R의 분류방법을 이용한 신용카드 승인 분석 비교

송종우*†

* 이화여자 대학교 통계학과

A Comparison of Classification Methods for Credit Card Approval Using R

Jongwoo Song*†

* Dept. of Statistics, Ewha Womans University

Key Words : R, Credit, Classification, Supervised Learning, Cross-Validation

Abstract

The policy for credit card approval/disapproval is based on the applier's personal and financial information. In this paper, we will analyze 2 credit card approval data with several classification methods. We identify which variables are important factors to decide the approval of credit card. Our main tool is an open-source statistical programming environment R which is freely available from http://www.r-project.org. It is getting popular recently because of its flexibility and a lot of packages(libraries) made by R-users in the world. We will use most widely used methods, LDA/QDA, Logistic Regression, CART (Classification and Regression Trees), neural network, and SVM (Support Vector Machines) for comparisons.

1. Introduction

When a person apply for a new credit card, a bank or a credit card company will use the person's personal and financial information to decide if he/she can have a credit card. In general, age, education level, income, expenditures, credit history and derogatory reports are considered important. Classification methods can be used to classify which groups of people get approved and which groups of people get rejected. Moreover, the result of classification will give us an insight that which variables are important factors and overall understanding of credit card approval policy.

There are many literatures that compared the performance of several classification methods. Dudoit et al(2002) compared several classification methods for gene expression data, Bauer and Kohavi (1999) compared classification methods based on voting algorithms, Hand and Henley(1997) compared statistical classification methods for consumer credit scoring. In general, there is no one method can perform better than all others in all kinds of data. In this paper, we try to find a method that can predict the credit card approval with low misclassification rate. In chapter 2, we introduce several widely used classification methods, including LDA/QDA, Logistic Regression, CART, neural network and SVM with sample R codes. In chapter 3, we analyze 2 credit card approval data with the methods introduced in chapter 2. In chapter 4, we compare the performance of classification methods. In chapter 5, we give concluding remarks and introduce other classification methods.

† 교신저자 josong@ewha.ac.kr

2. Classification Methods

In this chapter, we will use the following notation. The group variable G has M classes and X is the feature vector.

2.1 LDA(Linear Discriminant Analysis), QDA(Quadratic Discriminant Analysis)

Suppose the class conditional density of X given $G=k$ is $f_k(x)=f(x|G=k)$ and the prior probability of class k is π_k.

Then by the Bayes rule, the posterior probability of $G=k$ given $X=x$ is

$$P(G=k|X=x)=\frac{f_k(x)\pi_k}{\sum_{l=1}^{M} f_l(x)\pi_l}.$$

For LDA, we assume that the class conditional density follows the multivariate normal distribution with a common variance, $f_k(x)=MVN_p(\mu_k,\Sigma)$. For QDA, we do not assume the common variance. Therefore, each group can have different variance matrix Σ_k and the conditional density is$f_k(x)=MVN_p(\mu_k,\Sigma_k)$. (Hastie, T. et al. 2001). The decision boundary defined by LDA is linear and by QDA is quadratic. The function for LDA in R is lda() and QDA is qda() and we need a package **MASS**. The syntax for the lda() or qda() is

```
lda.res ← lda(Y~X1+X2+X3, data=mydata,
CV=TRUE)
```

lda.res object have 5 attributes. Among them, class variable is the predicted class and posterior is the posterior probability matrix. qda() function have the same syntax and output.

2.2 Logistic Regression

Logistic Regression assumes that the log posterior odds is linear in X,

$$\log\frac{P(G=k|X=x)}{P(G=M|X=x)}=\beta_{k0}+\beta_k^t x.$$

The parameter estimation can be done using IRLS(Iterative Reweighted Least Squares) method. In general, LDA and Logistic regression give similar results (Hastie, T. et al. 2001). If it is a 2-class problem then we can use glm() function with binomial family and logit link in R. If there are more than 2 classes then we have to use package **VGAM** and vglm() function with multinomial family or multinom() function in package **nnet**(Yee, T. W. and Wild, C. J., 1996).

2-class problem

```
logist.res1 ← glm(Y ~ X1+X2+X3, data=my-
data, family = binomial (link="logit"))
```

3 or more class problem

```
vglm(Y ~ X1+X2+X3, data=mydata, family=
multinomial)

multinom(Y ~ X1+X2+X3, data=mydata)
```

logis.res1 object have many attributes. Among them, fiited.values variable have the posterior probability matrix. If we use glm() function with binomial family then we can apply step() function to the result object of glm() for stepwise procedure. If we have many explanatory variables, then stepwise procedure can help reduce the number of variables.

