ds2_hw3

Ruihan Zhang

2023-03-21

```
library(caret)
```

```
## Loading required package: ggplot2

## Loading required package: lattice
```

```
library(knitr)
library(glmnet)
```

```
## Loading required package: Matrix

## Loaded glmnet 4.1-4
```

```
library(mlbench)
library(splines)
library(mgcv)
```

```
## Loading required package: nlme

## This is mgcv 1.8-40. For overview type 'help("mgcv-package")'.
```

```
library(pROC)
```

```
## Type 'citation("pROC")' for a citation.

##
## Attaching package: 'pROC'

## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```
library(MASS)
library(earth)
```

```
## Loading required package: Formula
```

```
## Loading required package: plotmo

## Loading required package: plotrix

## Loading required package: TeachingDemos
```

```r
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --

## v tibble  3.1.8      v dplyr   1.0.10
## v tidyr   1.3.0      v stringr 1.5.0
## v readr   2.1.2      v forcats 0.5.2
## v purrr   1.0.1
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::collapse() masks nlme::collapse()
## x tidyr::expand()   masks Matrix::expand()
## x dplyr::filter()   masks stats::filter()
## x dplyr::lag()      masks stats::lag()
## x purrr::lift()     masks caret::lift()
## x tidyr::pack()     masks Matrix::pack()
## x dplyr::select()   masks MASS::select()
## x tidyr::unpack()   masks Matrix::unpack()
```

```r
library(ggplot2)
library(pdp)
```

```
##
## Attaching package: 'pdp'
##
## The following object is masked from 'package:purrr':
##
##     partial
```

```r
library(vip)
```

```
##
## Attaching package: 'vip'
##
## The following object is masked from 'package:utils':
##
##     vi
```

```r
library(klaR)
```

```
##
## Attaching package: 'klaR'
##
## The following object is masked from 'package:TeachingDemos':
##
##     triplot
```

```r
library(AppliedPredictiveModeling)
```

```r
auto=read_csv("./auto.csv") %>%
  janitor::clean_names() %>%
  na.omit() %>%
  mutate(mpg_cat = as.factor(mpg_cat),
         year = as.factor(year),
         origin = as.factor(origin),
         mpg_cat = fct_relevel(mpg_cat, c("low", "high")))
```

```
## Rows: 392 Columns: 8
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (1): mpg_cat
## dbl (7): cylinders, displacement, horsepower, weight, acceleration, year, or...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
set.seed(2023)
train_index=createDataPartition(y=auto$mpg_cat, p=0.7, list = FALSE)
train=auto[train_index,]
test=auto[-train_index,]
#training data
x1=model.matrix(mpg_cat~.,train)[,-1]
y1=train$mpg_cat
#testing data
x2=model.matrix(mpg_cat~.,test)[,-1]
y2=auto$mpg_cat[-train_index]
```

```r
#1.
