**Categorical Data Project II Report**

**Georgia State University**

**Author: Yi Lu**

**Date: 4/29/2019**

**Contents**

[Table of Contents](file:///D:\Categorical%20Data%20Project%20I%20Report.docx#_Toc381203982) 2

[Objective](file:///D:\Categorical%20Data%20Project%20I%20Report.docx#_Toc381203984) 3

[Introduction](file:///D:\Categorical%20Data%20Project%20I%20Report.docx#_Toc381203985) 3

[Design](file:///D:\Categorical%20Data%20Project%20I%20Report.docx#_Toc381203985) 3

[Results and Conclusion 6](file:///D:\Categorical%20Data%20Project%20I%20Report.docx#_Toc381203988)

[Reference 12](file:///D:\Categorical%20Data%20Project%20I%20Report.docx#_Toc381203989)

[Appendix 12](file:///D:\Categorical%20Data%20Project%20I%20Report.docx#_Toc381203989)

**Objective**

The objective of this whole project is dealing with the credit approval rate data, the data set is originally from credit card approval rate and has been spited into a training data and a test data. The purpose of this data set, we focus on classification and prediction, and to build a logistic regression model and use it to obtain a classification rule which is used to predict whether a credit card application can be approved.

**Introduction**

For the data set. This dataset is interesting because there is a good mix of attributes -- continuous, nominal with small numbers of values, and nominal with larger numbers of values. There are also a few missing values.

After import in R, make an appropriate plot to explore the data set, determine whether a variable is continuous or categorical, if it is categorical, need to convert it to factor in r. Replacing the missing values. Make appropriate plots to explore the data set. First, without considering regularity, select a logistic regression model and then using the leave-one-out or cross-validation to the training data, calculate the ROC curves and AUC. Determine the optimal cut-off point and the corresponding classification rule. Apply the rule to the test data to calculate the classification accuracy. Then using the training data and the regularity methods such as ridge penalty or lasso penalty to build a logistic regression model. Use appropriate methods to choose tuning parameter and cut to cut-off point Then determine the final classification rule. Apply the classification rule to the test data and calculate the prediction accuracy. Compare it to the originally model without the regularity. Make the final conclusion.

**Design**

The data set used to build the model and classification rule consists of 15 variables. All attribute names and values have been changed to meaningless symbols to protect confidentiality of the data, brief attribute Information shown below:

A1: b, a.

A2: continuous.

A3: continuous.

A4: u, y, l, t.

A5: g, p, gg.

A6: c, d, cc, i, j, k, m, r, q, w, x, e, aa, ff.

A7: v, h, bb, j, n, z, dd, ff, o.

A8: continuous.

A9: t, f.

A10: t, f.

A11: continuous.

A12: t, f.

A13: g, p, s.

A14: continuous.

A15: continuous.

A16: +,- (class attribute)

Missing Attribute Values: 37 cases (5%) have one or more missing values. The missing values from particular attributes are:

A1: 12

A2: 12

A4: 6

A5: 6

A6: 9

A7: 9

A14: 13

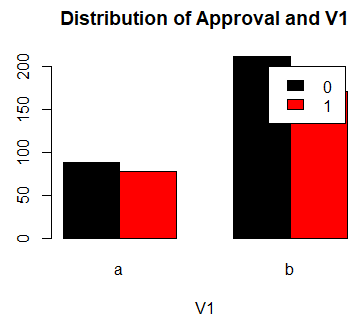
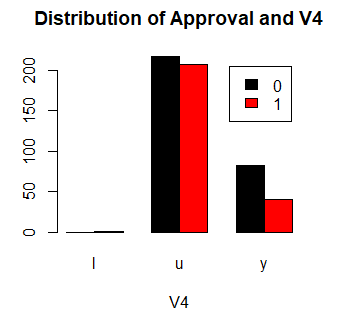
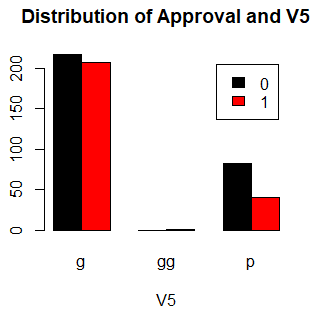
The way I determine which variable are categorical, and which one is continuous from the information I got. **The final variable A16 called it final approval status.** (The R code may refer each variable as V1 instead of A1).

