P8106 Data ScienceII Homework3

Yueran Zhang (yz4188)

2023-03-21

In this problem, you will develop a model to predict whether a given car gets high or low gas mileage based on the dataset "auto.csv". The dataset contains 392 observations. The response variable is mpg cat, which indicates whether the miles per gallon of a car is high or low.

Split the dataset into two parts: training data (70%) and test data (30%).

R Package

```
library(tidyverse)
library(knitr)
library(AppliedPredictiveModeling)
library(pROC)
library(caret)
library(klaR)
library(MASS)
```

Import Dataset

```
set.seed(1234)

# Load dataset + clean data
auto = read.csv("/Users/yueranzhang/Desktop/DSII/HW3/DataSet/auto.csv") %>%
janitor::clean_names() %>%
  na.omit() %>%
  distinct() %>%
  mutate(
    cylinders = as.factor(cylinders),
    year = as.factor(year),
    origin = case_when(origin == "1" ~ "American",
                       origin == "2" ~ "European",
                       origin == "3" ~ "Japanese"),
    origin = as.factor(origin),
    mpg_cat = as.factor(mpg_cat),
    mpg_cat = fct_relevel(mpg_cat, "low")
  ) %>%
  as.data.frame()
```

```
# Data Partition
rowTrain <- createDataPartition(y = auto$mpg_cat,
                                p = 0.7,
                                list = FALSE)

# training data
x <- model.matrix(mpg_cat~.,auto)[rowTrain]
y <- auto$mpg_cat[rowTrain]
# test data
x2 <- model.matrix(mpg_cat~.,auto)[-rowTrain]
y2 <- auto$mpg_cat[-rowTrain]
```

Question A

Perform a logistic regression using the training data. Do any of the predictors appear to be statistically significant? If so, which ones? Set a probability threshold to determine class labels and compute the confusion matrix using the test data. Briefly explain what the confusion matrix is telling you.

```
set.seed(1234)
contrasts(auto$mpg_cat)
```

```
##      high
## low     0
## high    1
```

```
glm.fit <- glm(mpg_cat ~ .,
               data = auto,
               subset = rowTrain,
               family = binomial(link = "logit"))

# Check for statistically significant predictors
summary(glm.fit)
```

```
##
## Call:
## glm(formula = mpg_cat ~ ., family = binomial(link = "logit"),
##     data = auto, subset = rowTrain)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.59833  -0.06223   0.00012   0.06660   3.02804
##
## Coefficients:
##                  Estimate Std. Error z value Pr(>|z|)
## (Intercept)     1.887e+01  7.526e+00   2.507   0.0122 *
## cylinders4      6.281e+00  2.820e+00   2.227   0.0259 *
## cylinders5      2.042e+01  1.455e+03   0.014   0.9888
## cylinders6      2.394e+00  3.118e+00   0.768   0.4427
```

```
## cylinders8      6.252e+00  4.179e+00   1.496   0.1347
## displacement    2.293e-02  2.277e-02   1.007   0.3138
## horsepower     -6.481e-02  4.548e-02  -1.425   0.1541
## weight         -6.795e-03  2.367e-03  -2.870   0.0041 **
## acceleration   -2.939e-01  2.548e-01  -1.153   0.2488
## year71         -2.452e+00  2.288e+00  -1.072   0.2839
## year72         -1.489e+00  2.015e+00  -0.739   0.4597
## year73          2.129e-01  2.265e+00   0.094   0.9251
## year74          2.116e+00  2.793e+00   0.758   0.4487
## year75          8.181e-01  2.035e+00   0.402   0.6877
## year76          5.917e-01  2.276e+00   0.260   0.7949
## year77          3.950e+00  3.446e+00   1.146   0.2518
## year78          1.532e+00  2.198e+00   0.697   0.4860
## year79          3.909e+00  2.254e+00   1.734   0.0829 .
## year80          7.043e+00  2.825e+00   2.493   0.0127 *
## year81          5.671e+00  2.491e+00   2.277   0.0228 *
## year82          4.555e+00  2.188e+00   2.081   0.0374 *
## originEuropean  3.053e+00  1.556e+00   1.963   0.0497 *
## originJapanese  1.180e+00  1.282e+00   0.921   0.3573
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 382.617  on 275  degrees of freedom
## Residual deviance:  63.071  on 253  degrees of freedom
## AIC: 109.07
##
## Number of Fisher Scoring iterations: 14
```