glm_fit=glm(mpg_cat~., data = auto, subset = train_index, family =binomial(link = "logit"))
summary(glm_fit)
```

```
##
## Call:
## glm(formula = mpg_cat ~ ., family = binomial(link = "logit"),
##     data = auto, subset = train_index)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.0585  -0.0673   0.0028   0.1359   3.4148
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  22.287033   6.589777   3.382 0.000719 ***
## cylinders    -0.873335   0.724940  -1.205 0.228319
## displacement  0.007416   0.021833   0.340 0.734095
## horsepower   -0.027269   0.039025  -0.699 0.484702
## weight       -0.005532   0.002191  -2.524 0.011594 *
## acceleration -0.192443   0.249056  -0.773 0.439706
## year71        0.318705   2.397011   0.133 0.894226
```

```
## year72       -1.206294   1.362235  -0.886 0.375873
## year73       -1.735417   1.452223  -1.195 0.232084
## year74        1.975910   1.809505   1.092 0.274850
## year75        1.167625   1.497279   0.780 0.435490
## year76        2.213028   1.621090   1.365 0.172207
## year77        1.388917   1.823066   0.762 0.446145
## year78        1.544688   1.505540   1.026 0.304890
## year79        4.607034   1.941760   2.373 0.017663 *
## year80        5.194261   2.073119   2.506 0.012227 *
## year81        4.785045   1.801277   2.656 0.007896 **
## year82        3.532248   1.690508   2.089 0.036666 *
## origin2       1.201654   1.163232   1.033 0.301590
## origin3       0.341372   1.139917   0.299 0.764581
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 382.62  on 275  degrees of freedom
## Residual deviance:  78.48  on 256  degrees of freedom
## AIC: 118.48
##
## Number of Fisher Scoring iterations: 8
```

The predictors weight, year79, year80, year81 and year82 are statistically significant, because the p-value of these predictors are smaller than 0.05

```
pred_prob=predict(glm_fit, newdata = auto[-train_index,],
                           type = "response")
pred=rep("low", length(pred_prob))
pred[pred_prob>0.5]="high"
cm=confusionMatrix(data = factor(pred, levels = c("low","high")),
                reference = auto$mpg_cat[-train_index],
                positive = "high")
cm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction low high
##       low   52    5
##       high   6   53
##
##                Accuracy : 0.9052
##                  95% CI : (0.8367, 0.9517)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.8103
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.9138
##             Specificity : 0.8966
```

```
