# Logistic Model to classify the kddcup99 data

## Process of the data

* The last column is the label, we treat them as two part. One is” normal “and the other is” attack”
* The second, third and fourth column are characters. We use the following code to transfer these character into numbers:

for (i in 1:41){

kddcup\_train[,i]=as.numeric(kddcup\_train[,i])

kddcup\_test[,i]=as.numeric(kddcup\_test[,i])

}

* Finally we transfer the data into matrix only with numbers

## Implement of the data

### Cross validation to find the optimal lambda

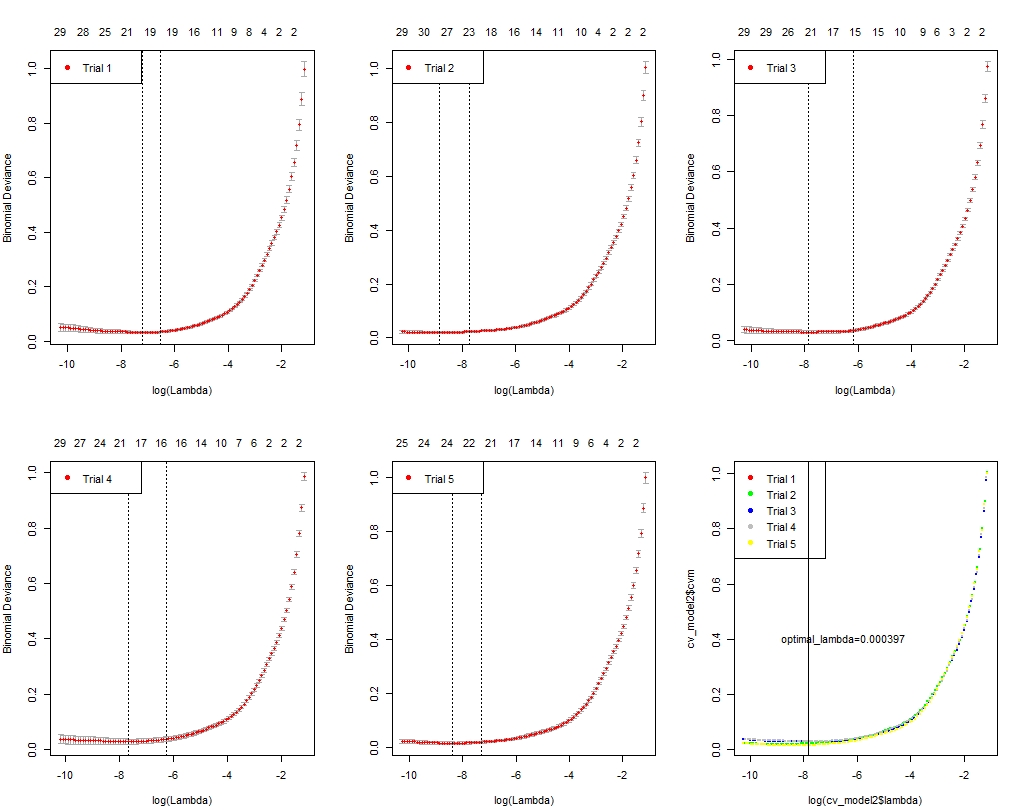
We implement the data with cross validation (generalized linear model) to find the optimal lambda with the following model:

cv\_model=cv.glmnet(a1[1:5000,-42],a1[1:5000,42],family="binomial")

I sample the data with size 5000 and 2000 respectively. 5000 data is the training data and 2000 data is the testing data.

I run the algorithm 5 times and you can see the figures in the next page.

From the figures, we can see that the optimal lambda is 0.000397.



### Train with larger size training data (20000) and larger size test data(5000)

model=glmnet(kddcup\_train[1:20000,-42],kddcup\_train[1:20000,42],family="binomial")

#do test on test data

start\_time=proc.time()

pre.test=predict(model,kddcup\_test[1:5000,-42],0.000397,"class")

end\_time=proc.time()

run\_time=end\_time-start\_time

errorrate.test=sum((pre.test!=kddcup\_test[,42]))/nrow(kddcup\_test)

The result we get are in the following:

> errorrate.test

[1] 0.0084

> run\_time

user system elapsed

0.02 0.03 2.66

|  |  |
| --- | --- |
| Average time of one data stream(s) | Error rate |
| 1.33e(-4) | 0.0084 |

# Code Appendix :

#logistic model of kddcup99 classification

#this is the package we used

library("glmnet")

