**DataMining Report**

team member: MF1732055, Shengda,Huang(Solo)

**1.Introduction**

In this report, I used the credit card dataset to do some fraud detection(<https://www.kaggle.com/dalpozz/creditcardfraud>). The datasets contains transactions made by credit cards in September 2013 by enropean cardholders. This dataset presents transactions that occurred in two days, where we have 492 frauds out of 284,807 transactions. The dataset is highly unbalanced, the positive class (frauds) account for 0.172% of all transactions. And all the variables except ‘Time’ And ‘Amount’ has been transformed with PCA.

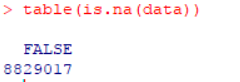
In this project, I used the randomForest method to do the detection, with the SMOTE method to resample the highly imbalanced dataset.

**2.Data Preprocessing**

First we load data from dataset file

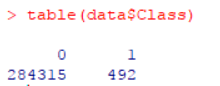


We check whether the data contains missing values.



So there is no missing value in data

However, the dataset is highly imbalanced as we can see from below



If we don’t deal with the imbalance problem, we can still achieve a very high accuracy in classification even if we just simply predict every car. Obviously, it’s not suitable, since the minority class play an importance role in reality and our major task is to find out such anomaly. Since most classification algorithm can handle the class imbalance problem well, we need to take some method to deal with the problem.

Generally, there are five ways to redistribute the data and roughly they can be classified into three class: over-sampling, under-sampling, combination of over-sampling and under-sampling. More details can be found in this article: https://www.analyticsvidhya.com/blog/2017/03/imbalance

d-classification-problem/. Here, I use the SMOTE method to redistribute the data( code show as below)

library(DMwR)

data$Class=as.factor(data$Class)

split=sample(1:nrow(data),nrow(data)\*0.7)

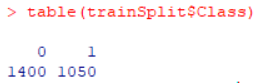
trainset=data[split,]

testset=data[-split,]

trainSplit=SMOTE(Class~., trainset, perc.over=200, perc.under=200)

At first we divide the dataset into training set and test set with the percentage of 0.7 : 0.3. Next we use the SMOTE method to sampling the training set

The redistributed data is showed as below, where two thirds of the 1 class (minority class in origin data) data are synthetic



We will used this sampled data to train our random forest model

**3.MODEL, PREDICTION AND EVALUATION**

1. Model

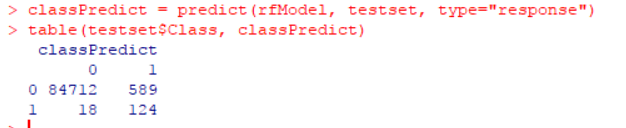
We use the randomForest library, model code showed as below



1. Prediction

We do two type of prediction, one predicted value, one class probabilities.

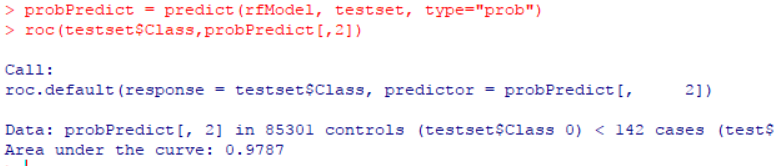
Classification result is showed as below:



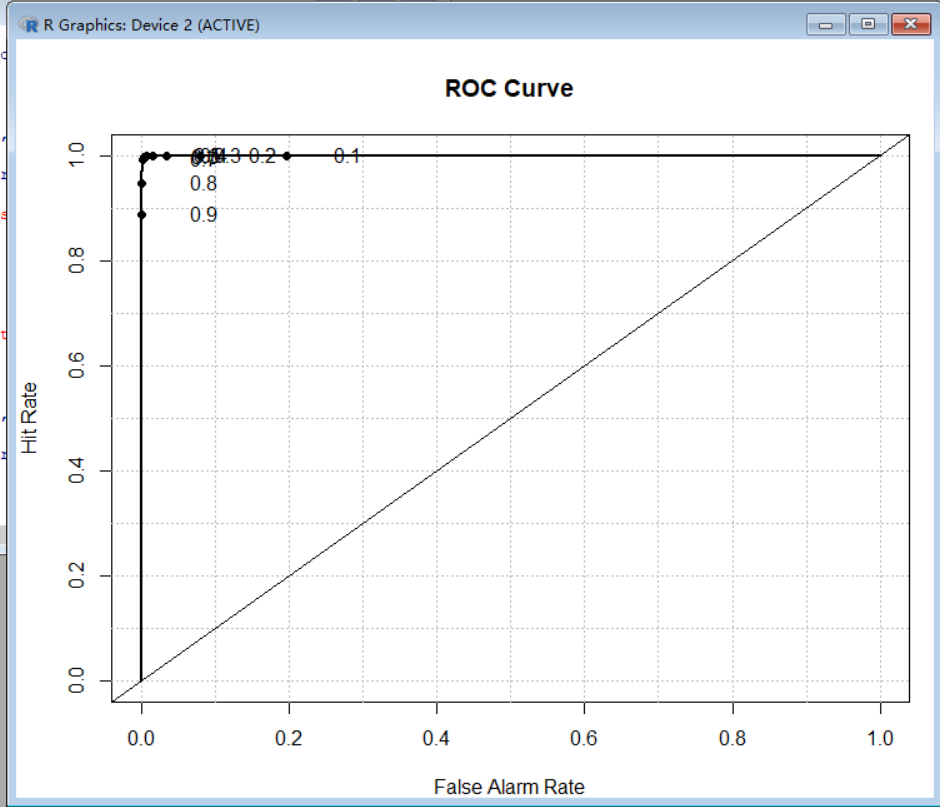
[Specificity](https://en.wikipedia.org/wiki/Specificity_(tests))=124/(589+124)=0.1739