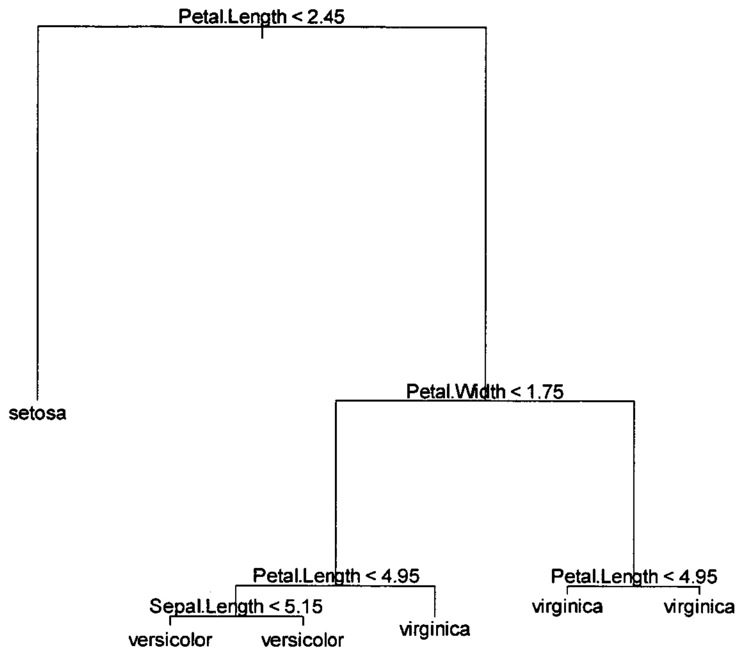
2.3 CART(Classification And Regression Tree)

CART is a very popular datamining tool because of its simplicity(ease of interpretation) and fairly good prediction power. It keeps partitioning data with explanatory variables in rectangle (binary split) which gives the minimal impurity until the terminal node has a predefined minimum size. Then it fits the response variable in each partition. If the response variable is numerical then it is a regression and if categorical then it is a classification. The goal is to try to find a partition so that the response is homogeneous in each partition. There are several issues in tree method. First, the tree size which is the number of final nodes, can be considered the complexity of model. In principle, the more complex model has better prediction power. However, if model is too complex, then it is hard to interpret and there can be a overfitting problem. Therefore, we have to balance between the complexity and goodness of fit. For tree, we can use a cost-complexity criterion to find a optimal size (Brieman et al. 1984). Secondly, tree result can be unstable because of its hierarchical structure.(Hastie, T. et al. 2001) In other words, if very small portion of data changed, then the tree result can be totally different because if the first split is different then the following splits can be all different. Bagging can fix this problem but the results of bagged tree is not a tree any more. In R, there are 2 functions for CART, rprat() and tree(). We need a package **tree** to use CART.

```
> library(tree)
> tree.res ← tree(Species ~., data=iris)
> plot(tree.res)
> text(tree.res)
```

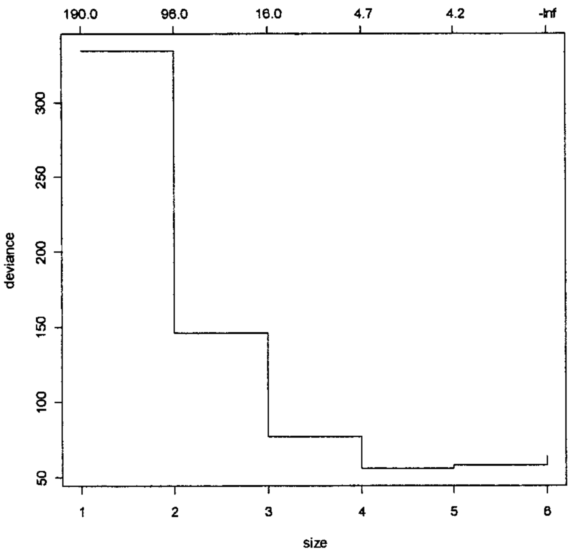
Figure 2.1 is the result of tree() in R for iris data which is built-in data in R. To find the optimal size of tree, we can use the following commands.

```
> trl ← tree(Species ~., data=iris)
> trl.cv ← cv.tree(trl)
> for (i in 2:10){
+ trl.cv$dev ← trl.cv$dev+ cv.tree(trl)$dev
+ }
```



<Figure 2.1> The result of CART for iris data

```
> tr1.cv$dev ← tr1.cv$dev/10
> plot(tr1.cv)
```