##          Pos Pred Value : 0.8983
##          Neg Pred Value : 0.9123
##              Prevalence : 0.5000
##          Detection Rate : 0.4569
##    Detection Prevalence : 0.5086
##       Balanced Accuracy : 0.9052
##
##        'Positive' Class : high
##
```

```
cm$byClass["Balanced Accuracy"]
```

```
## Balanced Accuracy
##         0.9051724
```

From the confusion matrix, the accuracy can be calculated by (52+53)/(52+53+5+6)=0.90517. The accuracy is close to 1, so the model works well. TN is 52. TP is 53. FN is 5. FP is 6. The 95% CI is (0.8367, 0.9517). The no information rate is 0.5. The p-value is less than 2e-16, which is pretty small, so the null hypothesis is rejected. Therefore, our model is significant. The kappa value is 0.8103, which is close to 1, so the model works well. Both sensitivity and specificity is close to 1.

```
#caret
ctrl=trainControl(method = "cv",
                     summaryFunction = twoClassSummary,
                     classProbs = TRUE)
glm_model=train(x1,y1,,method="glm",metric="ROC", trControl=ctrl)
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
summary(glm_model)
```

```
##
## Call:
## NULL
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.0585  -0.0673   0.0028   0.1359   3.4148
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  22.287033   6.589777   3.382 0.000719 ***
## cylinders    -0.873335   0.724940  -1.205 0.228319
## displacement  0.007416   0.021833   0.340 0.734095
## horsepower   -0.027269   0.039025  -0.699 0.484702
## weight       -0.005532   0.002191  -2.524 0.011594 *
## acceleration -0.192443   0.249056  -0.773 0.439706
## year71        0.318705   2.397011   0.133 0.894226
## year72       -1.206294   1.362235  -0.886 0.375873
## year73       -1.735417   1.452223  -1.195 0.232084
## year74        1.975910   1.809505   1.092 0.274850
## year75        1.167625   1.497279   0.780 0.435490
```

```
## year76        2.213028   1.621090   1.365 0.172207
## year77        1.388917   1.823066   0.762 0.446145
## year78        1.544688   1.505540   1.026 0.304890
## year79        4.607034   1.941760   2.373 0.017663 *
## year80        5.194261   2.073119   2.506 0.012227 *
## year81        4.785045   1.801277   2.656 0.007896 **
## year82        3.532248   1.690508   2.089 0.036666 *
## origin2       1.201654   1.163232   1.033 0.301590
## origin3       0.341372   1.139917   0.299 0.764581
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 382.62  on 275  degrees of freedom
## Residual deviance:  78.48  on 256  degrees of freedom
## AIC: 118.48
##
## Number of Fisher Scoring iterations: 8
```

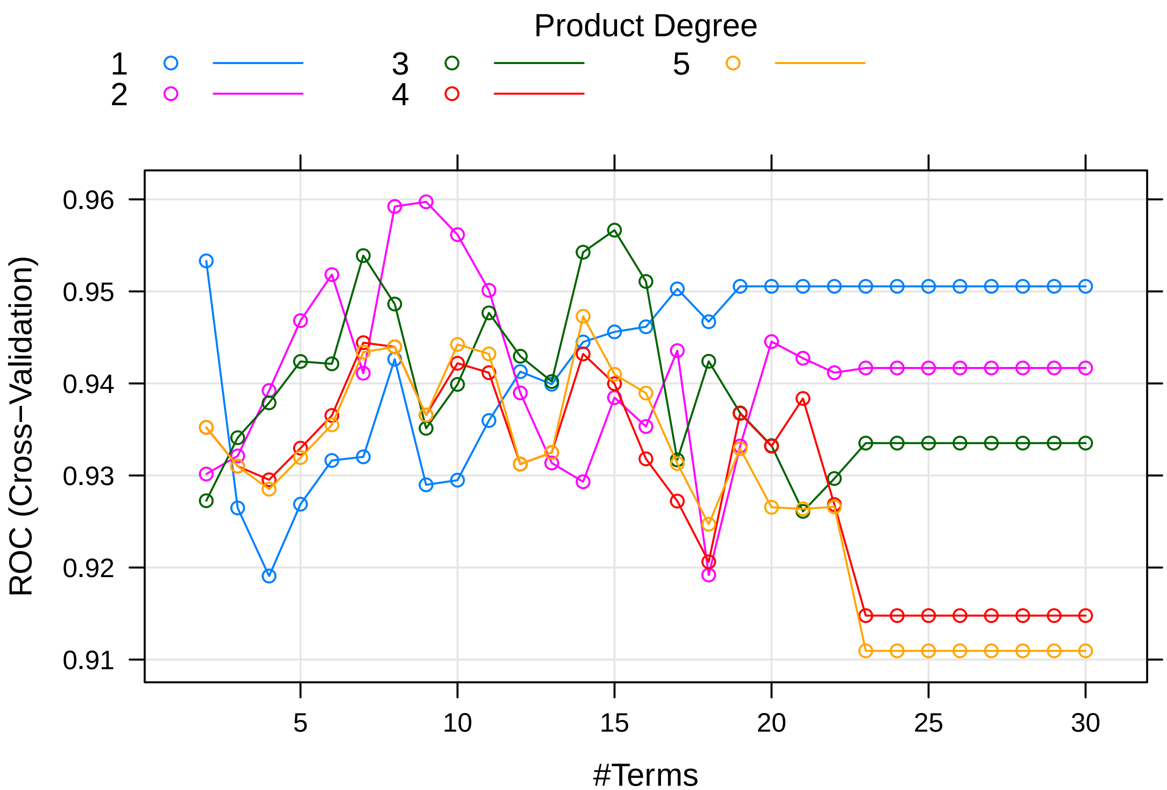
```r
#2.