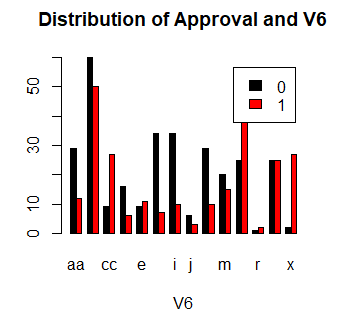
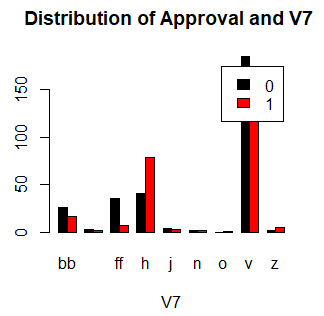
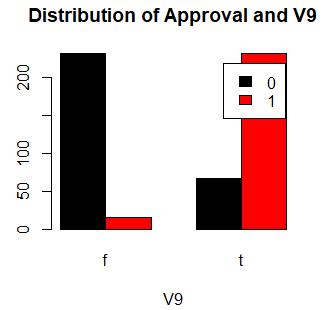
|  |  |
| --- | --- |
| **Categorical** | **Continuous** |
| 1. A1 2. A4 3. A5 4. A6 5. A7 6. A9 7. A10 8. A12 9. A13 | 1. A2 2. A3 3. A8 4. A11 5. A14 6. A15 |

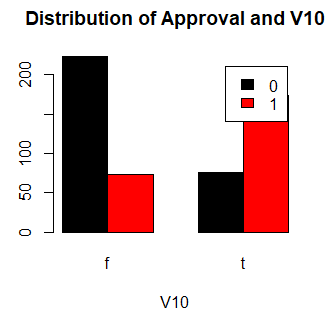
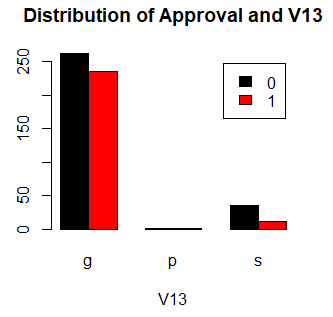
After we determine the variables and replacing some missing attributes. Want to determine which of these variables has main effects have the most significant effect on the final approval rate. First, there are training and testing two different data set. Using training to build and select our model, and then using the testing data set to calculate the prediction model accuracy.

**(3)**

Let’s looking at the plot of categorical data analysis first.

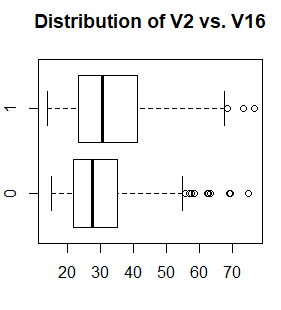
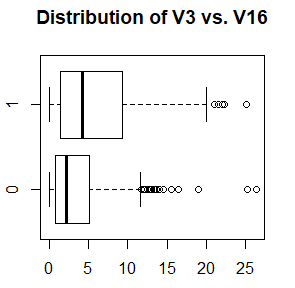
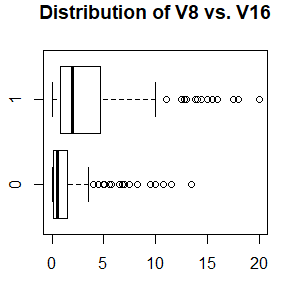
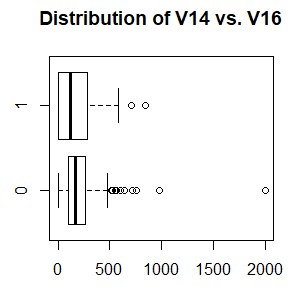
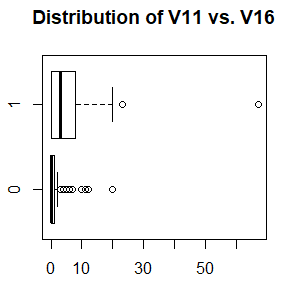
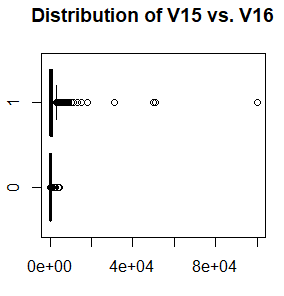
  

From the plot of categorical data, that shown that A9 and A10 are the two most significant factors of the approval rate.

Let’s looking at the plot of continuous data analysis then.

From the boxplot of continuous variables of the data. The plots also shown that A11 and A15 are the two most significant factor of the approval rate.

Now select the model without regularity and the result shown in the results sections. First using stepwise, forward, and backward three methods.