<Figure 2.2> Deviance vs tree size

The above code computes the deviance of tree using 10-fold CV (cross-validation). Since the choice of 10 partitions for CV is random, we compute CV 10 times and take an average of 10 deviances. As we can see from the figure 2.2, the minimum deviance is achieved when the size is

4. So we can find the best tree with size 4 with the following commands.

```
>final.tr ← prune.tree(tr1,best=4)
```

2.4 Neural networks

Neural networks assume a model that the response variable(output layer) has a relationship with explanatory variables (input layer) and there is hidden layer between them. It assumes that input layer affects all nodes in hidden layer and the response is affected by all nodes in hidden layer (Anderson 1995). The number of hidden layers can be large but one ore two hidden layers are used most commonly. A neural network model can be used for both regression and classification problem. To use neural network in R, we need a package **nnet** which can be used to fit a single hidden layer model. A package nnet is the part of the base R, so we don't need to install it separately.

```
>library(nnet)
>nn.res ← nnet(Y ~X1+X2+X3, data=mydata,
size=10, decay=0.01, maxit=1000)
```

There are several arguments in nnet() function. The formula is same as in lm(), size is the number of hidden units, decay is the updating weight, and maxit is the maximum number of iterations. The result object have many attributes and fitted.values is the posterior probability matrix.

2.5 SVM(Support Vector Machines)

SVM is one of the most popular classification methods now because of its good performance. Basically, SVM try to find a decision boundary that has the maximum margin. A margin is the distance between decision boundary to the nearest data point. SVM in R is well explained in Karatzoglou, A. and Meyer, D.(2006). There are 4 packages available in R that supports SVM. They are **kernlab, e1071, klaR, svmpath. e1071** uses an award-winning SVM algorithm libsvm. **kernlab** provides a lot of kernel-based methods. **klaR** supports SVMlight and Regularized Discriminant Analysis. **svmpath** finds an optimal cost parameter for SVM. SVM was originally developed to handle 2-class problem. There are 2 popular approaches for multi-class problems. Suppose there are K classes. First, we can consider it is K 2-class problems (one-against-all). Fix one class and consider all the rest classes as the other class. So we will have K classifiers. To predict the class for new observation, compute the decision values from all K classifiers and assign the observation to the class with the highest decision value. Second method is all possible pairwise classification (one-against-one). When we build classifiers, consider all combinations of 2 class problem. Then there are k(k-1)/2 classifiers. To predict the class for new observation, we can find out which class has been chosen mostly (majority rule or max wins). In this paper, we will show the example using **e1071**.

```
> library(e1071)
> svm.res ← svm(Y~X1+X2+X3, data=mydata, type= "C-classification", kernel="radial", cost=10, gamma=0.25)
```

As we can see from the above command, there are several arguments in svm() function. Type argument specifies classification or regression, kernel argument specifies which kernel will be used, gamma is for the kernel function and cost is cost of constraints violation. We can find optimal tuning parameters (gamma and cost) using tune.svm() function. For example, if we like to find best parameter sets for gamma (0.1, 0.2, ..., 1) and cost (0.1, 0.2, ..., 1) then we can use the following command.

```
> tune1 ← tune.svm(Species~., data=iris, gamma=seq(0.1,1,0.1), cost=seq(0.1,1,0.1))
```

For iris data, the best parameters are gamma=0.3, cost=0.2 in terms of minimum cross-validation value.

3. Credit Card Approval Analysis

In this chapter, we analyze two datasets. First data is from (Greene 2007). There are 1319 observations(people) with 13 variables. The response variable is Cardhldr which is 1 if credit card is approved and 0, otherwise. There are 12 explanatory variables as we can see in table 3.1 There are 1023 people got accepted their credit card applications and 296 people got rejected. Ages vary from 0.17 to 83.5. One thing to note is that there are 7 observations with age less than 1 year. Income level varies from $210 to $135,000. There are 581 people have their own home (738 rent) and 1,228 people are not self-employed. We made a all possible pairwise plots to check the relationship among the variables especially relationship with the response variable. However, we could not find any clear pattern even if we used conditional plot with lattice package. We applied 6 classification methods introduced in Chapter 2. For LDA, all the explanatory variables must be continuous, in principle. Therefore, we removed ownrent and selfempl variables from the explanatory variables when we fit LDA.

All but QDA could build a classifier. For QDA, there is a rank deficiency error, so we skip it. Although the fitted models are a little different for different models, we could say that the chance of getting approved becomes higher if a person's Majordrg is smaller, Age, Income, Avgexp higher, non self-employed, and home-owner.