set.seed(2023)

mars.fit=train(auto[train_index,1:7],
                    auto$mpg_cat[train_index],
                    method = "earth",
                    tuneGrid = expand.grid(degree = 1:5,
                                           nprune = 2:30),
                    metric = "ROC",
                    trControl = ctrl)
summary(mars.fit)
```

```
## Call: earth(x=tbl_df[276,7], y=factor.object, keepxy=TRUE,
##             glm=list(family=function.object, maxit=100), degree=2, nprune=9)
##
## GLM coefficients
##                                                      high
## (Intercept)                                      -4.39379
## h(6-cylinders)                                    4.39334
## h(6-cylinders) * year72                       -1144.02683
## h(250-displacement) * year72                     17.57656
## h(250-displacement) * year73                     -0.02805
## h(4-cylinders) * h(250-displacement)             -0.07958
## h(cylinders-4) * h(250-displacement)              0.02873
## h(250-displacement) * h(weight-2671)             -0.00006
## h(250-displacement) * h(14.3-acceleration)        0.02364
##
## GLM (family binomial, link logit):
##  nulldev  df       dev  df   devratio     AIC iters converged
##  382.617 275   63.2795 267      0.835   81.28    21         1
##
## Earth selected 9 of 33 terms, and 6 of 19 predictors (nprune=9)
## Termination condition: Reached nk 39
## Importance: displacement, cylinders, year72, year73, weight, acceleration, ...
```