#import the data into the R studio

kddcup <- read.csv("C:/Users/Christina/Desktop/kddcup.data\_10\_percent\_corrected", header=FALSE)

# we sample the data to get 5000 training data and 2000 test data

kddcup=as.matrix(kddcup)

data=function(kddcup){

kddcup\_1=sample((nrow(kddcup)),size=5000,replace = FALSE, prob = NULL)

kddcup\_2=sample((nrow(kddcup)),size=2000,replace = FALSE, prob = NULL)

kddcup\_train=kddcup[kddcup\_1,]

kddcup\_test=kddcup[kddcup\_2,]

#process the data in order to apply the logistic model

# process the last column into two parts

for (i in 1:2000){

if (kddcup\_test[i,42]=="normal."){

kddcup\_test[i,42]=1}

else{

kddcup\_test[i,42]=2

}

}

for (i in 1:5000){

if (kddcup\_train[i,42]=="normal."){

kddcup\_train[i,42]=1}

else{

kddcup\_train[i,42]=2

}

}

kddcup\_train=as.data.frame(kddcup\_train)

kddcup\_test=as.data.frame(kddcup\_test)

colnames(kddcup\_train)=c(paste("x.",1:41,sep=""),"Y")

colnames(kddcup\_test)=c(paste("x.",1:41,sep=""),"Y")

#process the data in order to apply the svm method

#process the x and through this way we can transfer the characters into numbers

for (i in 1:41){

kddcup\_train[,i]=as.numeric(kddcup\_train[,i])

kddcup\_test[,i]=as.numeric(kddcup\_test[,i])

}

kddcup\_train[,42]=as.factor(kddcup\_train[,42])

kddcup\_train[,2]=as.factor(kddcup\_train[,2])

kddcup\_train[,3]=as.factor(kddcup\_train[,3])

kddcup\_train[,4]=as.factor(kddcup\_train[,4])

kddcup\_train[,7]=as.factor(kddcup\_train[,7])

kddcup\_train[,12]=as.factor(kddcup\_train[,12])

kddcup\_train[,21]=as.factor(kddcup\_train[,21])

kddcup\_train[,22]=as.factor(kddcup\_train[,22])

kddcup\_test[,42]=as.factor(kddcup\_test[,42])

kddcup\_test[,2]=as.factor(kddcup\_test[,2])

kddcup\_test[,3]=as.factor(kddcup\_test[,3])

kddcup\_test[,4]=as.factor(kddcup\_test[,4])

kddcup\_test[,7]=as.factor(kddcup\_test[,7])

kddcup\_test[,12]=as.factor(kddcup\_test[,12])

kddcup\_test[,21]=as.factor(kddcup\_test[,21])

kddcup\_test[,22]=as.factor(kddcup\_test[,22])

kddcup\_train=data.matrix(kddcup\_train)

kddcup\_test=data.matrix(kddcup\_test)

kddcup\_train\_test=rbind(kddcup\_train,kddcup\_test)

return (kddcup\_train\_test)

}

# first we use the generalized linear model and cross validation to deternime the optimal lambda

#we run the algorithm 5 times

a1=data(kddcup)

a2=data(kddcup)

a3=data(kddcup)

a4=data(kddcup)

a5=data(kddcup)

cv\_model1=cv.glmnet(a1[1:5000,-42],a1[1:5000,42],family="binomial")

cv\_model2=cv.glmnet(a2[1:5000,-42],a2[1:5000,42],family="binomial")

cv\_model3=cv.glmnet(a3[1:5000,-42],a3[1:5000,42],family="binomial")

cv\_model4=cv.glmnet(a4[1:5000,-42],a4[1:5000,42],family="binomial")

cv\_model5=cv.glmnet(a5[1:5000,-42],a5[1:5000,42],family="binomial")

#plot the figures to find optimal lambda

par(mfrow=c(2,3))

plot(cv\_model1,pch=19,col="red")

legend("topleft",legend="Trial 1",col="red",pch=19)

plot(cv\_model2,pch=19,col="green")

legend("topleft",legend="Trial 2",col="red",pch=19)

plot(cv\_model3,pch=19,col="blue")

legend("topleft",legend="Trial 3",col="red",pch=19)

plot(cv\_model4,pch=19,col="grey")

legend("topleft",legend="Trial 4",col="red",pch=19)

plot(cv\_model5,pch=19,col="yellow")

legend("topleft",legend="Trial 5",col="red",pch=19)

plot(log(cv\_model2$lambda),cv\_model2$cvm,pch=19,col="red",cex=0.1)