- After performing a glm model using the training data, the variables `weight(Vehicle weight)`, `year79(Model year 79)`, `year80`, `year81`, `year82` and `origin2(European origin)` appear to be statistically significant.

```
# By default, we set this classification threshold to 0.5.


test.pred.prob = predict(glm.fit, newdata = auto[-rowTrain,],
                         type = "response")
test.pred = rep("low", length(test.pred.prob))
test.pred[test.pred.prob > 0.5] = "high"

confusionMatrix(data = fct_rev(as.factor(test.pred)), # Reverse order of factor levels:)
                reference = auto$mpg_cat[-rowTrain],
                 positive = "high")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction low high
##       low   49    5
##       high   9   53
##
```

```
##                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 : 0.4227
##
##             Sensitivity : 0.9138
##             Specificity : 0.8448
##          Pos Pred Value : 0.8548
##          Neg Pred Value : 0.9074
##              Prevalence : 0.5000
##          Detection Rate : 0.4569
##    Detection Prevalence : 0.5345
##       Balanced Accuracy : 0.8793
##
##        'Positive' Class : high
##
```

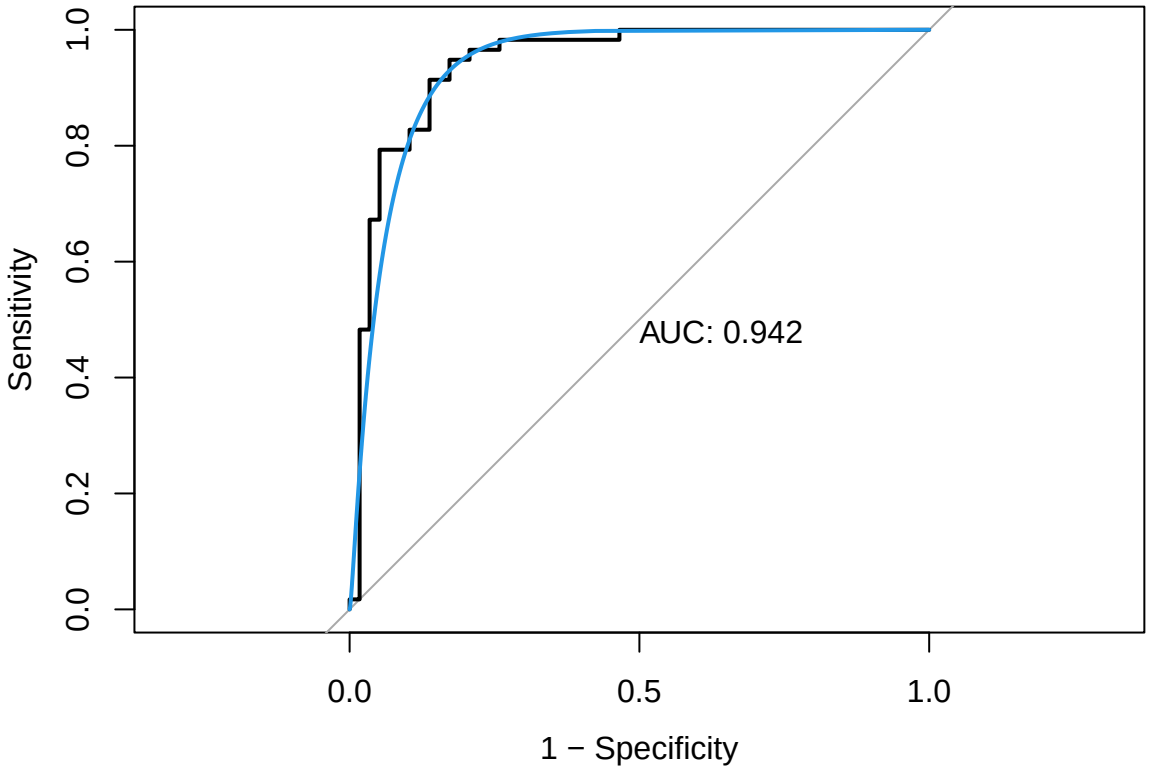
```
## Double check for the order
# fct_rev(levels(as.factor(test.pred)))
# levels(auto$mpg_cat[-rowTrain])
```

```
set.seed(1234)
roc.glm <- roc(auto$mpg_cat[-rowTrain], test.pred.prob)
```