## **Results and Conclusion**

**(4)**

**Selecting Model**

Fit the model using three selection methods: forward, backward, and stepwise selections. Chose the best model by finding the model with the lowest AIC. For the AIC comparison shown below.

|  |  |  |
| --- | --- | --- |
| Selection Methods | Model | AIC |
| Forward | Approved ~V9 + V15+ V11 + V6 + V14 + V4 + V8 | 373.82 |
| Backward | Same | 373.82 |
| Stepwise | Same | 373.82 |

The model processed by three methods are shown that they are the same.

Step: AIC=373.82

V16 ~ V9 + V15 + V11 + V6 + V14 + V4 + V8

Df Deviance AIC

<none> 331.82 373.82

+ V3 1 329.85 373.85

+ V10 1 330.14 374.14

+ V7 7 319.01 375.01

+ V12 1 331.39 375.39

+ V1 1 331.57 375.57

+ V2 1 331.80 375.80

+ V13 2 331.69 377.69

# Checking the model, by looking at the anova.

> anova(select.f)

Analysis of Deviance Table

Model: binomial, link: logit

Response: V16

Terms added sequentially (first to last)

Df Deviance Resid. Df Resid. Dev

NULL 546 753.54

V9 1 316.74 545 436.80

V15 1 40.25 544 396.56

V11 1 19.51 543 377.05

V6 13 33.76 530 343.28

V14 1 3.73 529 339.56

V4 2 5.41 527 334.15

V8 1 2.33 526 331.82

Coefficients:

Estimate Std. Error z value Pr(>|z|)

(Intercept) -6.491e+01 8.830e+02 -0.074 0.941399

V9t 3.380e+00 3.568e-01 9.471 < 2e-16 \*\*\*

V15 8.127e-04 2.275e-04 3.572 0.000354 \*\*\*

V11 1.561e-01 4.855e-02 3.216 0.001298 \*\*

V6c 5.528e-01 5.389e-01 1.026 0.304988

V6cc 1.413e+00 7.322e-01 1.930 0.053660 .

V6d 9.852e-01 8.306e-01 1.186 0.235550

V6e 9.940e-01 9.386e-01 1.059 0.289604

V6ff -8.862e-01 8.525e-01 -1.040 0.298554

V6i -4.615e-01 6.691e-01 -0.690 0.490383

V6j 3.371e-03 1.210e+00 0.003 0.997776

V6k -2.231e-01 6.665e-01 -0.335 0.737761

V6m 4.130e-01 6.632e-01 0.623 0.533489

V6q 6.434e-01 5.764e-01 1.116 0.264302

V6r 1.006e+00 3.597e+00 0.280 0.779696

V6w 1.170e+00 6.145e-01 1.904 0.056912 .

V6x 4.001e+00 1.097e+00 3.648 0.000264 \*\*\*

V14 -2.015e-03 9.490e-04 -2.124 0.033697 \*

V4u 6.174e+01 8.830e+02 0.070 0.944256

V4y 6.096e+01 8.830e+02 0.069 0.944958

V8 7.804e-02 5.348e-02 1.459 0.144484

The model checking that the P-value is significant, and Cooks distances is relatively large. hoslem.test(select.f$y,fitted(fit),g=10)

Hosmer and Lemeshow goodness of fit (GOF) test

data: select.f$y, fitted(fit)

X-squared = 14.289, df = 8, p-value = 0.07453

And the by using Hosmer-Lemeshow Test. The P-value is about 0.074. we will say that the model is a good fit for our data.

Using the leave on out and cross-validation to find the ROC curve and AUC:

From the code

n\_11

[1] 208

n\_10

[1] 40

n\_01

[1] 49

n\_00

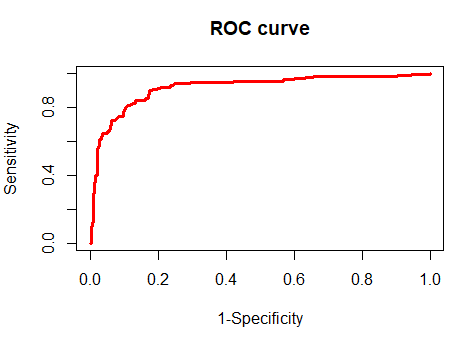
[1] 250

|  |  |  |
| --- | --- | --- |
|  |  |  |
| Approved = 1 | 208 | 40 |
| Approved = 0 | 49 | 250 |

The sensitivity and specificity:

The probability of success and the probability of failure:

The Roc curve shown below.



auc.1

[1] 0.9156867

The AUC number is about 0.9156. The top of the curve towards to the y-value of 1 at the end. Meaning it’s accurate. Using the values from the ROC curve, AUC, and the probabilities determined above to calculate the maximum prediction accuracy and the optimal cutoff point which 85.9% and 0.453 respectively.

pi0[274]

[1] 0.4529058

pred.accuracy.1[274]

[1] 0.8592322

The correspond classification rule:

Applied the classification rule to the test data, and then calculated classification accuracy by using the formula below.