```
## Number of terms at each degree of interaction: 1 1 7
## Earth GCV 0.04144919    RSS 9.76534    GRSq 0.8354025    RSq 0.8584733
```

```
plot(mars.fit)
```




```
kable(mars.fit$bestTune,"simple")
```

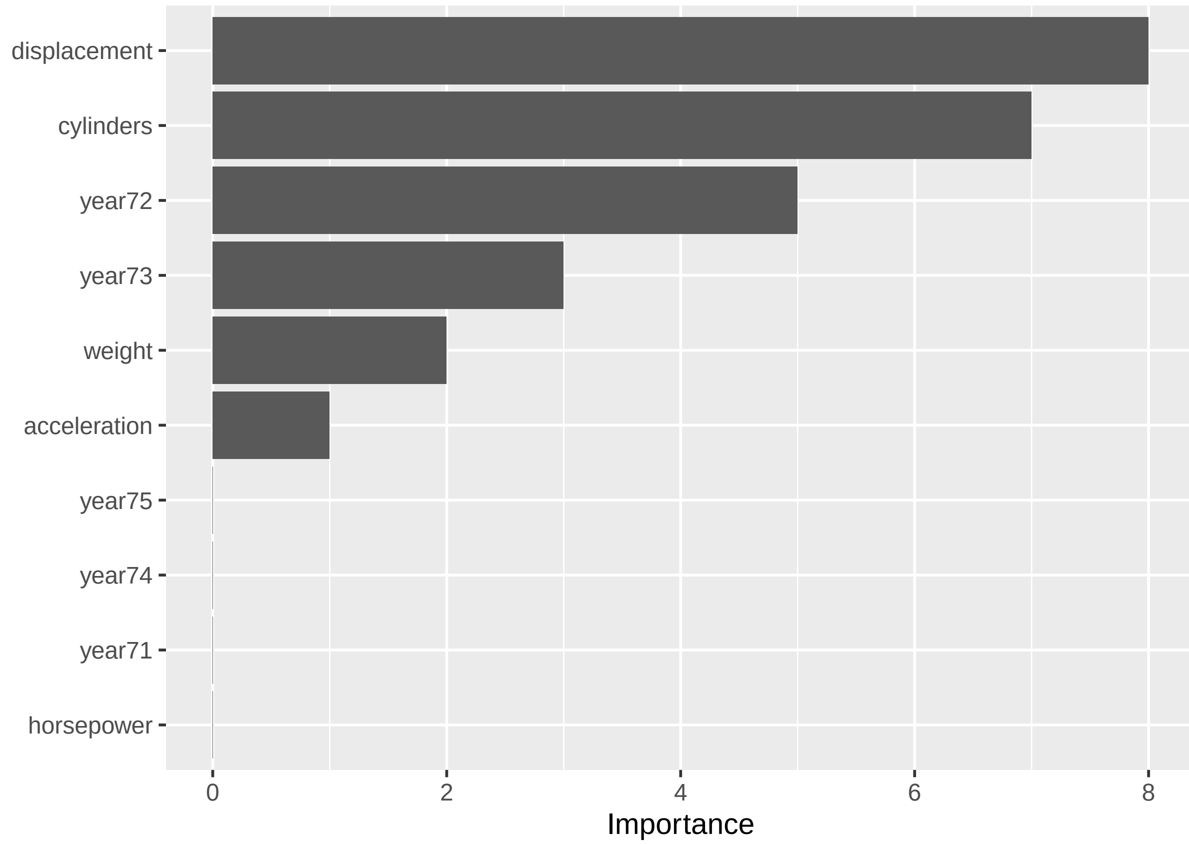
	nprune	degree
37	9	2

```
coef(mars.fit$finalModel)
```

```
##                                 (Intercept)
##                               -4.393787e+00
##       h(cylinders-4) * h(250-displacement)
##                                2.873138e-02
##       h(4-cylinders) * h(250-displacement)
##                               -7.957990e-02
## h(250-displacement) * h(14.3-acceleration)
##                                2.364000e-02
##               h(250-displacement) * year72
##                                1.757656e+01
##               h(250-displacement) * year73
```

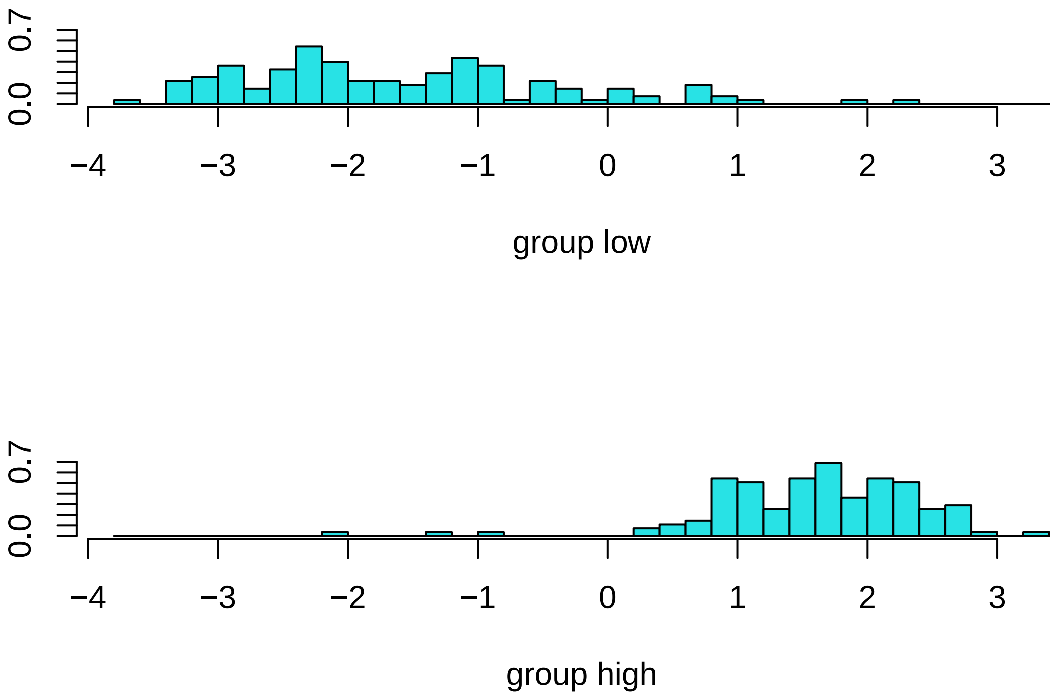
```
##                              -2.805051e-02
##                             h(6-cylinders)
##                               4.393339e+00
##        h(250-displacement) * h(weight-2671)
##                              -6.454739e-05
##                    h(6-cylinders) * year72
##                              -1.144027e+03
```

```
vip(mars.fit$finalModel)
```



When the degree is 2 with 9 terms, the AUC is the highest.

