation range of the correlation coefficient is from -1 to 1. The closer the coefficient is to 1, the stronger the positive correlation is; the closer the coefficient is to -1, the stronger the negative correlation is; the closer the coefficient is to 0, there is no linear correlation between the two attributes. , note that the absence of a linear correlation does not imply the absence of other nonlinear dependencies. This can draw heatmaps to aid machine learning.

Here is the reference link

https://blog.csdn.net/JT\_WPC/article/details/104603322

Here is my implementation



1. **Select the feachers**

**A. When I found the number is too large and the training data set too small, The first thing that came to my mind was trying to select significant feachers and low the weight for other feachers.**

This is the direct way to avoid overfit.

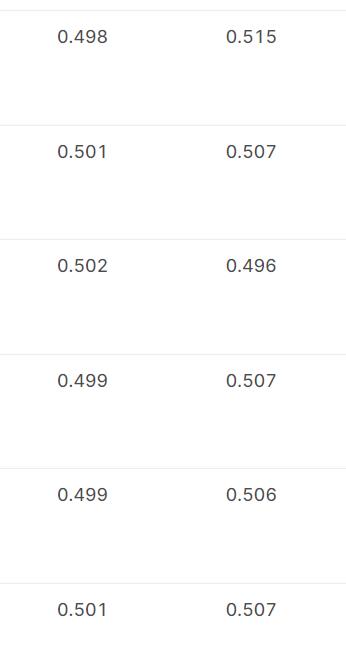
As a newbie, I tried using a linear regression model first.

Ordinary linear regression: The mean squared difference between the actual value and the predicted value of the output model is as small as possible (ie least squares estimation method), but it is easy to fall into overfitting (ie low deviation), and subsequent regression methods will have regularisation method to reduce data.

A simple logistic regression model was then used directly to make predictions. The results were poor.

I used the code from this link:

**https://zhuanlan.zhihu.com/p/41132059**

****

I've tried changing various parameters to no avail. (There are many test cases)

**B. Then I tried to use (RFE + Ordinary Linear Regression)**

I used the code from this link to do the feachers selection.

<https://blog.csdn.net/qq_33704653/article/details/80085116>

RFE is recursive feature elimination regression feature elimination, so that only the most important features of no\_features are retained in the process of regression feature elimination, which can avoid overfitting, but RFE will discard some variables. It is not as good as the following methods to assign weights to variables .

But the problem is the result has no improvement. Instead of the model problem, it might be my problem in the implementation. So I give up dive into this model.

**C. After research, I found that the lasso regression is really suitable for this data set. Here is the reason:**

As shown earlier, our dataset training set is much smaller than the test set, and there are 300 features, but the number of training sets is only 250.

The lasso regression method is derived from data where the number of independent variables is much larger than the number of samples, such as gene locus data. After sequencing, each person may have tens of millions of loci, but the number of people sequenced may be hundreds of cases, so the traditional Cox The forward method of regression, the backward method, the Stepwise method, and the Wald method are not applicable. The situation where lasso regression can be adapted is: the sample size is relatively small, but there are many indicators, that is, the problem of small N and large P. Suitable for high-dimensional statistics, traditional methods cannot deal with such data. This fits the characteristics of our dataset, so lasso regression should be used.