[Negative predictive value](https://en.wikipedia.org/wiki/Negative_predictive_value)=124(124+18)=0.8732

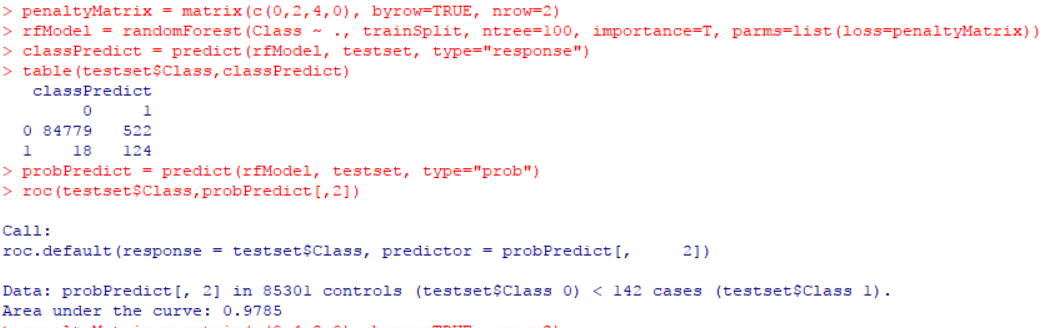
Class probabilities result is showed as below:



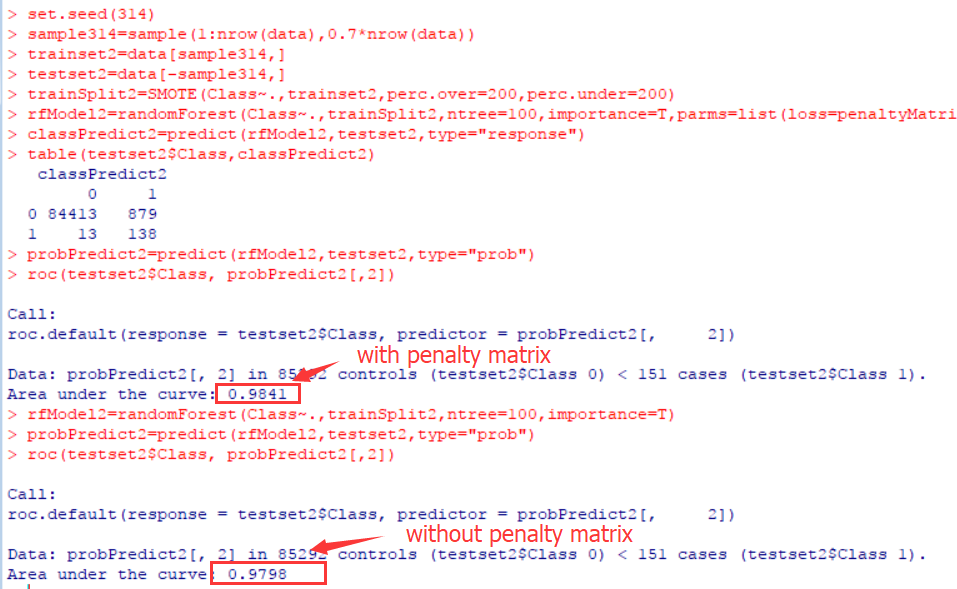
As we can see, we get AUC:0.9787( not bad ). And the roc curve:



1. We add a penalty matrix, the result showed as below:



It’s seen that penalty matrix does not work well. However, after trying several other sample trainset and dataset(with the same split percentage of 0.7 : 0.3), it does gain a bit improvement in AUC, e.g.