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

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

```
plot(roc.glm, legacy.axes = TRUE, print.auc = TRUE)
plot(smooth(roc.glm), col = 4, add = TRUE)
```



- In the confusion matrix, the class of interest/our target class is high gas mileage (positive) class.
- The diagonal elements are the correctly predicted samples. A total of `52+53=105` samples were correctly predicted out of the total `52+5+6+53=116` samples. The accuracy score reads as `0.9052(90.52%)` 95%CI `(0.8058, 0.9324)` for the given data and observations. If a model will perform at 90.52% accuracy then the error rate will be `1-0.905 =9.5%`. Other important indicators, like specificity(0.8966) and sensitivity(0.9138).
- The confusion matrix also tells us that our no information rate is 50%, which means that if we had no information and made the same class prediction for all observations, our model would be 50% accurate.
- Our p-value is <2e-16 (close to 0) tells us that our accuracy is statistically significantly better than our no information rate.
- The model' sensitivity is 91.38% (true detected positives out of all actual positives);Specificity is 84.48% (true detected negatives out of all actual negatives);positive predictive value is 85.48% (true detected positives out of all predicted positives); negative predictive value of 90.74% (true detected negatives out of all predicted negatives). Our sensitivity and specificity average is 87.93%, which is our balanced accuracy. Our kappa is 0.7586, indicates pretty good reliability.
- Another important metric that measures the overall performance of a classifier is the "Area Under ROC" (AUC) value.We also made this plot and observe the area measured under the ROC curve. A higher value of AUC represents a better classifier. The AUC of the practical learner above is 90% (from the plot we know AUC=0.957) which is a pretty good score.

Question B

Train a multivariate adaptive regression spline (MARS) model using the training data.

```
set.seed(1234)

ctrl <- trainControl(method = "repeatedcv",
                     repeats = 5,
                     summaryFunction = twoClassSummary,
                     classProbs = TRUE)

model.mars <- train(x = auto[rowTrain,1:7],
                    y = auto$mpg_cat[rowTrain],
                    method = "earth",
                    tuneGrid = expand.grid(degree = 1:3,
                                           nprune = 3:25),
                    metric = "ROC",
                    trControl = ctrl)
```

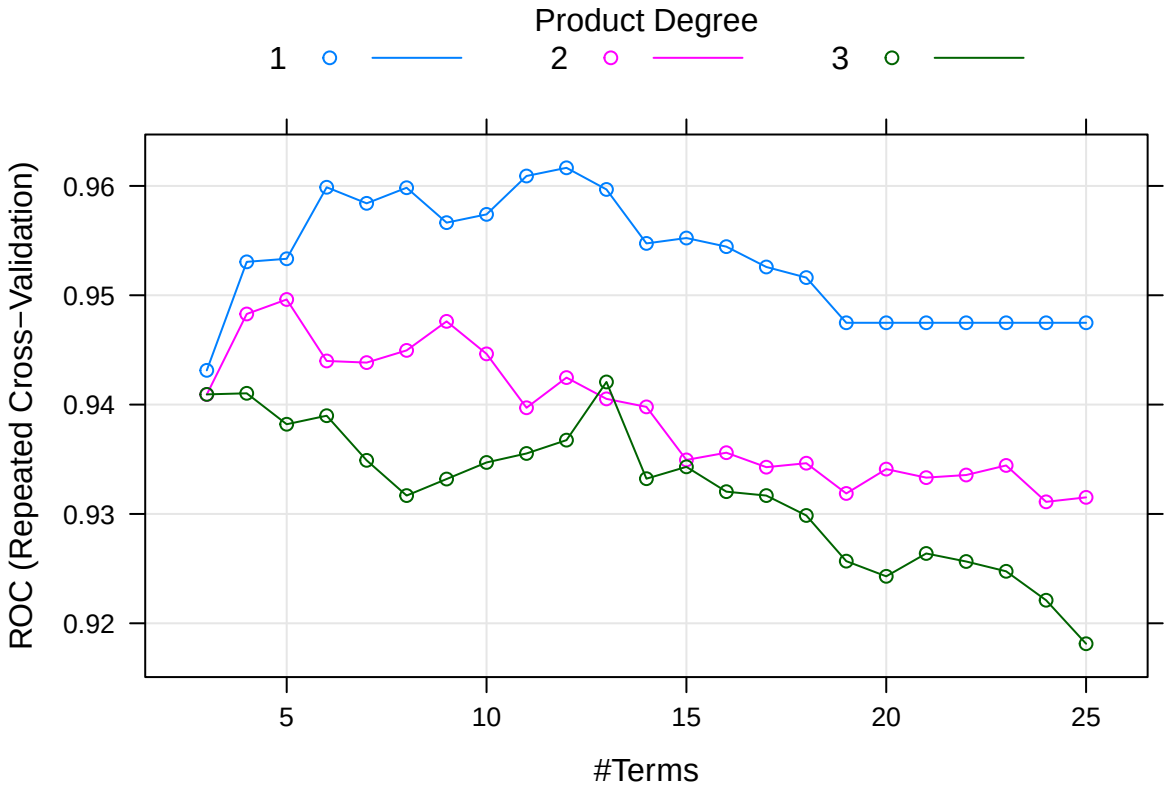
```
## Loading required package: earth

## Loading required package: Formula

## Loading required package: plotmo

## Loading required package: plotrix

## Loading required package: TeachingDemos

##
## Attaching package: 'TeachingDemos'

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

