UTD 2016 Spring

CS6301.5U1 Advanced Computational Methods for Data Science

# Assignment 9 – Classification of Credit Approval

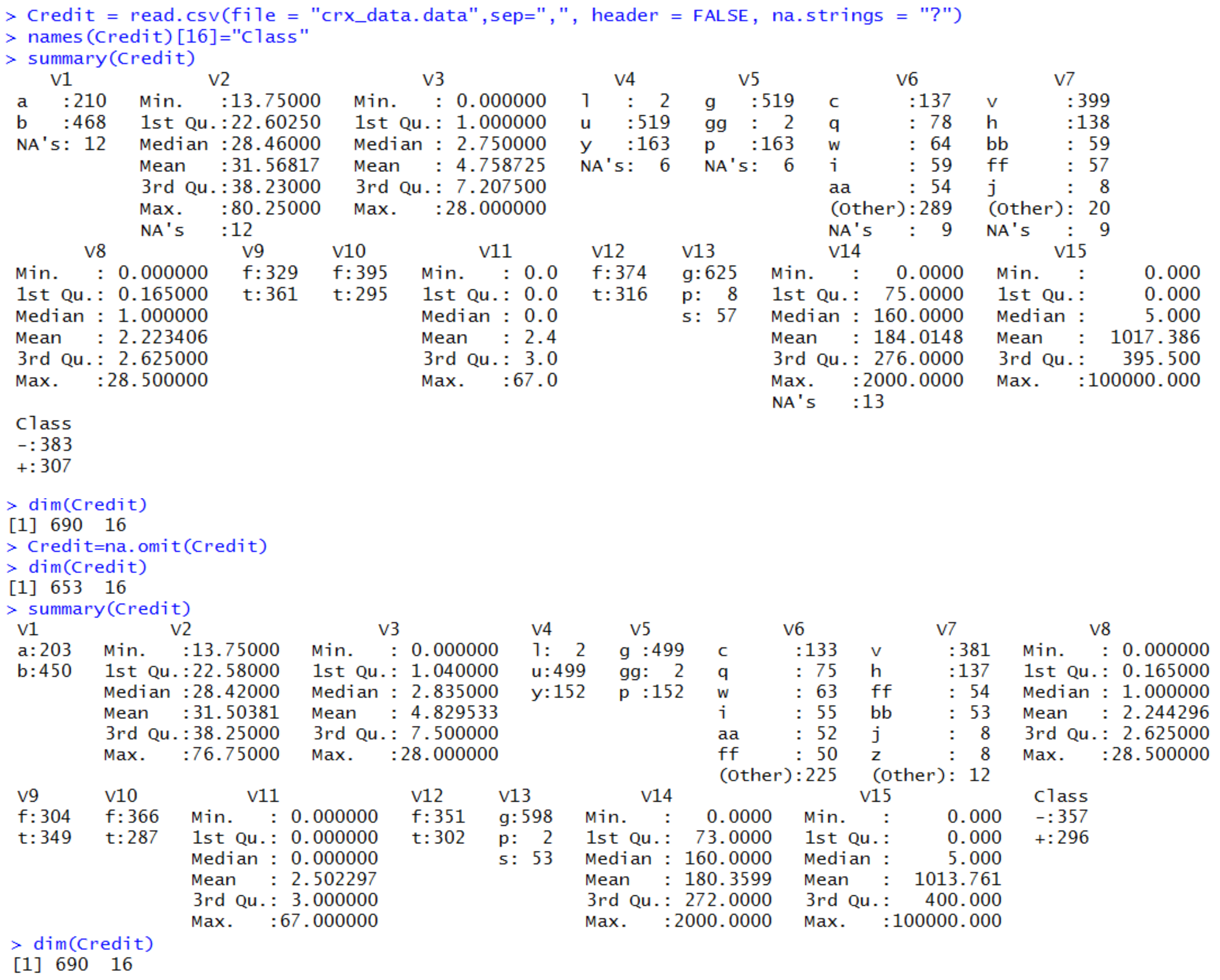
(Using Logistic Regression, Random Forest, Neural Network and SVM)

Name: Yu Zhang, Mingyue Sun

NetID: yxz141631, mxs151730

## 1. Data Information and Preprocessing

The data we are using is about credit card information. All the attribute name and values are replaced by symbols. But we do know that the 16th column is the classification value. Therefore, we renamed the column as “Class”. Also, there are some “?” in the data, probably are the missing values. So we defined them as NAs. After importing the data, we found that  the dimension is 690*16. Then we tried to remove the NAs. The dimension reduced to 653*16. Only 37 records were removed. So, the removal will not influence the outcome so much. We also noticed that some categories in some columns has only few records. This may negatively influence the fitting result. But we don’t know the real meaning of them, in this case, we gave up regrouping those values to others.

