Tests Comparison

Elena Kargopoltsev

21 02 2021

Question #1.Read and revise the kc house sales data set

```
House=read.csv("kc house sales.csv")

House$date=as.Date(House$date,format="%Y%m%d")

House$view=as.factor(House$view)

House$waterfront=as.factor(House$waterfront)

House$zipcode=as.factor(House$zipcode)

House[15871,3]=3

attach(House)
```

Question #2.Use the glm() function to perform a multiple regression with log(price) as the response and all other variables except sqft_basement in House as the predictors. Add an argument, family=gaussian within the glm() function to perform a linear regression.

```
glm.fit=glm(log(price)~.-sqft_basement,data=House,family=gaussian)
```

Question #3.Use the cv.glm() function to perform 10-fold cross validation of the model developed in question 2. Use set.seed(100) before performing the cross validation. The cv.glm() function is available in the boot package. Use the names() function to print the names of the variables available in the cross validation output object. Among the variables, delta contains the test MSE. Print the test MSE using this variable.

```
library(boot)

set.seed(100)

cv.err=cv.glm(House,glm.fit, K=10)

names(cv.err)
```

```
## [1] "call"  "K"     "delta" "seed"
```

```
cv.err$delta
```

```
## [1] 0.03364500 0.03362814
```

Question #4.Use the glm() function to perform a multiple regression with log(price) as the response and all other variables except sqft_basement in House and their interactions except zipcode as the predictors. Again, add an argument, family=gaussian within the glm() function to perform a linear regression.

```
glm.fit2=glm(log(price)~(.-sqft_basement-zipcode)^2+zipcode,House,family=gaussian)
```

Question #5.Use the cv.glm() function to perform 10-fold cross validation of the model developed in question 4. Use set.seed(100) before performing the cross validation. Print the test MSE. The cross validation will take about 50 seconds and result in some warnings due to a rank deficiency.

```
set.seed(100)

cv.err2=cv.glm(House, glm.fit2, K=10)

names(cv.err2)
```

```
## [1] "call"  "K"     "delta" "seed"
```

```
cv.err2$delta
```

```
## [1] 0.03125455 0.03114150
```

Question #6.Compare the test MSE of the above two models. Is the difference between the two models practically significant? The response variable of the two models is log(price) instead of price. Thus, to interpret the test MSE correctly, we need to convert it back to its original unit using the exp() or exponential function.

```
exp(cv.err$delta)
```

```
## [1] 1.034217 1.034200
```

```
exp(cv.err2$delta)
```

```
## [1] 1.031748 1.031631
```

#Difference between the two model is not practically significant, the second model is better because has a lower MSE.

Question #7.Use the read.csv() function to read the bank marketing data into R. Call the loaded data Bank. Convert the class of variable y from character to factor using Bank$y = as.factor(Bank$ y). Attach Bank to the R search path.

```
Bank=read.csv("bank marketing.csv")

Bank$y=as.factor(Bank$y)

attach(Bank)
```

Question #8.Perform a logistic regression with y as the response and all other variables except duration as predictors.

```
glm.fit3=glm(y~. -duration, data=Bank, family=binomial)
```

Question #9.Use the cv.glm() function to perform 10-fold cross validation of the logistic regression model. Use set.seed(200) before performing the cross validation. Print the test error rate. In order to use this function for estimating the test error rate of a classification model, a new cost function needs to be defined as explained on slide 18 of PA5. The cross validation will take about 30 seconds and result in some warnings due to a rank deficiency.

```
set.seed(200)

costfunction=function(r, pi){
err1=ifelse(r==1 & pi<0.5, 1, 0)
err0=ifelse(r==0 & pi>0.5, 1, 0)
return(mean(err1+err0))
}

cv.err3=cv.glm(Bank, glm.fit3, K=10)

cv.err3$delta
```

```
## [1] 0.07845008 0.07842966
```

Question #10.The cv.glm() function provides the test error rate of a classification model, but it does not provide the the predicted class and the posterior probability of each observation in the data set, which are required to construct a confusion matrix and an ROC curve. Use the general approach explained on slides 27 and 28 of PA5 to perform 10-fold cross validation of the logistic regression model. The cross validation will take about 30 seconds and result in some warnings due to a rank deficiency.

```
set.seed(10)

Bank2=Bank[sample(nrow(Bank)),]
k=10
bank.prob=rep(0,nrow(Bank))
bank.pred=rep("no", nrow(Bank))
folds=rep(1:k, length=nrow(Bank))

for(i in 1:k){
fold = which(folds == i)
glm.fit4=glm(y~. -duration, data=Bank[-fold,], family=binomial)
bank.prob[fold]=predict(glm.fit4, newdata=Bank[fold,], type="response")
bank.pred[fold]=ifelse(bank.prob[fold]>0.5, "yes","no")}
```