```
summary(model.mars)
```

```
## Call: earth(x=data.frame[276,7], y=factor.object, keepxy=TRUE,
##             glm=list(family=function.object, maxit=100), degree=1, nprune=12)
##
## GLM coefficients
##                             high
## (Intercept)           0.9660326
## cylinders4            3.1362638
## cylinders5           16.6870040
## h(displacement-140)   2.4297056
## h(displacement-145) -3.4791018
## h(displacement-156)   1.3606990
## h(displacement-171) -0.5337751
```

```
## h(displacement-200)  0.2709808
## h(horsepower-80)    -0.2238246
## h(horsepower-88)     0.1630662
## h(weight-2904)      -0.0732373
## h(weight-2950)       0.0709560
##
## GLM (family binomial, link logit):
##  nulldev  df       dev  df   devratio     AIC iters converged
##  382.617 275   77.6346 264      0.797   101.6    14         1
##
## Earth selected 12 of 26 terms, and 5 of 22 predictors (nprune=12)
## Termination condition: Reached nk 45
## Importance: cylinders4, displacement, weight, horsepower, cylinders5, ...
## Number of terms at each degree of interaction: 1 11 (additive model)
## Earth GCV 0.05106947    RSS 11.84386    GRSq 0.7971997    RSq 0.8283498
```