```
#3.
set.seed(2023)
lda_fit=lda(mpg_cat~., data=auto, subset=train_index)
plot(lda_fit)
```




```
lda_fit$scaling
```

```
##                        LD1
## cylinders    -0.6330461045
## displacement -0.0006067008
## horsepower    0.0101564649
## weight       -0.0009549525
## acceleration -0.0635785096
## year71        0.4736405821
## year72        0.0179567748
## year73       -0.0758425539
## year74        0.9473677997
## year75        0.2649570035
## year76        0.6429583548
## year77        0.7169187913
## year78        0.2963530158
## year79        1.2533575787
## year80        1.5369245404
## year81        1.6226917506
## year82        1.5570246564
## origin2       0.4476293185
## origin3       0.2628033617
```

```
lda.pred=predict(lda_fit,newdata = test)
head(lda.pred$posterior)
```

```
##         low         high
## 1 0.9996787 0.0003212537
## 2 0.9994263 0.0005736901
## 3 0.9997312 0.0002688218
## 4 0.9993203 0.0006796832
## 5 0.9987197 0.0012803443
## 6 0.9996102 0.0003897596
```

```
#use caret
set.seed(2023)
lda_fit2=train(mpg_cat~., data=train,method="lda",metric="ROC",trControl=ctrl)
lda_fit2$results
```

```
##   parameter       ROC      Sens      Spec      ROCSD    SensSD     SpecSD
## 1      none 0.9657771 0.8763736 0.9703297 0.05623982 0.1221558 0.03834799
```

```
coef(lda_fit2$finalModel)
```

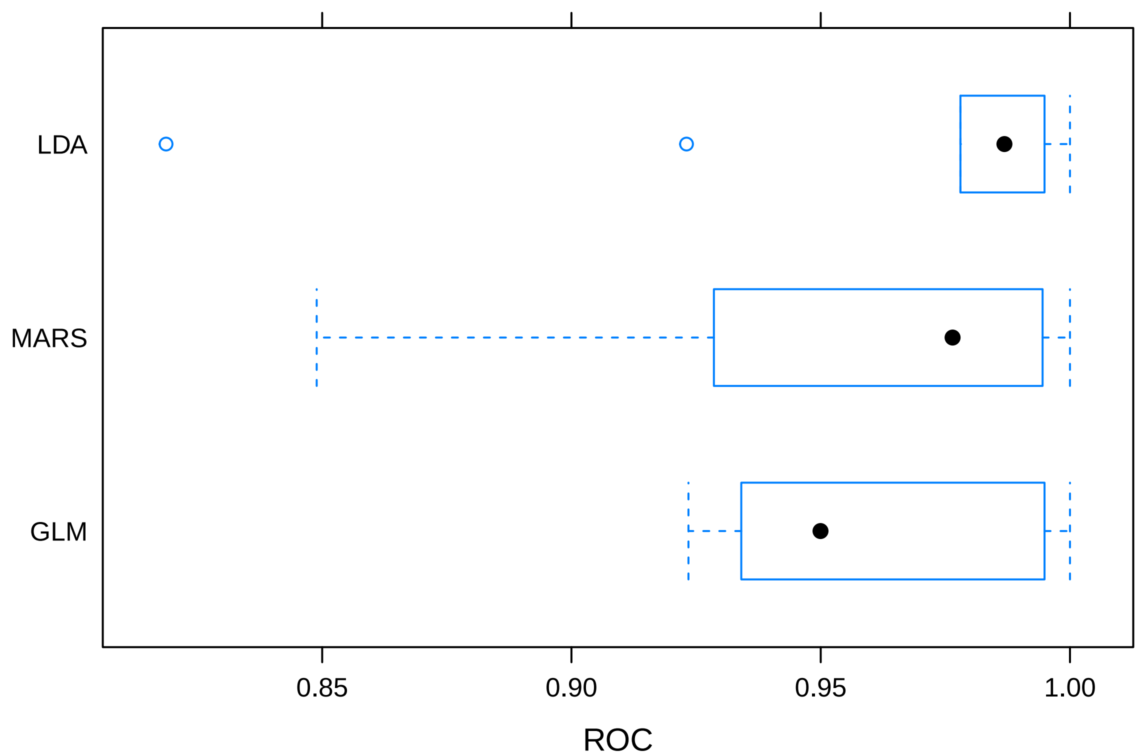
```
##                        LD1
## cylinders    -0.6330461045
## displacement -0.0006067008
## horsepower    0.0101564649
## weight       -0.0009549525
## acceleration -0.0635785096
## year71        0.4736405821
## year72        0.0179567748
## year73       -0.0758425539
## year74        0.9473677997
## year75        0.2649570035
## year76        0.6429583548
## year77        0.7169187913
## year78        0.2963530158
## year79        1.2533575787
## year80        1.5369245404
## year81        1.6226917506
## year82        1.5570246564
## origin2       0.4476293185
## origin3       0.2628033617
```

```
#4.