Question #11.Use the table() function to construct a confusion matrix based on the predicted classes obtained in question 10. In the confusion matrix, the actual classes of the reshuffled data set in the general approach should be used. Compute the test error rate.

```
table(bank.pred,Bank$y)
```

```
##
## bank.pred    no   yes
##        no  35989  3588
##        yes   559  1052
```

```
sum(Bank$y!=bank.pred)/nrow(Bank)
```

```
## [1] 0.1006847
```

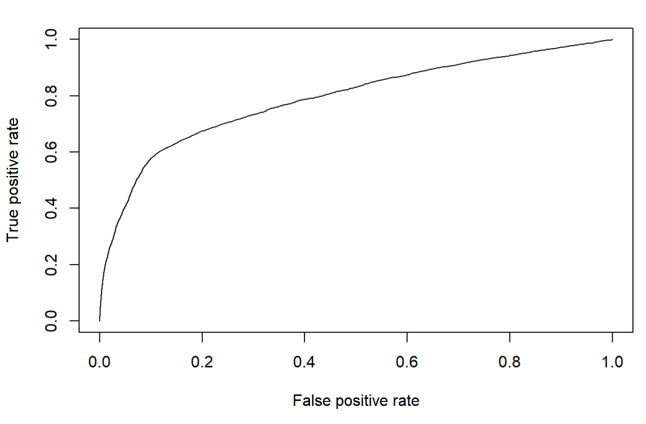
#Test error rate in 9 question lower, so this test error rate not consistent.

Question #12.Construct an ROC curve and find the AUC for the logistic regression model based on the posterior probabilities obtained in question 10.

```
library(ROCR)

rocplot=function(pred, truth, ...){
  predob=prediction(pred, truth)
  perf=performance(predob, "tpr", "fpr")
  plot(perf,...)}

rocplot(bank.prob, Bank$y)
```




```
auc=function(pred, truth){
  predob=prediction(pred, truth)
  perf=performance(predob, "auc")
  return(perf@y.values[[1]])}

auc(bank.prob, Bank$y)
```

```
## [1] 0.7913395
```

Question #13.Perform LDA with y as the response and all other variables except duration as predictors. In the lda() function, add CV=T to get cross validation results. Use the names() function to print the names of the variables in the LDA output object. There will be some warnings due to collinearity.

```
library(MASS)
```

```
##
## Attaching package: 'MASS'
```

```
## The following object is masked from 'Bank':
##
##     housing
```

```
lda.fit=lda(y~. -duration,data=Bank, CV=T)

names(lda.fit)
```

```
## [1] "class"     "posterior" "terms"     "call"      "xlevels"
```

Qestion #14.Construct a confusion matrix for the LDA model using class in the output object obtained in question 13. Compute the test error rate.

```
table(lda.fit$class,y)
```

```
##      y
##          no   yes
##   no  34918  2916
##   yes  1630  1724
```

```
sum(lda.fit$class!=y)/nrow(Bank)
```

```
## [1] 0.110372
```

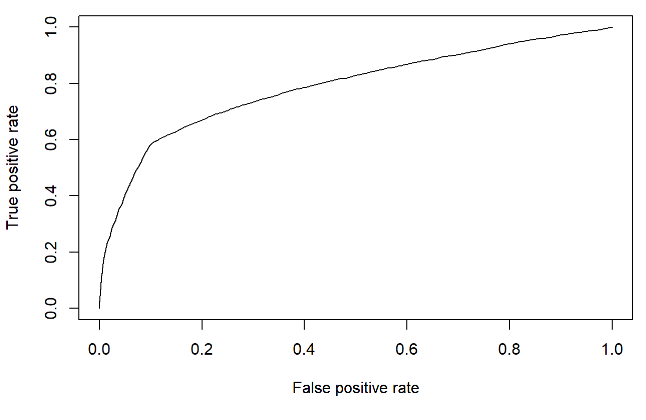
Question #15.Construct an ROC curve and find the AUC for the LDA model. In the rocplot() and auc() functions, use posterior[,2] in the output object obtained in question 13. Due to the collinearity, the posterior probability of some observations cannot be calculated.

```
na.obs=which(is.na(lda.fit$posterior[, 2]))

length(na.obs)
```

```
## [1] 21
```

```
rocplot(lda.fit$posterior[-na.obs, 2], y[-na.obs])
```




```
auc(lda.fit$posterior[-na.obs, 2], y[-na.obs])
```

```
## [1] 0.7879733
```