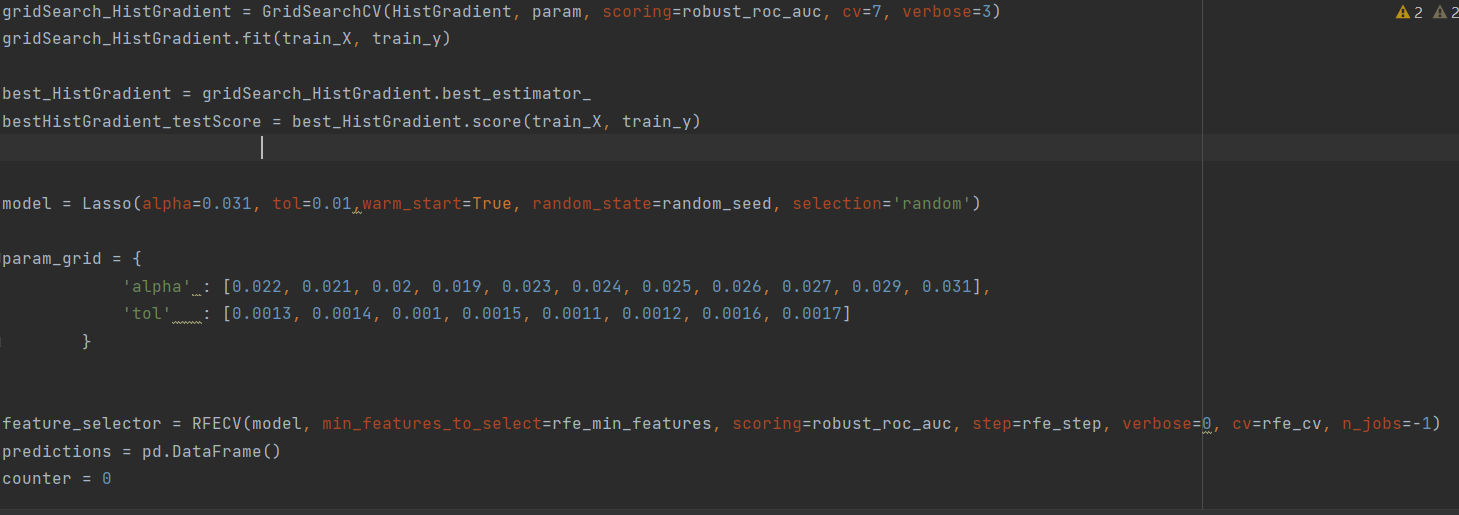
**Here are some reference links:**

[**https://blog.csdn.net/weixin\_44225602/article/details/112914311**](https://blog.csdn.net/weixin_44225602/article/details/112914311)

[**https://www.codeleading.com/article/48412214277/**](https://www.codeleading.com/article/48412214277/)

Here is the final implementation.

But the process is actually quite tortuous.

****

At first,to get the parameter for lasso, I used a stupid but productive approach just like force attack: try almost every possible solution.

Here is a sample:

# features selected by RFECV with lasso

rfe\_min\_features = 12

rfe\_step = 16

rfe\_cv = 21

sss\_n\_splits = 21

sss\_test\_size = 0.4

grid\_search\_cv = 21

noise\_std = 0.01

r2\_threshold = 0.2

random\_seed = 210

param\_grid = {

'alpha' : [0.022, 0.021, 0.02, 0.019, 0.023, 0.024, 0.025, 0.026, 0.027, 0.029, 0.031],

'tol' : [0.0013, 0.0014, 0.001, 0.0015, 0.0011, 0.0012, 0.0016, 0.0017]

}

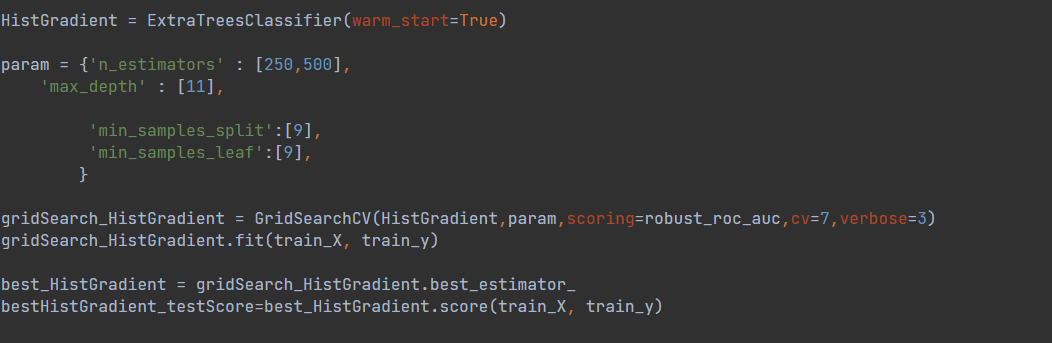
Also I used it combined with logic regression to find the best feachers.

This is the best feature I got from lasso, but it is based on a test and stupited continues trying.

features = ['16', '33', '43', '45', '52', '63', '65', '73', '90', '91', '117', '133', '134', '149', '189', '199', '217', '237', '258', '295']

Please use “logistic\_regression” to see the process.

1. **Select the feachers with lasso**

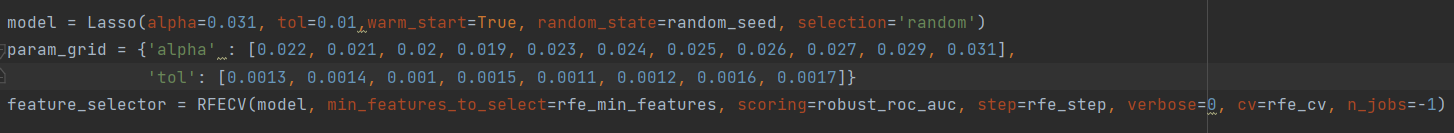
****

First, use the hist gradient to get the best feachers.

It is need to calculate the score for gridSearch\_HistGradient() and feature\_selector()

(without any normalisation )

Then comes to lasso:



The feacher\_selector will get the fit feachers

1. **Finally use the model to do the training.**

Split the data.

Get the needed features.

Remove the no needed features

Grid is used for lassoThen score the result

Reference link: https://www.kaggle.com/code/nayrouzhamdy/trial-9

1. **Here is he final anser**