set.seed(2023)
res=resamples(list(GLM=glm_model,MARS=mars.fit,LDA=lda_fit2))
summary(res)
```

```
##
## Call:
## summary.resamples(object = res)
##
## Models: GLM, MARS, LDA
## Number of resamples: 10
##
## ROC
```

```
##            Min.   1st Qu.    Median      Mean   3rd Qu. Max. NA's
## GLM  0.9234694 0.9365188 0.9499608 0.9614207 0.9947998    1    0
## MARS 0.8489011 0.9362245 0.9764521 0.9597331 0.9933281    1    0
## LDA  0.8186813 0.9784144 0.9868524 0.9657771 0.9936224    1    0
##
## Sens
##            Min.   1st Qu.    Median      Mean   3rd Qu. Max. NA's
## GLM  0.7142857 0.8571429 0.8901099 0.8851648 0.9285714    1    0
## MARS 0.8461538 0.8736264 0.9285714 0.9340659 1.0000000    1    0
## LDA  0.6428571 0.8461538 0.8928571 0.8763736 0.9821429    1    0
##
## Spec
##            Min.   1st Qu.    Median      Mean   3rd Qu. Max. NA's
## GLM  0.7857143 0.9244505 0.9285714 0.9351648 1.0000000    1    0
## MARS 0.7857143 0.8571429 0.9258242 0.9126374 0.9821429    1    0
## LDA  0.9230769 0.9285714 1.0000000 0.9703297 1.0000000    1    0
```

```
bwplot(res,metric = "ROC")
```



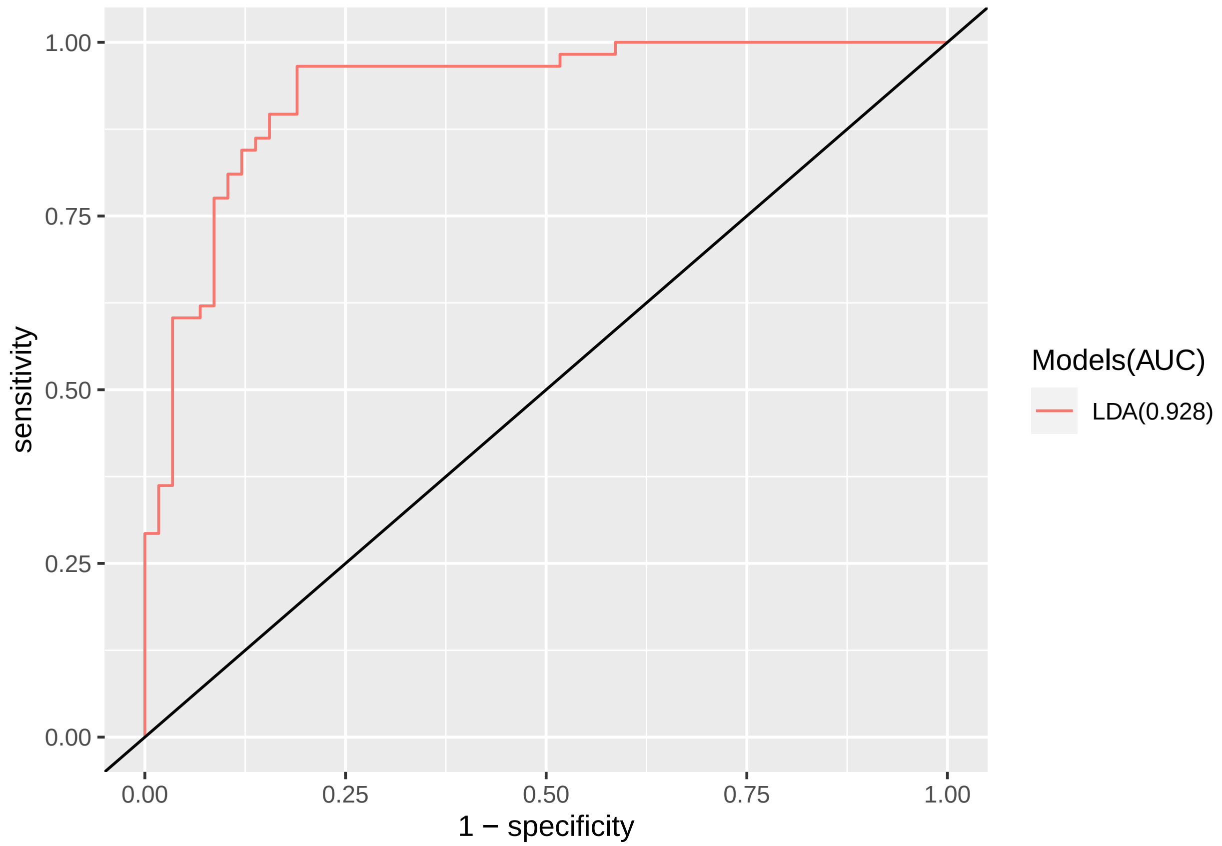
Based on the graph, the LDA model has the highest AUC. Thus, I choose the LDA model to predict the response.

```
lda_pred=predict(lda_fit2, newdata = test,type = "prob")[,2]
lda_roc=roc(test$mpg_cat, lda_pred)
```

```
## Setting levels: control = low, case = high
```