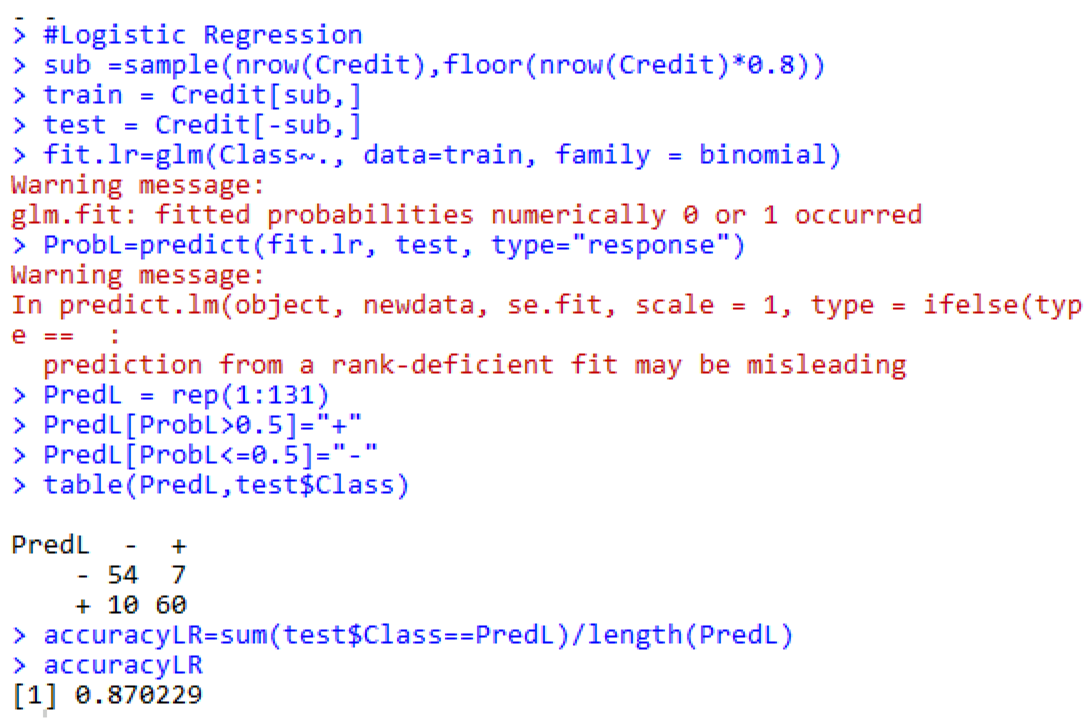


```
> Credit = read.csv(file = "crx_data.data",sep=",", header = FALSE, na.strings = "?")
> names(Credit)[16]="Class"
> summary(Credit)
    V1              V2                  V3                 V4         V5            V6              V7
 a    :210   Min.    :13.75000   Min.    : 0.000000   l   :  2   g   :519   c       :137   v       :399
 b    :468   1st Qu.:22.60250   1st Qu.: 1.000000   u   :519   gg  :  2   q       : 78   h       :138
 NA's: 12   Median :28.46000   Median : 2.750000   y   :163   p   :163   w       : 64   bb      : 59
             Mean    :31.56817   Mean    : 4.758725   NA's:  6   NA's:  6   i       : 59   ff      : 57
             3rd Qu.:38.23000   3rd Qu.: 7.207500                          aa      : 54   j       :  8
             Max.    :80.25000   Max.    :28.000000                         (Other):289   (Other): 20
             NA's    :12                                                    NA's    :  9   NA's    :  9
       V8               V9      V10          V11         V12     V13          V14               V15
 Min.    : 0.000000   f:329   f:395   Min.    : 0.0   f:374   g:625   Min.    :    0.0000   Min.    :      0.000
 1st Qu.: 0.165000   t:361   t:295   1st Qu.: 0.0   t:316   p:  8   1st Qu.:   75.0000   1st Qu.:      0.000
 Median : 1.000000                   Median : 0.0           s: 57   Median :  160.0000   Median :      5.000
 Mean    : 2.223406                   Mean    : 2.4                   Mean    :  184.0148   Mean    :   1017.386
 3rd Qu.: 2.625000                   3rd Qu.: 3.0                   3rd Qu.:  276.0000   3rd Qu.:    395.500
 Max.    :28.500000                   Max.    :67.0                   Max.    :2000.0000   Max.    :100000.000
                                                                      NA's    :13
 Class
 -:383
 +:307

> dim(Credit)
[1] 690  16
> Credit=na.omit(Credit)
> dim(Credit)
[1] 653  16
> summary(Credit)
 V1          V2                  V3                 V4        V5           V6             V7              V8
 a:203   Min.    :13.75000   Min.    : 0.000000   l:  2   g :499   c       :133   v       :381   Min.    : 0.000000
 b:450   1st Qu.:22.58000   1st Qu.: 1.040000   u:499   gg:  2   q       : 75   h       :137   1st Qu.: 0.165000
         Median :28.42000   Median : 2.835000   y:152   p :152   w       : 63   ff      : 54   Median : 1.000000
         Mean    :31.50381   Mean    : 4.829533                    i       : 55   bb      : 53   Mean    : 2.244296
         3rd Qu.:38.25000   3rd Qu.: 7.500000                    aa      : 52   j       :  8   3rd Qu.: 2.625000
         Max.    :76.75000   Max.    :28.000000                    ff      : 50   z       :  8   Max.    :28.500000
                                                                   (Other):225   (Other): 12
 V9      V10           V11              V12      V13          V14               V15              Class
 f:304   f:366   Min.    : 0.000000   f:351   g:598   Min.    :    0.0000   Min.    :      0.000   -:357
 t:349   t:287   1st Qu.: 0.000000   t:302   p:  2   1st Qu.:   73.0000   1st Qu.:      0.000   +:296
                 Median : 0.000000           s: 53   Median :  160.0000   Median :      5.000
                 Mean    : 2.502297                   Mean    :  180.3599   Mean    :   1013.761
                 3rd Qu.: 3.000000                   3rd Qu.:  272.0000   3rd Qu.:    400.000
                 Max.    :67.000000                   Max.    :2000.0000   Max.    :100000.000
 > dim(Credit)
 [1] 690  16
```