<Table 3.1> Explanatory variables for Greene credit card approval data

Cardhldr	1 if accepted, 0 otherwise
Majordrg	Number of major derogatory reports
Age	Age n years plus twelfths of a year
Income	Yearly income (divided by 10,000)
Exp_Inc	Ratio of monthly credit card expenditure to yearly income
Avgexp	Average monthly credit card expenditure
Ownrent	1 if owns their home, 0 if rent
Selfempl	1 if self employed, 0 if not
Depndt	1+number of dependents
Inc_per	Income divided by number of dependents
Cur_add	months living at current address
Major	number of major credit cards
Active	number of active credit accounts

However, some variables are not significant and one very dominating variable is Avgexp (Average monthly credit card expenditure). It is very obvious from TREE result and the frequency table between Cardhldr and Avgexp. We made a new variable avgexp0 which is 0 if avgexp=0, 1 otherwise. The table is as following.

		avgexp 0	
		0	1
Cardhldr	0	296	0
	1	21	1002

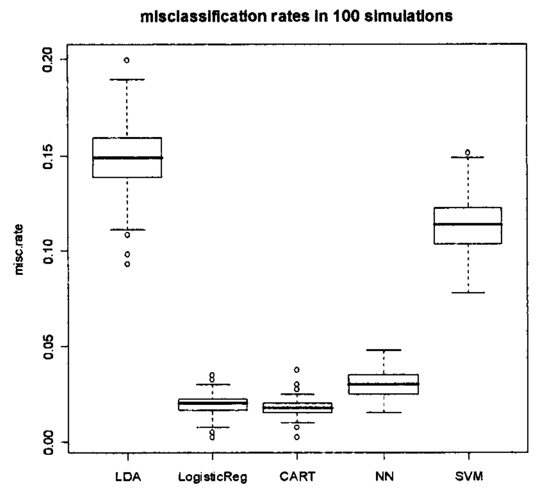
As we can see 296 people with avgexp=0 got rejected their application and only 21 people got accepted. One can wonder why the Majordrg (number of major derogatory reports) is not dominating factor because it seems like very important variable. The reason is that most of high Majordrg people have zero avgexp value. Since they are very highly correlated, we could use avgexp variable only to build a classifier. The result is very interesting. If you do not have a credit card then it is very hard to get a new one. If you already have credit card and keep using it then it is very easy to get another one regardless of other financial status. The second data is from UCI Machine Learning Repository(7). The data has zero/one response variable for approval/disapproval and 15 explanatory variables with 6 continuous variables and 9 categorical variables. However, all the variables are masked to protect confidentiality of the data. Since we cannot interpret the classification results, we will use this data for performance comparison only in Chapter 4.

4. Performance Comparison

In this chapter, we compare the classification methods introduced in chapter 2 with two credit card approval data. We use 70% of data for training (model fitting) and 30% for testing (evaluating). We can use **sample()** function in R to choose training/test sets randomly. We repeat 100 times and compute mean and standard deviation of misclassification rates. One of good features in R is that we can use **predict()** function to compute the fitted value in test data for all 5 methods. It is because R is object-oriented language, so we can apply the same method predict() if the object has the right class. The full code and results for the simulation is available at http://home.ewha.ac.kr/~josong/classification.html.

<Table 4.1> Mean(Std.Dev) misclassification rates for credit approval data1 in 100 simulations

Methods	Misclassification rate
LDA	0.141(0.015)
LogisticRegression	0.019(0.006)
CART	0.017(0.007)
Neural network	0.029(0.006)
SVM	0.109(0.013)



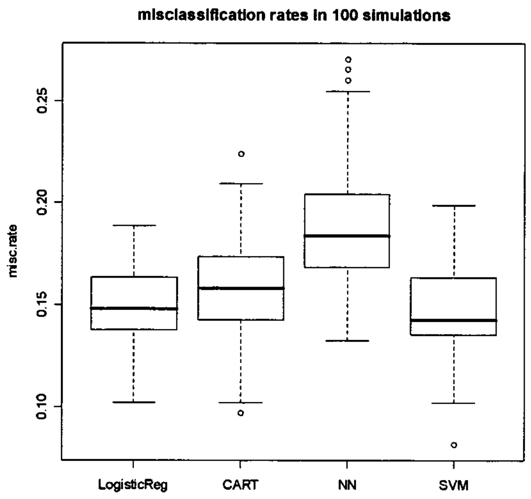
<Figure 4.1> misclassification rates for credit approval data1 in 100 mulations

As we can see from table 4.1 and figure 4.1, CART, Logistic Regression, and neural network perform well and LDA and SVM perform worse.

