Castellano-Lab08

> Download the winequality-red_binary.csv file from Canvas.

```
data = read.csv("C:/Users/Castellano/Downloads/winequality-red_binary.csv")
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(glmnet)
```

```
## Warning: package 'glmnet' was built under R version 3.6.3
```

```
## Loading required package: Matrix
```

```
## Loaded glmnet 3.0-2
```

```
library(ROCR)
```

```
## Warning: package 'ROCR' was built under R version 3.6.3
```

```
## Loading required package: gplots
```

```
## Warning: package 'gplots' was built under R version 3.6.3
```

```
##
## Attaching package: 'gplots'
```

```
## The following object is masked from 'package:stats':
##
##     lowess
```

```
library(e1071)
```

```
## Warning: package 'e1071' was built under R version 3.6.3
```

Classification

Use the first half of the training data (rows) to train a Logistic Regression model to make prediction for the independent variable “quality_bin”,

```
N <- nrow(data)
train_data <- data[1:(N/2),]
test_data <- data[-(1:(N/2)),]

train_y <- select(train_data, quality_bin)
train_x <- select(train_data, -id, -quality_bin)
```

Logistic Model

```
glm1 <- glm(quality_bin ~
              fixed.acidity +
              volatile.acidity +
              citric.acid +
              residual.sugar +
              chlorides +
              free.sulfur.dioxide +
              density +
              pH +
              sulphates +
              alcohol,
            family="binomial",
            data=train_data)
```

Prediction over Training Data

```
train.y_hat=factor(as.numeric(predict(glm1, newdata=train_data, type = "response")>0.
5))
```

Confusion Table

```
table(train_data$quality_bin, train.y_hat)
```

```
##    train.y_hat
##       0   1
##   0 345  80
##   1 135 239
```

Accuracy

```
mean(train_data$quality_bin == train.y_hat)
```

```
## [1] 0.7309136
```

and then test on the second half of the training data. What is the classification error you obtain? What is the AUC of the testing? Make a ROC curve of it.

Prediction over Testing Data, Confusion Table, Accuracy

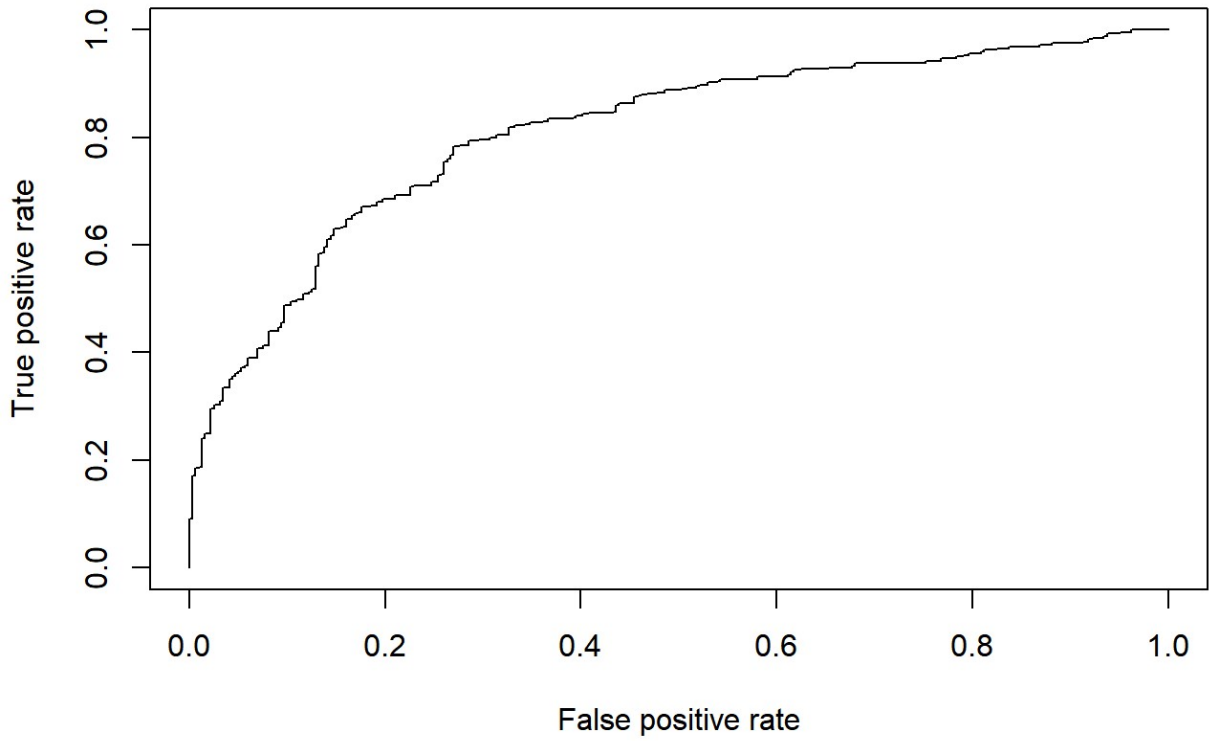
```
test.y_hat=factor(as.numeric(predict(glm1, newdata=test_data, type= "response")>0.5))
table(test_data$quality_bin, test.y_hat)
```

```
##    test.y_hat
##       0   1
##   0 215 104
##   1  92 389
```

```
mean(test_data$quality_bin == test.y_hat)
```

```
## [1] 0.755
```

```
test.y <- test_data$quality_bin
test.y_hat <- predict(glm1, newdata=test_data, type= "response")
test.auc<- performance(prediction(test.y_hat,test.y),"auc")
test.pred <- prediction(test.y_hat, test.y)
test.perf <- performance(test.pred, "tpr","fpr")
plot(test.perf)
```




```
test.auc@y.values
```

```
## [[1]]
## [1] 0.8102112
```