## 2. Modeling with Logistic Regression

In this part, we will use logistic regression model to complete the classification. Two steps are included in this section: 1) Building the LR model with the training set of input data and predit the testing set  with this model to see the accuracy of it on the testing. 2) Using 10-folds cross validation, to estimate the performance of the model. The performance is evaluated in accuracy, i.e. number of correctly classified items divided by number of all items.

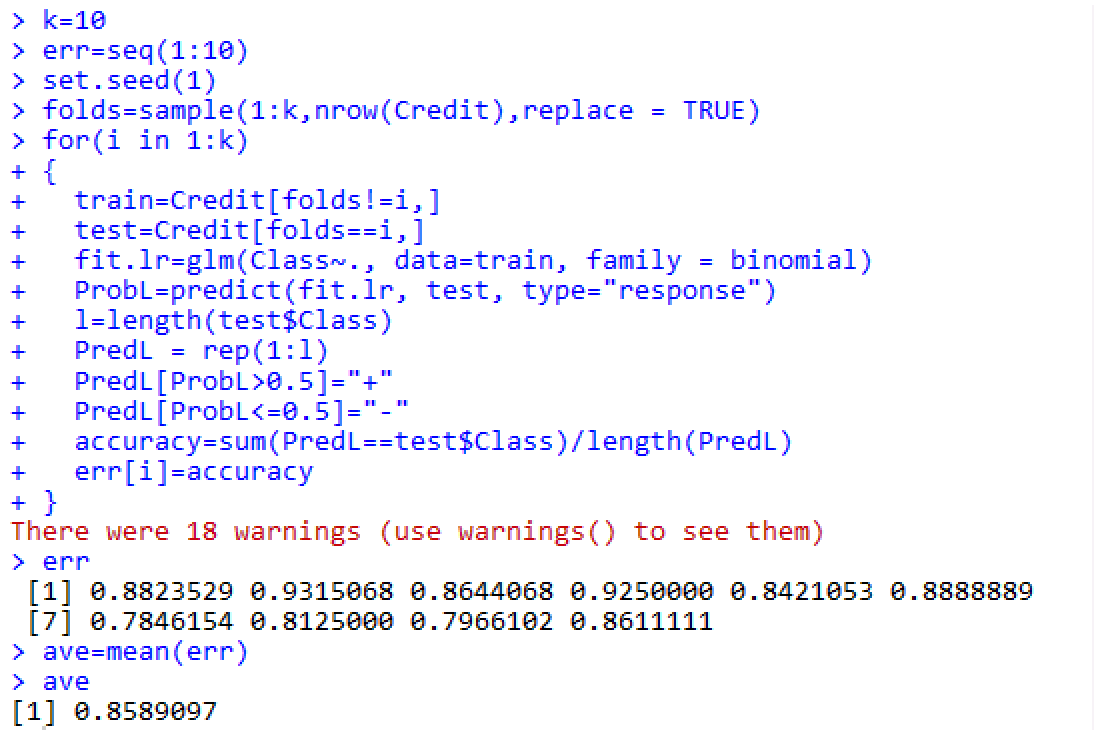


```
> #Logistic Regression
> sub =sample(nrow(Credit),floor(nrow(Credit)*0.8))
> train = Credit[sub,]
> test = Credit[-sub,]
> fit.lr=glm(Class~., data=train, family = binomial)
Warning message:
glm.fit: fitted probabilities numerically 0 or 1 occurred
> ProbL=predict(fit.lr, test, type="response")
Warning message:
In predict.lm(object, newdata, se.fit, scale = 1, type = ifelse(typ
e ==  :
  prediction from a rank-deficient fit may be misleading
> PredL = rep(1:131)
> PredL[ProbL>0.5]="+"
> PredL[ProbL<=0.5]="-"
> table(PredL,test$Class)

PredL  -  +
    - 54  7