points(log(cv\_model2$lambda),cv\_model2$cvm,pch=19,col="green",cex=0.1)

points(log(cv\_model3$lambda),cv\_model3$cvm,pch=19,col="blue",cex=0.1)

points(log(cv\_model4$lambda),cv\_model4$cvm,pch=19,col="grey",cex=0.1)

points(log(cv\_model5$lambda),cv\_model5$cvm,pch=19,col="yellow",cex=0.1)

legend("topleft",legend=c('Trial 1','Trial 2','Trial 3','Trial 4','Trial 5'),col=c('red','green','blue','grey','yellow'),pch=19)

abline(v=log((cv\_model1$lambda.min+cv\_model2$lambda.min+cv\_model3$lambda.min+cv\_model4$lambda.min+cv\_model5$lambda.min)/5))

text(x=-6.5,y=0.4,label="optimal\_lambda=0.000397")

# we train the model with a larger training data,say 20000 training data and 5000 testing data

kddcup\_1=sample((nrow(kddcup)),size=20000,replace = FALSE, prob = NULL)

kddcup\_2=sample((nrow(kddcup)),size=5000,replace = FALSE, prob = NULL)

kddcup\_train=kddcup[kddcup\_1,]

kddcup\_test=kddcup[kddcup\_2,]

#process the data in order to apply the logistic model

# process the last column into two parts

for (i in 1:5000){

if (kddcup\_test[i,42]=="normal."){

kddcup\_test[i,42]=1}

else{

kddcup\_test[i,42]=2

}

}

for (i in 1:20000){

if (kddcup\_train[i,42]=="normal."){

kddcup\_train[i,42]=1}

else{

kddcup\_train[i,42]=2

}

}

kddcup\_train=as.data.frame(kddcup\_train)

kddcup\_test=as.data.frame(kddcup\_test)

colnames(kddcup\_train)=c(paste("x.",1:41,sep=""),"Y")

colnames(kddcup\_test)=c(paste("x.",1:41,sep=""),"Y")

#process the data in order to apply the svm method

#process the x and through this way we can transfer the characters into numbers

for (i in 1:41){

kddcup\_train[,i]=as.numeric(kddcup\_train[,i])

kddcup\_test[,i]=as.numeric(kddcup\_test[,i])

}

kddcup\_train[,42]=as.factor(kddcup\_train[,42])

kddcup\_train[,2]=as.factor(kddcup\_train[,2])

kddcup\_train[,3]=as.factor(kddcup\_train[,3])

kddcup\_train[,4]=as.factor(kddcup\_train[,4])

kddcup\_train[,7]=as.factor(kddcup\_train[,7])

kddcup\_train[,12]=as.factor(kddcup\_train[,12])

kddcup\_train[,21]=as.factor(kddcup\_train[,21])

kddcup\_train[,22]=as.factor(kddcup\_train[,22])

kddcup\_test[,42]=as.factor(kddcup\_test[,42])

kddcup\_test[,2]=as.factor(kddcup\_test[,2])

kddcup\_test[,3]=as.factor(kddcup\_test[,3])

kddcup\_test[,4]=as.factor(kddcup\_test[,4])

kddcup\_test[,7]=as.factor(kddcup\_test[,7])

kddcup\_test[,12]=as.factor(kddcup\_test[,12])

kddcup\_test[,21]=as.factor(kddcup\_test[,21])

kddcup\_test[,22]=as.factor(kddcup\_test[,22])

kddcup\_train=data.matrix(kddcup\_train)

kddcup\_test=data.matrix(kddcup\_test)

kddcup\_train\_test=rbind(kddcup\_train,kddcup\_test)

#get the model

model=glmnet(kddcup\_train[1:20000,-42],kddcup\_train[1:20000,42],family="binomial")

#do test on test data

start\_time=proc.time()

pre.test=predict(model,kddcup\_test[1:5000,-42],0.000397,"class")

end\_time=proc.time()

run\_time=end\_time-start\_time

errorrate.test=sum((pre.test!=kddcup\_test[,42]))/nrow(kddcup\_test)

#

kddcup\_train\_x=data.matrix(subset(kddcup\_train,select=-Y))

kddcup\_train\_y=subset(kddcup\_train,select=Y)

kddcup\_test\_x=subset(kddcup\_test,select=-Y)

kddcup\_test\_y=subset(kddcup\_test,select=Y)

model=glmnet(kddcup\_train\_x,kddcup\_train\_y,"binomial")

> errorrate.test

[1] 0.0084

> run\_time

user system elapsed

0.02 0.03 2.66