Classification Error

Conduct 5-fold cross-validation using the entire training data to estimate the classification error. Is it larger or smaller than the classification error you obtain in (1)? Which one will be closer to the true classification error of your classifier, do you think?

```
library(dplyr)
library(glmnet)
library(ROCR)
library(e1071)
```

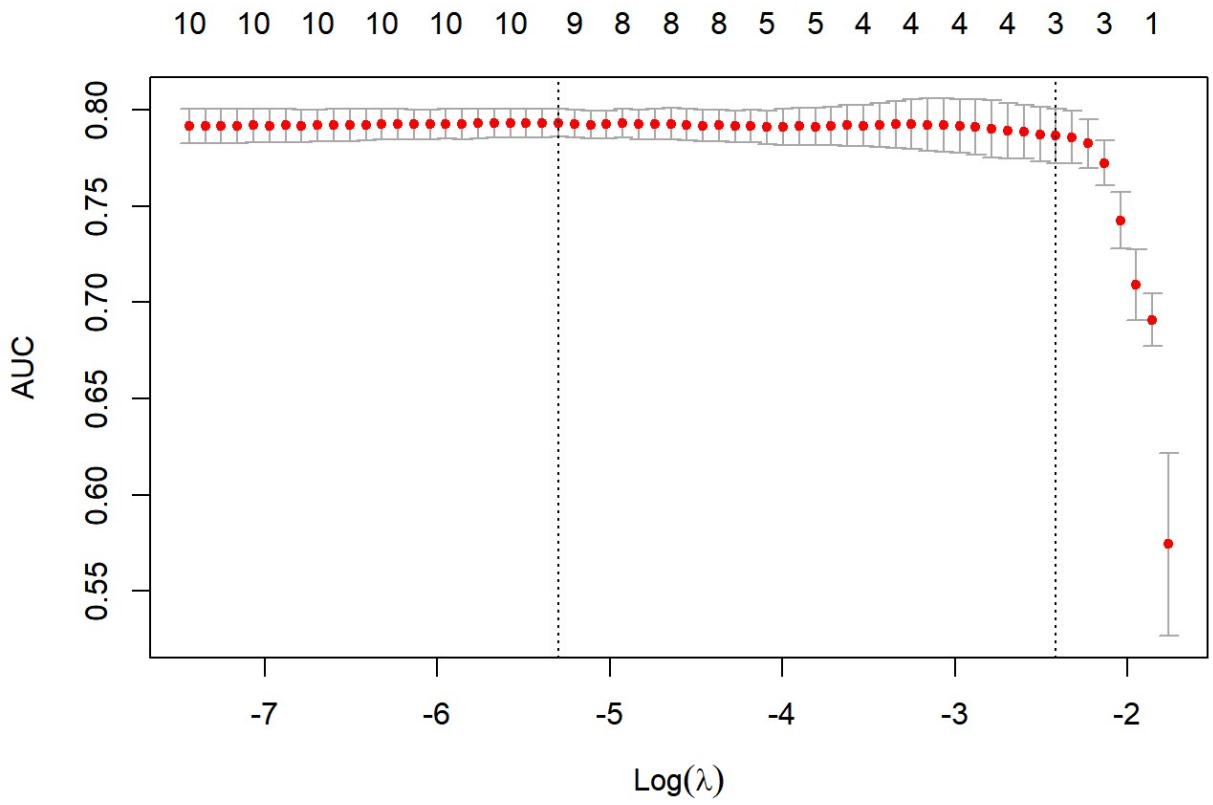
```
x<- select(data, -id, -quality_bin) #remove ID column, and Target Column
y<- data$quality_bin
#remove constant features
x1<- x[,apply(x, 2, sd)>0]

train.inx<- 1:799

train.x<- as.matrix(x1[train.inx,]); train.y<- as.factor(y[train.inx])
test.x<- as.matrix(x1[-train.inx,]); test.y<- as.factor(y[-train.inx])
```

Logistic regression

```
glmnet.fit<-cv.glmnet(x=train.x, y=train.y,family="binomial",type.measure="auc",nfolds
=5)
plot(glmnet.fit)
```




```
train.yhat<- predict(glmnet.fit,type="response",newx=train.x)
train.auc<- performance(prediction(train.yhat[,1],train.y),"auc")@y.values[[1]]
```

```
test.yhat<- predict(glmnet.fit,type="response",newx=test.x)
test.auc<- performance(prediction(test.yhat[,1],test.y),"auc")@y.values[[1]]
```

```
train.auc
```

```
## [1] 0.7879302
```

```
test.auc
```

```
## [1] 0.8093444
```

```
test.pred <- prediction(test.yhat[,1],test.y)
test.perf <- performance(test.pred,"tpr","fpr")
plot(test.perf)
```