    + 10 60
> accuracyLR=sum(test$Class==PredL)/length(PredL)
> accuracyLR
[1] 0.870229
```

As we can see from above, we used about 80% of the original dataset as training set (522 samples) and 131 samples as testing set. This model could give out an accuracy of 0.870229 which is acceptable to us at this moment.

Then we used 10-folds cross validation, to further estimate the performance of the model and the result is shown below:



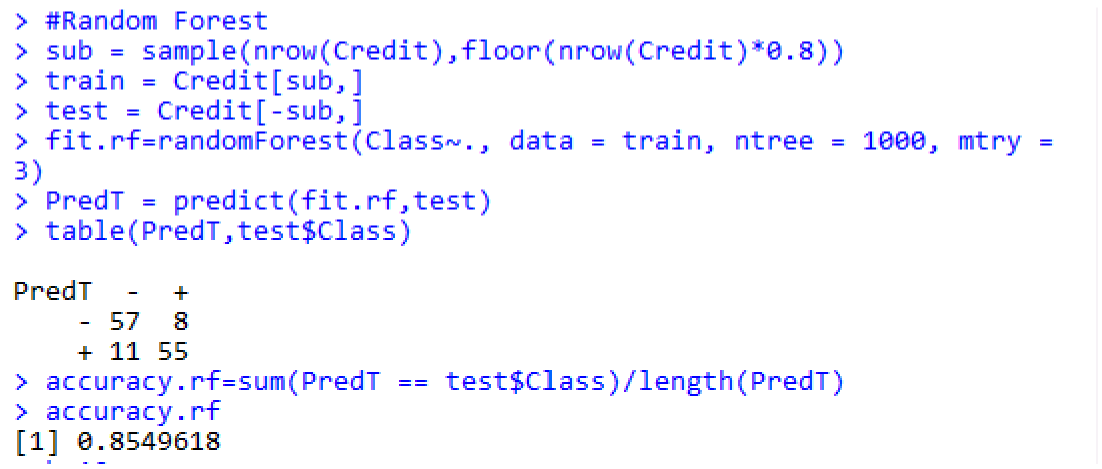
```
> k=10
> err=seq(1:10)
> set.seed(1)
> folds=sample(1:k,nrow(Credit),replace = TRUE)
> for(i in 1:k)
+ {
+   train=Credit[folds!=i,]
+   test=Credit[folds==i,]
+   fit.lr=glm(Class~., data=train, family = binomial)
+   ProbL=predict(fit.lr, test, type="response")
+   l=length(test$Class)
+   PredL = rep(1:l)
+   PredL[ProbL>0.5]="+"
+   PredL[ProbL<=0.5]="-"
+   accuracy=sum(PredL==test$Class)/length(PredL)
+   err[i]=accuracy
+ }
There were 18 warnings (use warnings() to see them)
> err
 [1] 0.8823529 0.9315068 0.8644068 0.9250000 0.8421053 0.8888889
 [7] 0.7846154 0.8125000 0.7966102 0.8611111
> ave=mean(err)
> ave
[1] 0.8589097
```