> n\_11

[1] 42

> n\_10

[1] 6

> n\_01

[1] 7

> n\_00

[1] 45

|  |  |  |
| --- | --- | --- |
|  |  |  |
| Approved = 1 | 42 | 6 |
| Approved = 0 | 7 | 45 |

classification accuracy is

**(5)**

We will use the model built from the lasso penalty to determine a classification since the lasso penalty changes the optimization problem from finding the maximum likelihood estimates to obtain the logistic regression coefficients, after applied the lasso penalty, the result shown below.

# the lasso accuracy is about 87%, tuning parameter is 0.0094 and the cut-off point is about 0.616

> max(accuracy)

[1] 0.8720293

> lambda.seq[j]

[1] 0.009406598

> pi0[k]

[1] 0.6161616

The correspond classification rule with lasso penalty:

**(6)**

Applied the classification rule to the test data, and then calculated classification accuracy by using the formula below.

> n\_11

[1] 36

> n\_10

[1] 12

> n\_01

[1] 2

> n\_00

[1] 50

|  |  |  |
| --- | --- | --- |
|  |  |  |
| Approved = 1 | 36 | 12 |
| Approved = 0 | 2 | 50 |

The classification accuracy is:

**Conclusion/Comparison**

Conclusion about this project, the classification rule applied on the logistic regression model without the lasso penalty 87% is a little bit better than the model built with the lasso penalty 86% of predicting the credit applicants get approve or not.

## **Reference**

<https://archive.ics.uci.edu/ml/datasets/Credit+Approval>

## **Appendix**

**R Code:**

> Train.data=read.table("project.1.data.2.train.txt",sep=",",na.strings="?")

> id\_no\_miss=(apply(is.na(Train.data),1,sum)==0)

> data=Train.data[id\_no\_miss,]

> data$V16=as.numeric(data$V16)

> data$V16=data$V16-1

> data$V16=as.factor(data$V16)

> str(data)

'data.frame': 547 obs. of 16 variables:

$ V1 : Factor w/ 2 levels "a","b": 2 1 1 2 2 2 2 1 2 2 ...

$ V2 : num 30.8 58.7 24.5 27.8 20.2 ...

$ V3 : num 0 4.46 0.5 1.54 5.62 ...

$ V4 : Factor w/ 3 levels "l","u","y": 2 2 2 2 2 2 2 2 3 3 ...

$ V5 : Factor w/ 3 levels "g","gg","p": 1 1 1 1 1 1 1 1 3 3 ...

$ V6 : Factor w/ 14 levels "aa","c","cc",..: 13 11 11 13 13 10 12 3 9 13 ...

$ V7 : Factor w/ 9 levels "bb","dd","ff",..: 8 4 4 8 8 8 4 8 4 8 ...

$ V8 : num 1.25 3.04 1.5 3.75 1.71 ...

$ V9 : Factor w/ 2 levels "f","t": 2 2 2 2 2 2 2 2 2 2 ...

$ V10: Factor w/ 2 levels "f","t": 2 2 1 2 1 1 1 1 1 1 ...

$ V11: int 1 6 0 5 0 0 0 0 0 0 ...

$ V12: Factor w/ 2 levels "f","t": 1 1 1 2 1 2 2 1 1 2 ...

$ V13: Factor w/ 3 levels "g","p","s": 1 1 1 1 3 1 1 1 1 1 ...

$ V14: int 202 43 280 100 120 360 164 80 180 52 ...

$ V15: int 0 560 824 3 0 0 31285 1349 314 1442 ...

$ V16: Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 2 2 2 ...

**Ploting**

> boxplot(V2~V16,data=data,horizontal=T,main="Distribution of V2 vs. V16")

> boxplot(V3~V16,data=data,horizontal=T,main="Distribution of V3 vs. V16")

> boxplot(V8~V16,data=data,horizontal=T,main="Distribution of V8 vs. V16")

> boxplot(V11~V16,data=data,horizontal=T,main="Distribution of V11 vs. V16")

> boxplot(V14~V16,data=data,horizontal=T,main="Distribution of V14 vs. V16")

> boxplot(V15~V16,data=data,horizontal=T,main="Distribution of V15 vs. V16")

> ############

> counts <- table(data$V16, data$V1)

> barplot(counts, main="Distribution of Approval and V1", xlab="V1", col=c("black","red"), legend = rownames(counts), beside=TRUE)