```
plot(model.mars)
```




```
model.mars$bestTune
```

```
##    nprune degree
## 10     12      1
```

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

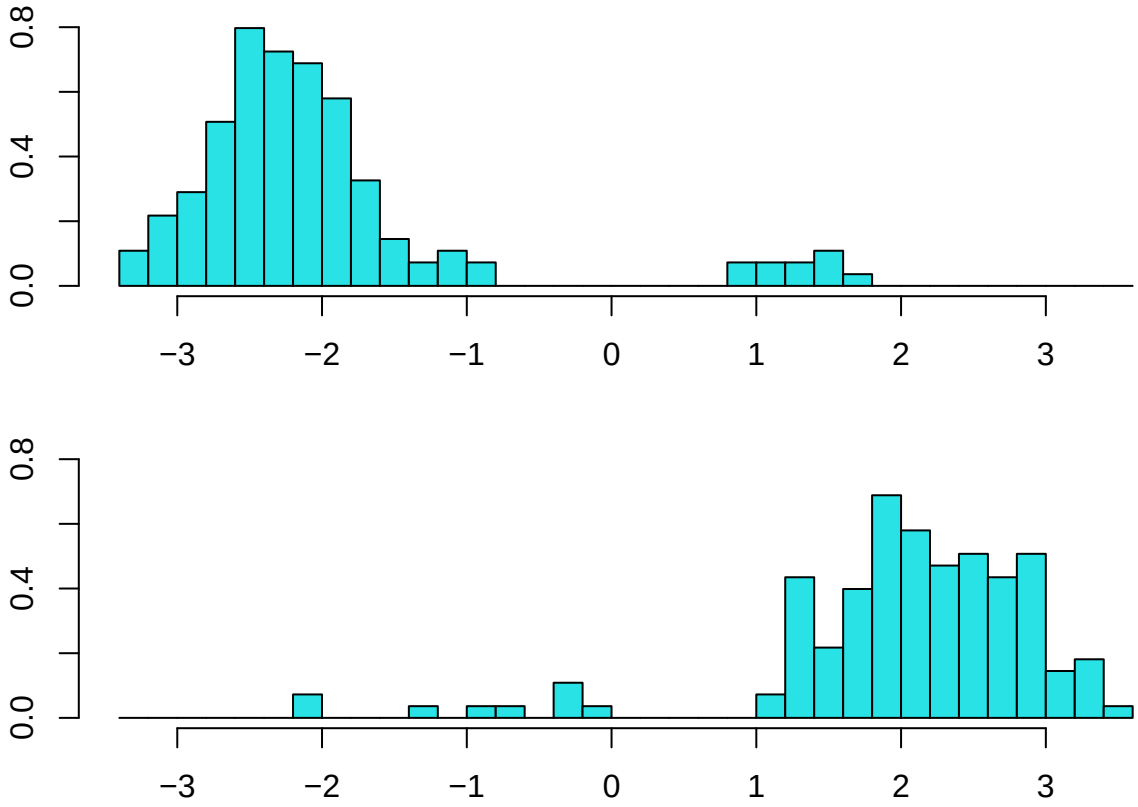
```
##         (Intercept)           cylinders4           cylinders5 h(displacement-171)
##           0.96603264           3.13626376          16.68700402          -0.53377506
## h(displacement-140) h(displacement-200) h(displacement-156) h(displacement-145)
##           2.42970564           0.27098082           1.36069898          -3.47910178
##    h(horsepower-88)     h(horsepower-80)      h(weight-2904)      h(weight-2950)
##           0.16306621          -0.22382459          -0.07323725           0.07095604
```

- For MARS model, Earth selected 12 of 26 terms, and 5 of 22 predictors (nprune=12). R-Squared(RSq) is `0.8283498`, indicates the test set with relatively higher variation in the reponse.

Question C

Perform LDA using the training data. Plot the linear discriminants in LDA.

```
set.seed(1234)
par(mar=c(1,1,1,1))
lda.fit <- lda(mpg_cat~., data = auto,
               subset = rowTrain)

par(mar = rep(2,4))

plot(lda.fit)
```




```
lda.fit$scaling
```

```
##                         LD1
## cylinders4      3.1841925195
## cylinders5      4.5213982667
## cylinders6      0.2845088170
## cylinders8      0.7905775470
## displacement   -0.0029436899
## horsepower      0.0017745854
## weight         -0.0006928485
## acceleration   -0.0538783115
## year71         -0.0907639974
## year72         -0.3717454291
## year73         -0.0537372994
## year74          0.4271431804
## year75          0.1086777417
## year76         -0.1013706688
## year77          0.3772175109
## year78          0.0637981477
## year79          0.5404322644
## year80          1.1520499182
## year81          1.2508852878
## year82          1.0532236521
## originEuropean  0.3189965647
## originJapanese  0.0465462651
```

- There are two classes in the LDA model, so we can plot 1 (`k = 2-1 = 1`) linear discriminant.

```
# Alternatively, let's use caret for LDA

set.seed(1234)

traindf <- auto[rowTrain,]

ctrl <- trainControl(method = "repeatedcv", repeats = 5,
                     summaryFunction = twoClassSummary,
                     classProbs = TRUE)

model.lda <- train(mpg_cat ~ .,
                   data = traindf,
                   method = "lda",
                   metric = "ROC",
                   trControl = ctrl)
model.lda$results
```