The accuracy of cross validation using logistic regression model is 0.8589097. From the individual results we can see that the lowest accuracy is 0.7846154, the highest is 0.9315068. The results may vary according to the resampling methods. And we can say that although after using CV, the accuracy for this model drops a little bit, but it is still a stable model which gives an acceptable accuracy.

## 3. Modeling with Random Forest

In this part we used Random Forest model function build random forest model for this dataset. And use the 10-folds cross validation to estimate the overall accuracy of this model in this case to see the performance of this model.

First of all, we still picked up 80% (522 samples) of the original dataset as training and the rest 131 as testing to build the model with training and predict the testing to see the accuracy:

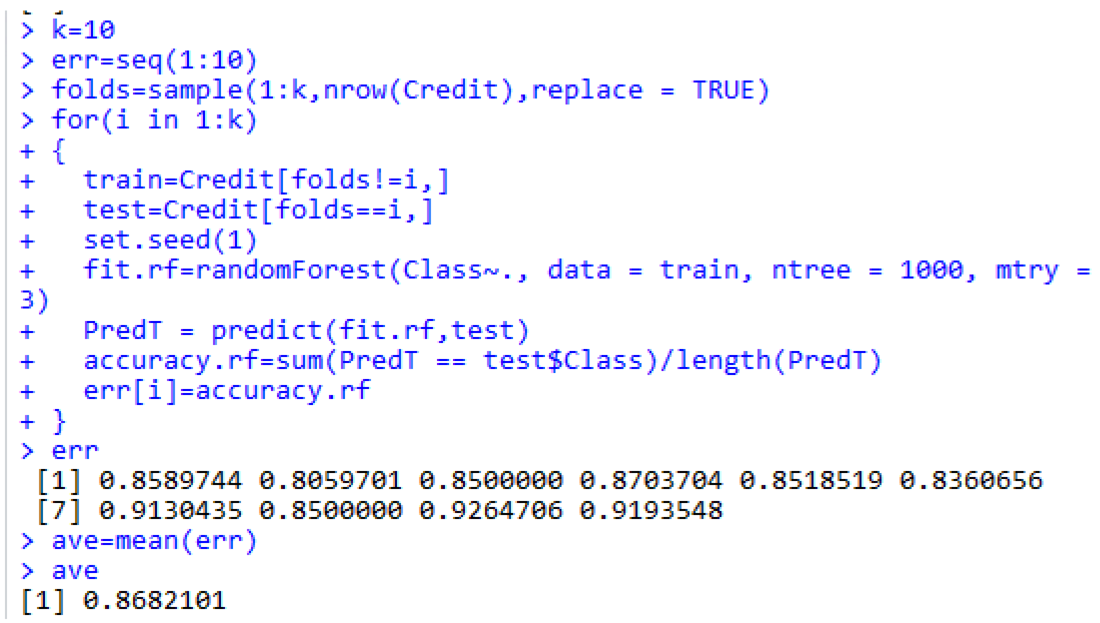


```
> #Random Forest
> sub = sample(nrow(Credit),floor(nrow(Credit)*0.8))
> train = Credit[sub,]
> test = Credit[-sub,]
> fit.rf=randomForest(Class~., data = train, ntree = 1000, mtry = 
3)
> PredT = predict(fit.rf,test)
> table(PredT,test$Class)

PredT  -  +
    - 57  8
    + 11 55
> accuracy.rf=sum(PredT == test$Class)/length(PredT)
> accuracy.rf
[1] 0.8549618
```

As we can see here, this random forest model gives out a model with 0.8549618 accuracy which is slightly lower than the LR model however it is a good prediction to our knowledge.