> counts <- table(data$V16, data$V4)

> barplot(counts, main="Distribution of Approval and V4", xlab="V4", col=c("black","red"), legend = rownames(counts), beside=TRUE)

> counts <- table(data$V16, data$V5)

> barplot(counts, main="Distribution of Approval and V5", xlab="V5", col=c("black","red"), legend = rownames(counts), beside=TRUE)

> counts <- table(data$V16, data$V6)

> barplot(counts, main="Distribution of Approval and V6", xlab="V6", col=c("black","red"), legend = rownames(counts), beside=TRUE)

> counts <- table(data$V16, data$V7)

> barplot(counts, main="Distribution of Approval and V7 ", xlab="V7", col=c("black","red"), legend = rownames(counts), beside=TRUE)

> counts <- table(data$V16, data$V9)

> barplot(counts, main="Distribution of Approval and V9", xlab="V9", col=c("black","red"), legend = rownames(counts), beside=TRUE)

> counts <- table(data$V16, data$V10)

> barplot(counts, main="Distribution of Approval and V10", xlab="V10", col=c("black","red"), legend = rownames(counts), beside=TRUE)

> counts <- table(data$V16, data$V12)

> barplot(counts, main="Distribution of Approval and V12", xlab="V12", col=c("black","red"), legend = rownames(counts), beside=TRUE)

> counts <- table(data$V16, data$V13)

> barplot(counts, main="Distribution of Approval and V13", xlab="V13", col=c("black","red"), legend = rownames(counts), beside=TRUE)

**Select Model.**

> fit.full=glm(V16~.,data=data,family=binomial)

Warning messages:

1: glm.fit: algorithm did not converge

2: glm.fit: fitted probabilities numerically 0 or 1 occurred

**###################**

> fit.null=glm(V16~1,data=data,family=binomial)

> select.f=step(fit.null,scope=list(lower=fit.null,upper=fit.full),direction="forward")