```
##   parameter       ROC      Sens      Spec      ROCSD   SensSD     SpecSD
## 1      none 0.9730246 0.9278388 0.9340659 0.02664249 0.059635 0.05500746
```

Question D

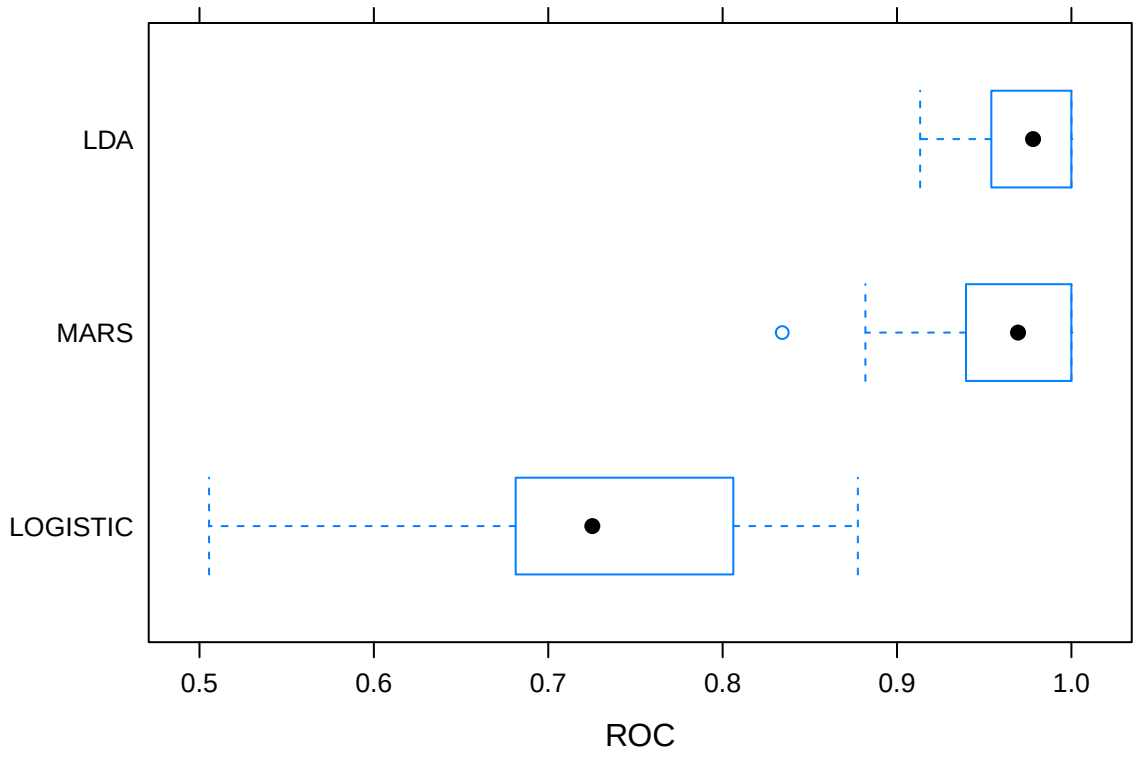
Which model will you use to predict the response variable? Plot its ROC curve using the test data. Report the AUC and the misclassification error rate.

```
set.seed(1234)
logit.caret = train(x = rowTrain,
                    y = y,
                    method = "glm",
                    metric = "ROC",
                    trControl = ctrl)

res = resamples(list(LOGISTIC = logit.caret,
                     MARS = model.mars,
                     LDA = model.lda))

summary(res)
```

```
##
## Call:
## summary.resamples(object = res)
##
## Models: LOGISTIC, MARS, LDA
## Number of resamples: 50
##
## ROC
##                Min.   1st Qu.    Median      Mean   3rd Qu.     Max. NA's
## LOGISTIC 0.5054945 0.6826923 0.7252747 0.7295918 0.8048469 0.877551    0
```

```
## MARS     0.8341837 0.9400020 0.9693878 0.9616562 0.9987245 1.000000    0
## LDA      0.9132653 0.9540816 0.9780220 0.9730246 1.0000000 1.000000    5
##
## Sens
##                Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## LOGISTIC 0.3571429 0.5714286 0.6428571 0.6379121 0.7142857 0.8461538    0
## MARS     0.7142857 0.8571429 0.9258242 0.9087912 0.9285714 1.0000000    0
## LDA      0.7857143 0.9230769 0.9285714 0.9278388 1.0000000 1.0000000    5
##
## Spec
##                Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## LOGISTIC 0.3846154 0.5714286 0.6428571 0.6360440 0.7142857 0.8571429    0
## MARS     0.6428571 0.8571429 0.9285714 0.9265934 1.0000000 1.0000000    0
## LDA      0.8461538 0.9230769 0.9285714 0.9340659 1.0000000 1.0000000    5
```

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



- Based on resampling from how our models perform on the training data, I prefer to use the LDA model for classification of our response variable `mpg_cat`, since it has the highest ROC.