Question #16.Perform QDA with y as the response and all other variables except duration, default, and loan as predictors. In the lda() function, add CV=T to get cross validation results.

```
qda.fit=qda(y~.-duration-default-loan, data=Bank, CV=T)
```

Question #17.Construct a confusion matrix for the QDA model. Compute the test error rate.

```
na.obs.qda=which(is.na(qda.fit$class))

table(qda.fit$class[-na.obs.qda], y[-na.obs.qda])
```

```
##
##          no   yes
##   no  32942  2102
##   yes  3606  2534
```

```
sum(qda.fit$class[-na.obs.qda]!=y[-na.obs.qda])/nrow(Bank[-na.obs.qda])
```

```
## [1] 0.1385841
```

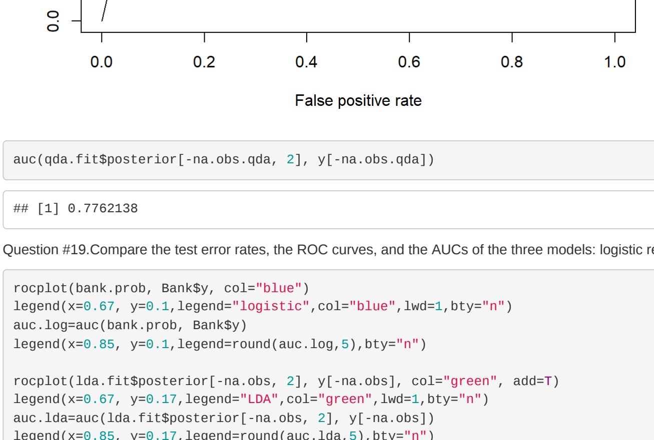
Question #18.Construct an ROC curve and find the AUC for the QDA model. Like the LDA model, some posterior probability are not available and need to be excluded.

```
na.obs.qda=which(is.na(qda.fit$posterior[, 2]))

length(na.obs.qda)
```

```
## [1] 4
```

```
rocplot(qda.fit$posterior[-na.obs.qda, 2], y[-na.obs.qda])
```




```
auc(qda.fit$posterior[-na.obs.qda, 2], y[-na.obs.qda])
```

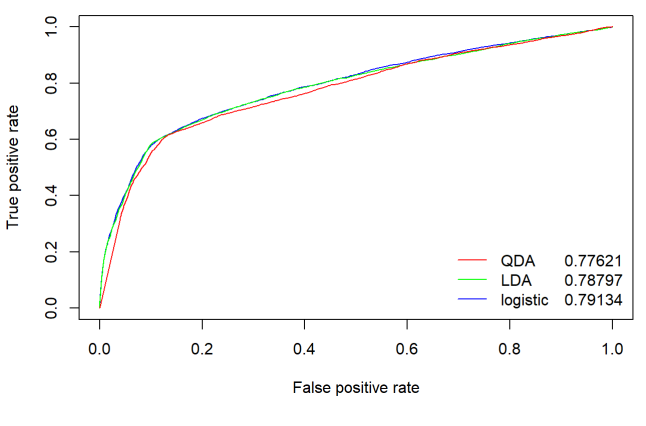
```
## [1] 0.7762138
```

Question #19.Compare the test error rates, the ROC curves, and the AUCs of the three models: logistic regression, LDA and QDA.

```
rocplot(bank.prob, Bank$y, col="blue")
legend(x=0.67, y=0.1,legend="logistic",col="blue",lwd=1,bty="n")
auc.log=auc(bank.prob, Bank$y)
legend(x=0.85, y=0.1,legend=round(auc.log,5),bty="n")

rocplot(lda.fit$posterior[-na.obs, 2], y[-na.obs], col="green", add=T)
legend(x=0.67, y=0.17,legend="LDA",col="green",lwd=1,bty="n")
auc.lda=auc(lda.fit$posterior[-na.obs, 2], y[-na.obs])
legend(x=0.85, y=0.17,legend=round(auc.lda,5),bty="n")

rocplot(qda.fit$posterior[-na.obs.qda, 2], y[-na.obs.qda],col="red", add=T)
legend(x=0.67, y=0.24,legend="QDA",col="red",lwd=1,bty="n")
auc.qda=auc(qda.fit$posterior[-na.obs.qda, 2], y[-na.obs.qda])
legend(x=0.85, y=0.24,legend=round(auc.qda,5),bty="n")
```



#Comparing the three models logistic model was better than LDA and QDA because it has a bigger AUC.