```
## Setting direction: controls < cases
```

```
lda_auc=lda_roc$auc[1]
modelName=c("LDA")
ggroc(list(lda_roc), legacy.axes = TRUE) + scale_color_discrete(labels=paste0(modelName,"(", round(lda_a
```



The AUC is 0.928.

```
test_pred_lda=predict(lda_fit2,newdata = test,type = "prob")
test_pred_prob=predict(glm_fit,newdata = test, type = "response")
pred2=rep("low",length(lda_pred))
pred2[lda_pred>0.5]="high"
confusionMatrix(data = as.factor(pred2),reference=test$mpg_cat,positive="high")
```

```
## Warning in confusionMatrix.default(data = as.factor(pred2), reference =
## test$mpg_cat, : Levels are not in the same order for reference and data.
## Refactoring data to match.
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction low high
##       low   46    2
##       high  12   56
##
```

```
##                Accuracy : 0.8793
##                  95% CI : (0.8058, 0.9324)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.7586
##
##  Mcnemar's Test P-Value : 0.01616
##
##             Sensitivity : 0.9655
##             Specificity : 0.7931
##          Pos Pred Value : 0.8235
##          Neg Pred Value : 0.9583
##              Prevalence : 0.5000
##          Detection Rate : 0.4828
##    Detection Prevalence : 0.5862
##       Balanced Accuracy : 0.8793
##
##        'Positive' Class : high
##
```

The misclassification rate of the LDA model is 1-0.8793=0.1207.