Then 10-folds cross validation was again used, to further estimate the performance of the model and the result is shown below:



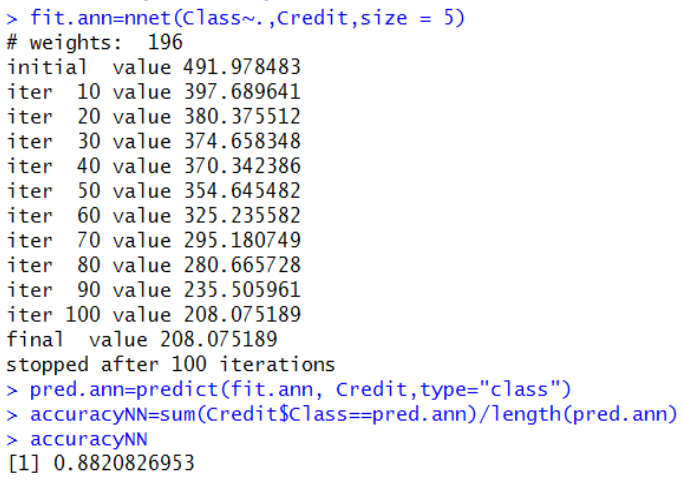
```
> k=10
> err=seq(1:10)
> folds=sample(1:k,nrow(Credit),replace = TRUE)
> for(i in 1:k)
+ {
+   train=Credit[folds!=i,]
+   test=Credit[folds==i,]
+   set.seed(1)
+   fit.rf=randomForest(Class~., data = train, ntree = 1000, mtry = 
3)
+   PredT = predict(fit.rf,test)
+   accuracy.rf=sum(PredT == test$Class)/length(PredT)
+   err[i]=accuracy.rf
+ }
> err
 [1] 0.8589744 0.8059701 0.8500000 0.8703704 0.8518519 0.8360656
 [7] 0.9130435 0.8500000 0.9264706 0.9193548
> ave=mean(err)
> ave
[1] 0.8682101
```

The accuracy of cross validation using random forest model is 0.8682101. From the individual results we can see that the lowest accuracy is 0.8059701, the highest is 0.9264706. The results may vary according to the resampling methods. And to our delight, we notice an increase of accuracy after using CV with this model, so it means CV helps improve accuracy of random forest model.

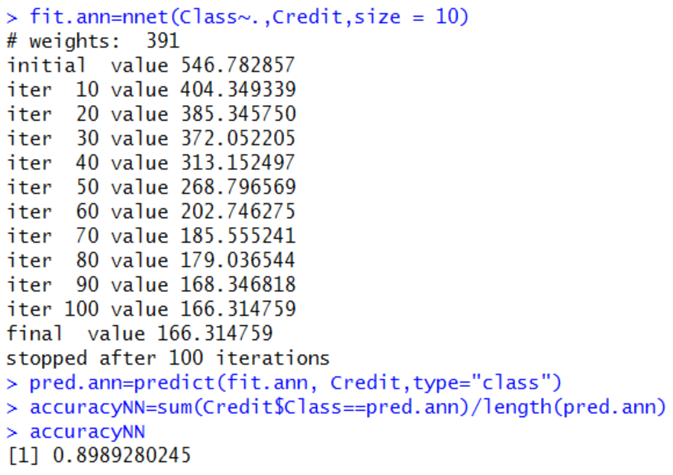
## 4. Modeling with Neural Network

In this part, we will use neural network model to complete the classification. Because it's a classification task, we chose to use "nnet" package in R. This package is used to build neural network model with only one hidden layer. Two steps are included in this section: 1) Training the data with different number of nodes in hidden layer (5,10,15) to compare the training errors. 2) Using 10-folds cross validation, to estimate the performance of the model. The performance is evaluated in accuracy, i.e. number of correctly classified items divided by number of all items.