Especially superior 3 methods' misclassification rates for test set are less than 3%. CART performs best because if there are couple of dominating factors then CART can find those variables very well. For second dataset, we used Logistic Regression, CART, neural network and SVM because LDA/QDA report errors.

<Table 4.2> Mean(Std.Dev) misclassification rates for credit approval data2 (UCI data) in 100 simulations

Methods	Misclassification rate
LogisticRegression	0.143(0.021)
CART	0.157(0.024)
Neural network	0.190(0.027)
SVM	0.145(0.021)



<Figure 4.2> misclassification rates for credit approval data2 (UCI data) in 100 simulations

As we can see from table 4.2 and figure 4.2, all methods perform similarly. The misclassification rates for Logistic regression, CART, SVM are very similar and neural network performs worst. Training time for each method varies significantly. LDA and CART are the fastest methods to train/predict and Logistic regression, SVM and neural network are slower, in order. It took a lot more to train neural network than other methods.

5. Conclusion

We have analyzed credit card approval data with 6 classification methods. We found that previous history of credit card usage is the most important factor for the approval and if you have any credit cards and have used them then it is very likely to get another one. Moreover, if you do not have any credit card then it is very hard to get a new one. It is because if you do not have a credit card then it is likely that you do not have any credit history. In that case, the credit card companies do not want to take a risk. Actually, when you get approved and receive your first credit card then the credit limit is usually very low (less than $1,000). We have introduced most commonly used 6 classification methods available in R with example data and commands. They have very similar syntax and we can predict the classes of new data with predict() function for all methods. We also compared the performance of these methods with two credit card approval datasets. Logistic regression and CART performed very well for these data. LDA/QDA have some problems to handle when there is multicolinearlity problem in X matrix. If the number of observations is smaller than the number of variables (n<p) then either we can't fit the model for LDA/QDA and Logistic regression or the fitting is usually not good. It doesn't happen in CART, Network and SVM. It took the longest time for network to train the data and CART and LDA took least of time to train. Especially, CART's performances were very good for both datasets. Moreover, it is very easy to interpret the result of CART, so we recommend to use CART for credit card approval data. However, there is no one method that can outperform all other methods in classification because the performance depends on

the data structure, characteristics, etc. (Hand and Henley 1997). To find a good classification methods for certain type of data is not an easy task. We have to try several methods and compare their performances to find an optimal method. There are other classification methods available in R and we will introduce some of them. **AMORE** package provides more flexible network model. **classPP** package provides Projection pursuit classifier. **knncat**, **knntree,** and knn() function in **e1071**, provide classification methods based on k-nearest neighbor algorithm. **randomForest** package provide a classification method based on random forest.

References

[1] Anderson, J. A. (1995) "*An Introduction to Networks*", MIT press, Massachusetts.

[2] Breiman, L., Friedman, J. H., Olshen, R. A., and Stone, C. J. (1984). *"Classification and Regression Trees"* Wadsworth and Brooks, Pacific Grove, CA

[3] Dudoit, S., Fridlyand, J., and Speed, T.P. *"Comparison of Discrimination Methods for the Classification of Tumors Using Gene Expression Data"* (2002), UC Berkeley technical report #576

[4] Kohavi, R. and Bauer, E. (1999) "An Empirical Comparison of Voting Classification Algorithms: Bagging, Boosting, and Variants", *Machine Learning*, Vol 36, pp. 105-142

[5] Greene, W. H. (2007) *"Econometric Analysis"*, 6th edition, Prentice Hall, New York.

[6] Hand, D. J. and Henley W.E. (1997) "Statistical Classification Methods in Consumer Credit Scoring: A Review", *Journal of the Royal Statistical Soceity*. Series A, Vol. 160, No. 3, PP.523-541

[7] Hastie, T., Tibshirani, R. and Friedman, J.(2001) *"The Elements of Statistical Learning"*, Springer, New York.

[8] Karatzoglou, A. and Meyer, D.(2006) "Support Vector Machines in R", *Journal of Statistical Software*, Vol 15, Article 9.

[9] Yee, T. W. and Wild, C. J. (1996) "Vector Generalized Additive Models", *Journal of the Royal Statistical Society*, Series B, Vol. 58, No. 3, pp. 481-493

[10] UCI Machine Learning Repository, http://mlearn.ics.uci.edu/MLRepository.html