Start: AIC=755.54

V16 ~ 1

Df Deviance AIC

+ V9 1 436.80 440.80

+ V11 1 612.58 616.58

+ V10 1 640.58 644.58

+ V15 1 657.02 661.02

+ V8 1 681.92 685.92

+ V6 13 660.39 688.39

+ V7 8 710.80 728.80

+ V3 1 735.36 739.36

+ V2 1 738.15 742.15

+ V4 2 741.92 747.92

+ V5 2 741.92 747.92

+ V13 2 744.28 750.28

+ V14 1 749.67 753.67

<none> 753.54 755.54

+ V12 1 752.79 756.79

+ V1 1 753.28 757.28

Step: AIC=440.8

V16 ~ V9

Df Deviance AIC

+ V15 1 396.56 402.56

+ V11 1 406.18 412.18

+ V10 1 411.15 417.15

+ V6 13 399.51 429.51

+ V4 2 424.18 432.18

+ V5 2 424.18 432.18

+ V8 1 429.16 435.16

+ V7 8 419.15 439.15

+ V13 2 431.76 439.76

<none> 436.80 440.80

+ V14 1 434.93 440.93

+ V3 1 436.22 442.22

+ V12 1 436.34 442.34

+ V2 1 436.45 442.45

+ V1 1 436.80 442.80

Step: AIC=402.56

V16 ~ V9 + V15

Df Deviance AIC

+ V11 1 377.05 385.05

+ V10 1 377.86 385.86

+ V6 13 360.22 392.22

+ V8 1 390.13 398.13

+ V5 2 391.12 401.12

+ V4 2 391.12 401.12

+ V14 1 394.33 402.33

<none> 396.56 402.56

+ V7 8 381.57 403.57

+ V12 1 396.24 404.24

+ V1 1 396.49 404.49

+ V2 1 396.54 404.54

+ V3 1 396.55 404.55

+ V13 2 396.11 406.11

Step: AIC=385.05

V16 ~ V9 + V15 + V11

Df Deviance AIC

+ V6 13 343.28 377.28

+ V8 1 372.28 382.28

+ V10 1 373.34 383.34

<none> 377.05 385.05

+ V4 2 373.15 385.15

+ V5 2 373.15 385.15

+ V7 8 362.05 386.05

+ V14 1 376.24 386.24

+ V1 1 376.67 386.67

+ V12 1 376.75 386.75

+ V3 1 376.90 386.90

+ V2 1 377.04 387.04

+ V13 2 376.95 388.95

Step: AIC=377.28

V16 ~ V9 + V15 + V11 + V6

Df Deviance AIC

+ V14 1 339.56 375.56

+ V5 2 338.60 376.60

+ V4 2 338.60 376.60

+ V8 1 340.89 376.89

<none> 343.28 377.28

+ V10 1 341.54 377.54

+ V12 1 342.47 378.47

+ V2 1 342.59 378.59

+ V3 1 342.81 378.81

+ V1 1 343.18 379.18

+ V13 2 343.17 381.17

+ V7 8 332.11 382.11

Step: AIC=375.56

V16 ~ V9 + V15 + V11 + V6 + V14

Df Deviance AIC

+ V4 2 334.15 374.15

+ V5 2 334.15 374.15

+ V8 1 337.05 375.05

<none> 339.56 375.56

+ V10 1 337.89 375.89

+ V3 1 338.45 376.45

+ V2 1 338.72 376.72

+ V12 1 339.10 377.10

+ V1 1 339.44 377.44

+ V13 2 339.38 379.38

+ V7 8 327.70 379.70

Step: AIC=374.15

V16 ~ V9 + V15 + V11 + V6 + V14 + V4

Df Deviance AIC

+ V8 1 331.82 373.82

<none> 334.15 374.15

+ V3 1 332.78 374.78

+ V10 1 332.82 374.82

+ V2 1 333.74 375.74

+ V1 1 333.74 375.74

+ V7 7 321.75 375.75

+ V12 1 333.94 375.94

+ V13 2 334.02 378.02

Step: AIC=373.82

V16 ~ V9 + V15 + V11 + V6 + V14 + V4 + V8

Df Deviance AIC

<none> 331.82 373.82

+ V3 1 329.85 373.85

+ V10 1 330.14 374.14

+ V7 7 319.01 375.01

+ V12 1 331.39 375.39

+ V1 1 331.57 375.57

+ V2 1 331.80 375.80

+ V13 2 331.69 377.69

There were 50 or more warnings (use warnings() to see the first 50)

> summary(select.f)

Call:

glm(formula = V16 ~ V9 + V15 + V11 + V6 + V14 + V4 + V8, family = binomial,

data = data)

Deviance Residuals:

Min 1Q Median 3Q Max

-2.8780 -0.3539 -0.1696 0.4198 3.0179

Coefficients:

Estimate Std. Error z value Pr(>|z|)

(Intercept) -6.491e+01 8.830e+02 -0.074 0.941399

V9t 3.380e+00 3.568e-01 9.471 < 2e-16 \*\*\*

V15 8.127e-04 2.275e-04 3.572 0.000354 \*\*\*

V11 1.561e-01 4.855e-02 3.216 0.001298 \*\*

V6c 5.528e-01 5.389e-01 1.026 0.304988

V6cc 1.413e+00 7.322e-01 1.930 0.053660 .

V6d 9.852e-01 8.306e-01 1.186 0.235550

V6e 9.940e-01 9.386e-01 1.059 0.289604

V6ff -8.862e-01 8.525e-01 -1.040 0.298554

V6i -4.615e-01 6.691e-01 -0.690 0.490383

V6j 3.371e-03 1.210e+00 0.003 0.997776

V6k -2.231e-01 6.665e-01 -0.335 0.737761

V6m 4.130e-01 6.632e-01 0.623 0.533489

V6q 6.434e-01 5.764e-01 1.116 0.264302

V6r 1.006e+00 3.597e+00 0.280 0.779696

V6w 1.170e+00 6.145e-01 1.904 0.056912 .

V6x 4.001e+00 1.097e+00 3.648 0.000264 \*\*\*

V14 -2.015e-03 9.490e-04 -2.124 0.033697 \*

V4u 6.174e+01 8.830e+02 0.070 0.944256

V4y 6.096e+01 8.830e+02 0.069 0.944958

V8 7.804e-02 5.348e-02 1.459 0.144484

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 753.54 on 546 degrees of freedom

Residual deviance: 331.82 on 526 degrees of freedom

AIC: 373.82

Number of Fisher Scoring iterations: 13

> anova(select.f)

Analysis of Deviance Table

Model: binomial, link: logit

Response: V16

Terms added sequentially (first to last)

Df Deviance Resid. Df Resid. Dev

NULL 546 753.54

V9 1 316.74 545 436.80

V15 1 40.25 544 396.56

V11 1 19.51 543 377.05

V6 13 33.76 530 343.28

V14 1 3.73 529 339.56

V4 2 5.41 527 334.15

V8 1 2.33 526 331.82

> cooks.distance<-cooks.distance(fit)

> which(cooks.distance>1)

named integer(0)

> hoslem.test(select.f$y,fitted(fit),g=10)

Hosmer and Lemeshow goodness of fit (GOF) test

data: select.f$y, fitted(fit)