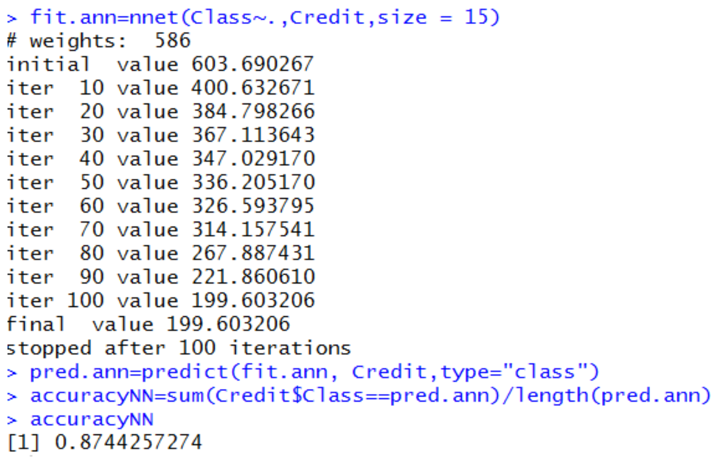
### 4.1 Testing Training Errors with Different Hidden Layers



```
> fit.ann=nnet(Class~.,Credit,size = 5)
# weights:  196
initial  value 491.978483
iter  10 value 397.689641
iter  20 value 380.375512
iter  30 value 374.658348
iter  40 value 370.342386
iter  50 value 354.645482
iter  60 value 325.235582
iter  70 value 295.180749
iter  80 value 280.665728
iter  90 value 235.505961
iter 100 value 208.075189
final  value 208.075189
stopped after 100 iterations
> pred.ann=predict(fit.ann, Credit,type="class")
> accuracyNN=sum(Credit$Class==pred.ann)/length(pred.ann)
> accuracyNN
[1] 0.8820826953
```



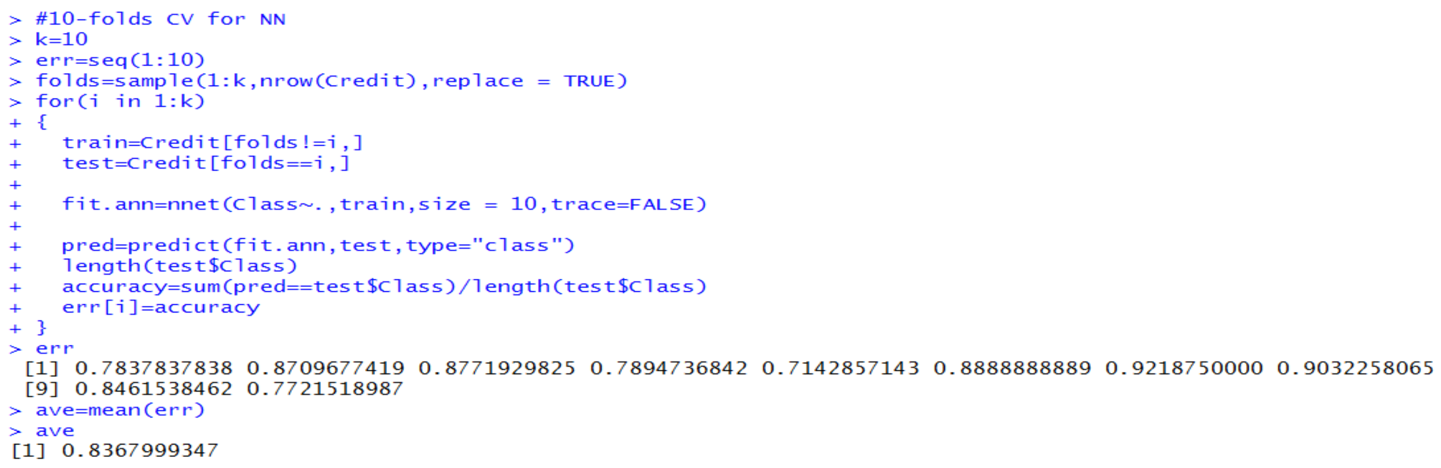
```
> fit.ann=nnet(Class~.,Credit,size = 10)
# weights:  391
initial  value 546.782857
iter  10 value 404.349339
iter  20 value 385.345750
iter  30 value 372.052205
iter  40 value 313.152497
iter  50 value 268.796569
iter  60 value 202.746275
iter  70 value 185.555241
iter  80 value 179.036544
iter  90 value 168.346818
iter 100 value 166.314759
final  value 166.314759
stopped after 100 iterations
> pred.ann=predict(fit.ann, Credit,type="class")
> accuracyNN=sum(Credit$Class==pred.ann)/length(pred.ann)
> accuracyNN
[1] 0.8989280245
```



```
> fit.ann=nnet(Class~.,Credit,size = 15)
# weights:  586
initial  value 603.690267
iter  10 value 400.632671
iter  20 value 384.798266
iter  30 value 367.113643
iter  40 value 347.029170
iter  50 value 336.205170
iter  60 value 326.593795
iter  70 value 314.157541
iter  80 value 267.887431
iter  90 value 221.860610
iter 100 value 199.603206
final  value 199.603206
stopped after 100 iterations
> pred.ann=predict(fit.ann, Credit,type="class")
> accuracyNN=sum(Credit$Class==pred.ann)/length(pred.ann)
> accuracyNN
[1] 0.8744257274
```