But it does increases a little bit false negative rate…

**4.VARIABLE IMPORTANCE**

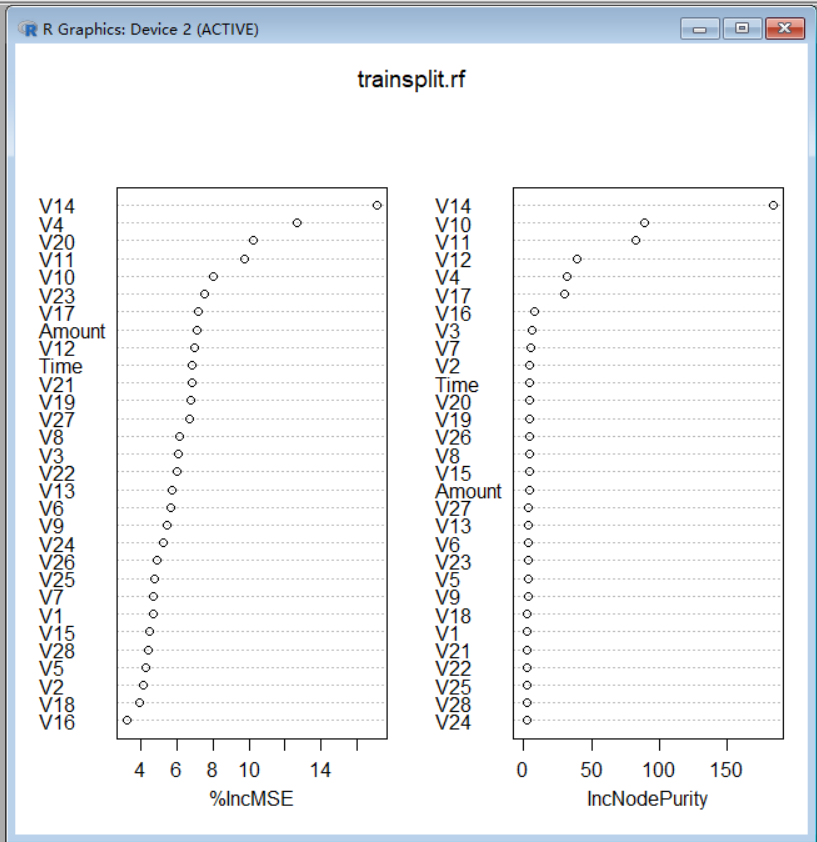
The randomForest method can also measure feature importance (with the importance parameter set to be TRUE. The variable importance is calculated using the out-of-bag-error. Here is an description of how it is calculated in the R package randomForest(quoted from Wikipedia(<https://en.wikipedia.org/wiki/Random_forest#Variable_importance)>):

“The first step in measuring the variable importance in a dataset  is to fit a random forest to the data. During the fitting process the [out-of-bag error](https://en.wikipedia.org/wiki/Out-of-bag_error) for each data point is recorded and averaged over the forest (errors on an independent test set can be substituted if bagging is not used during training).

To measure the importance of the j-th feature after training, the values of the j-th feature are permuted among the training data and the out-of-bag error is again computed on this perturbed data set. The importance score for the j-th feature is computed by averaging the difference in out-of-bag error before and after the permutation over all trees. The score is normalized by the standard deviation of these differences.”

It can be used as a means of variable selection. However, according to the paper “[Bias in random forest variable importance measures: Illustrations, sources and a solution](https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1796903/)”, the variable importance in randomForest is misleading due to biased variable selection in individual classification tree and the improper boostrap sampling with replacement. And according to randomForest page in wiki, the method has some drawbacks, which is biased in favor of those features with more levels.

Here I visualize the variable importance in my experiment:



As we can see, the V14 is a good predictor, as it rank top in both IncMSE(mean square error of OOB-predictions and IncNodePurity. Top in IncMSE means permuting the attribute can bring the worst predictions. Top in IncNodePurity means the attribute bring the most node purity when splitting node.

**5.** **COMPARISON**

Applying SMOTE method do actually improve the AUC of classification on a highly imbalanced dataset. Just take <https://www.kaggle.com/gpreda/credit-card-fraud-detection-with-rf-auc-0-93> for example, it achieve only auc of 0.93.

There are several other interesting and efficient way to do this job, such as using netural network, XGBOOST and so on. But due to the homework pressure from other courses ... , I haven't take a try.