X-squared = 14.289, df = 8, p-value = 0.07453

> library(car)

Loading required package: carData

Warning message:

package ‘car’ was built under R version 3.5.3

> vif(fit)

GVIF Df GVIF^(1/(2\*Df))

V9 1.181118 1 1.086793

V11 1.088524 1 1.043324

V15 1.086682 1 1.042441

V6 1.326315 13 1.010921

V14 1.136687 1 1.066155

V8 1.093229 1 1.045576

(4) ROC, AUC, and leave one out and cross-validation.

pi0=seq(1,0, length.out = 500)

nsample=nrow(data)

roc.1=NULL

for (k in 1:length(pi0))

{

nsample=nrow(data)

n\_11.1=n\_11.2=n\_10.1=n\_10.2=n\_01.1=n\_01.2=n\_00.1=n\_00.2=0

for (i in 1:nsample)

{

training=data[-i,]

test=data[i,]

fit.1.training=glm(V16 ~ V9 + V15 + V11 + V6+ V14 + V8, data=training, family=binomial)

if(predict(fit.1.training, test, type="response")>=pi0[k])

{

Y.pred.1=1

} else

{Y.pred.1=0}

if((test$V16==1)&(Y.pred.1==1))

{n\_11.1=n\_11.1+1}

if((test$V16==1)&(Y.pred.1==0))

{n\_10.1=n\_10.1+1}

if((test$V16==0)&(Y.pred.1==1))

{n\_01.1=n\_01.1+1}

if((test$V16==0)&(Y.pred.1==0))

{n\_00.1=n\_00.1+1}

}

sen=n\_11.1/(n\_10.1+n\_11.1)

spe=n\_00.1/(n\_00.1+n\_01.1)

roc.1=rbind(roc.1, c(1-spe, sen))

}

>plot(roc.1, type="s",xlim=c(0,1), ylim=c(0,1), col="red", lwd=3, main="ROC curve", xlab="1-Specificity", ylab="Sensitivity").

>auc.1=sum(roc.1[-500, 2]\*(roc.1[-1,1]-roc.1[-500,1]))

auc.1

[1] 0.9156867

#The cross-validation.

nsample=nrow(data)

n\_11=0

n\_10=0

n\_01=0

n\_00=0

for(i in 1:nsample)

{

training=data[-i,]

test=data[i,]

fit.1.training=glm(V16~ V9 + V15 + V11 + V6+ V14 + V8, data=training, family=binomial)

if(predict(fit.1.training, test, type="response")>0.5)

{

Y.pred=1

} else

{Y.pred=0}

if((test$V16==1)&(Y.pred==1))

{n\_11=n\_11+1}

if((test$V16==1)&(Y.pred==0))

{n\_10=n\_10+1}

if((test$V16==0)&(Y.pred==1))

{n\_01=n\_01+1}

if((test$V16==0)&(Y.pred==0))

{n\_00=n\_00+1}

}

There were 50 or more warnings (use warnings() to see the first 50)

n\_11

[1] 208

n\_10

[1] 40

n\_01

[1] 49

n\_00

[1] 250

pred.accuracy.1=roc.1[,2]\*(208+40)/(40+208+49+250)+(1-roc.1[,1])\*(250+49)/(49+250+40+208)

which.max(pred.accuracy.1)

[1] 274

pi0[274]

[1] 0.4529058

pred.accuracy.1[274]

[1] 0.8592322

> Test.data=read.table("project.1.data.2.test.txt",sep=",",na.strings="?")

> id\_no\_miss=(apply(is.na(Test.data), 1,sum)==0)

> data.1=Test.data[id\_no\_miss,]

> colnames(data.1)[16]="Approved"

> data.1$Approved=as.numeric(data.1$Approved)

> data.1$Approved=data.1$Approved-1

> data.1$Approved=as.factor(data.1$Approved)

> nsample=nrow(data.1)

> n\_11=0

> n\_10=0

> n\_01=0

> n\_00=0

> for(i in 1:nsample)

+ {

+ test=data.1[i,]

+ if(predict(fit.1.training, test, type="response")>0.4529058)

+ {

+ Y.pred=1

+ } else

+ {Y.pred=0}

+ if((data.1$Approved[i]==1)&(Y.pred==1))

+ {n\_11=n\_11+1}

+ if((data.1$Approved[i]==1)&(Y.pred==0))

+ {n\_10=n\_10+1}

+ if((data.1$Approved[i]==0)&(Y.pred==1))

+ {n\_01=n\_01+1}

+ if((data.1$Approved[i]==0)&(Y.pred==0))