From the results above we can see the training errors when hidden nodes sizes are 5, 10, 15 are 0.8820826953, 0.8989280245, 0.8744257274. The difference is small, but hidden nodes do have infect to the predict results. Too many or very few of nodes will negatively influence the result. So, in the following section, we will use 10 as the number of hidden nodes.

### 4.2 Cross Validation for Model Evaluation of Neural Network



```
> #10-folds CV for NN
> k=10
> err=seq(1:10)
> folds=sample(1:k,nrow(Credit),replace = TRUE)
> for(i in 1:k)
+ {
+   train=Credit[folds!=i,]
+   test=Credit[folds==i,]
+ 
+   fit.ann=nnet(Class~.,train,size = 10,trace=FALSE)
+ 
+   pred=predict(fit.ann,test,type="class")
+   length(test$Class)
+   accuracy=sum(pred==test$Class)/length(test$Class)
+   err[i]=accuracy
+ }
> err
 [1] 0.7837837838 0.8709677419 0.8771929825 0.7894736842 0.7142857143 0.8888888889 0.9218750000 0.9032258065
 [9] 0.8461538462 0.7721518987
> ave=mean(err)
> ave
[1] 0.8367999347
```

The accuracy of cross validation using neural net model is 0.8367999347. From the individual results we can see that the lowest accuracy is 0.7142857143, the highest is 0.921875. The results may vary according to the resampling methods and the inner computation trace of the “nnet” function. The results in terms of accuracy, however, is not bad.

## 5. Modeling with Support Vector Machine

In this part we used SVM model function “svm” provided by “e1071” package in R to build SVM model for this dataset. And use the 10-folds cross validation to estimate the overall accuracy of this model in this case to see the performance of SVM model.

```
> #10-folds CV for SVM
> k=10
> err=seq(1:10)
> folds=sample(1:k,nrow(Credit),replace = TRUE)
> for(i in 1:k)
+ {
+   train=Credit[folds!=i,]
+   test=Credit[folds==i,]
+
+   fit.svm=svm(Class~.,data=train)
+
+   pred=predict(fit.svm,test)
+   length(test$Class)
+   accuracy=sum(pred==test$Class)/length(test$Class)
+   err[i]=accuracy
+ }
> err
 [1] 0.8965517241 0.8245614035 0.8730158730 0.8970588235 0.8571428571 0.8837209302 0.8888888889 0.8709677419
 [9] 0.8591549296 0.7887323944
> ave=mean(err)
> ave
[1] 0.8639795566
```

From the result above we can see that the overall accuracy using SVM is 0.8639795566, which is good for classification in this small dataset. For each individual cases, the lowest accuracy is 0.7887323944, the highest one is 0.8970588325. The remaining results are around 0.85. This indicates that the performance of SVM is relatively stable. Like other cases in cross validation, results may vary in different cases because of the distribution of samples in each folds. We tested several times, all the results are around 0.85. Thus, SVM is a powerful classification model in this case.

## 6. Conclusion

The performances in 10-folds cross validation of those four different models are shown in the table below. The Logistic Regression model gives an average accuracy of 0.8589097 which although is slightly worse than the accuracy of original LR model that we built, we could still consider this model to be a stable one. The performance of Random Forest is 0.8682101 and to our delight, we did notice an increase of accuracy when we applied CV to this dataset than the original model which was built without CV. In terms of Neural Network, the classification accuracy is correlated with the number of hidden nodes (probably also correlated with iteration/propagation times and the number of hidden layers). The result is not bad, which in this case, is 0.8367999347. But to get better performance, many parameters may need to be tuned. SVM is quite good in this experiment. The overall accuracy reached 0.8639795566, and each individual cases in 10 folds are very stable, most of which are around 0.85. It’s a good classification model in this case.

| | Logistic Regression | Random Forest | Neural Network | SVM |
|---|---|---|---|---|
| CV Accuracy | 0.8589097 | 0.8682101 | 0.8367999347 | 0.8639795566 |