+ {n\_00=n\_00+1}

+ }

>

> n\_11

[1] 42

> n\_10

[1] 6

> n\_01

[1] 7

> n\_00

[1] 45

|  |
| --- |
| > Train=read.table("project.1.data.2.train.txt",sep=",",na.strings="?")  > Test=read.table("project.1.data.2.test.txt",sep=",",na.strings="?")  > All=rbind(Train, Test)  > id\_no\_miss=(apply(is.na(All), 1,sum)==0)  > data.2=All[id\_no\_miss,]  > colnames(data.2)[16]="Approved"  >  > data.2$Approved=as.numeric(data.2$Approved)  > data.2$Approved=data.2$Approved-1  > data.2$Approved=as.factor(data.2$Approved)  > fit.all=glm(Approved~., family = binomial, data = data.2)  Warning message:  glm.fit: fitted probabilities numerically 0 or 1 occurred  > y.all= data.2$Approved  > x.all=model.matrix(fit.all)[,-1]  > y=y.all[1:547]  > x=x.all[1:547,]  > library(glmnet)  Loading required package: Matrix  Loading required package: foreach  Loaded glmnet 2.0-16  Warning message:  package ‘glmnet’ was built under R version 3.5.3  > fit.lasso=glmnet(x, y, family="binomial")  > lambda.seq=fit.lasso$lambda  > pi0=seq(0,1, length.out = 100)  > correct.num=matrix(0, length(lambda.seq), length(pi0))  > nsample=nrow(x)  > for(i in 1:nsample)  + {  + x.train=x[-i,]  + x.test=matrix(x[i,],1,ncol(x))  + y.train=y[-i]  + y.test=y[i]  + fit.lasso.training=glmnet(x.train, y.train, family="binomial")  + for(j in 1:length(lambda.seq))  + {  + pred.prob=predict(fit.lasso.training, newx=x.test, s=lambda.seq[j], type="response")  + for(k in 1:length(pi0))  + {  + if(pred.prob>=pi0[k])  + {  + Y.pred.1=1  + }else  + {Y.pred.1=0}  + if((y.test==1)&(Y.pred.1==1))  + {correct.num[j,k]=correct.num[j,k]+1}  + if((y.test==0)&(Y.pred.1==0))  + {correct.num[j,k]=correct.num[j,k]+1}  + }  + }  + }  > accuracy=correct.num/nsample  > accuracy=correct.num/nsample  > max(accuracy)  [1] 0.8720293 |
| > which(accuracy==max(accuracy), arr.ind=TRUE)  row col  [1,] 40 62  > j=40  > k=62  > lambda.seq[j]  [1] 0.009406598  > pi0[k]  [1] 0.6161616  > lambda.opt=lambda.seq[j]  > fit.lasso=glmnet(x, y, family="binomial")  > coef(fit.lasso, s=lambda.opt)  38 x 1 sparse Matrix of class "dgCMatrix"  1  (Intercept) -2.6877747486  V1b .  V2 .  V3 .  V4u 0.0500094981  V4y -0.3829907716  V5gg .  V5p -0.0021884075  V6c .  V6cc 0.4990450131  V6d .  V6e .  V6ff -0.8085554490  V6i -0.4603112421  V6j .  V6k -0.4342264015  V6m .  V6q .  V6r .  V6w 0.4039452116  V6x 2.0706121872  V7dd .  V7ff .  V7h 0.3053003060  V7j 0.1695685636  V7n 1.1548814601  V7o .  V7v .  V7z .  V8 0.0546771117  V9t 3.0363527137  V10t 0.5695866234  V11 0.0930922495  V12t .  V13p .  V13s .  V14 -0.0009391974  V15 0.0001106858 |
| |  | | --- | | > Y.test=y.all[-(1:547)]  > X.test=x.all[-(1:547),]  > Y.pred=as.vector(predict(fit.lasso, newx=X.test, s=lambda.opt, type="response")>pi0[k])  > Y.pred=1\*Y.pred  > n\_11=0  > n\_10=0  > n\_01=0  > n\_00=0  > for(i in 1:100)  + {  + if((Y.test[i]==1)&(Y.pred[i]==1))  + {n\_11=n\_11+1}  + if((Y.test[i]==1)&(Y.pred[i]==0))  + {n\_10=n\_10+1}  + if((Y.test[i]==0)&(Y.pred[i]==1))  + {n\_01=n\_01+1}  + if((Y.test[i]==0)&(Y.pred[i]==0))  + {n\_00=n\_00+1}  + }  > n\_11  [1] 36  > n\_10  [1] 12  > n\_01  [1] 2  > n\_00  [1] 50 | |