```
#  ROC and prediction
lda.predict = predict(model.lda, newdata = auto[-rowTrain, 1:7], type = "prob")[,2]
roc.lda = roc(auto$mpg_cat[-rowTrain], lda.predict)
```

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

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

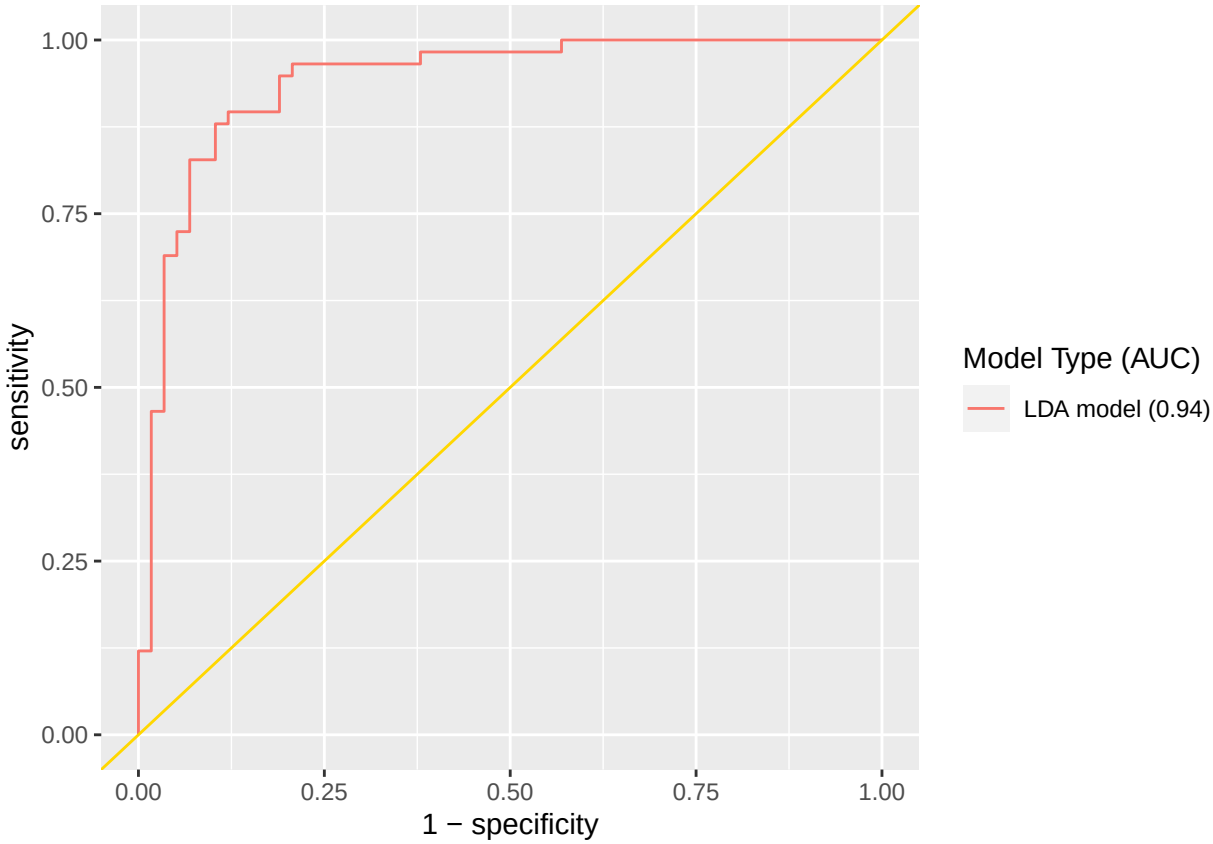
```
#  AUC and misclassification rate
auc_lda = roc.lda$auc[1]
auc_lda
```

```
## [1] 0.9402497
```

```
# Obtain the classes
lda_class = lda.predict %>%
  as.data.frame() %>%
  mutate(
    class = case_when(. < 0.50 ~ "low",
                      . > 0.50 ~ "high")
  ) %>%
  dplyr::select(class) %>%
  as.matrix()

# Confusion matrix and misclassification error rate
confusionMatrix(data = fct_rev(as.factor(lda_class)),
                reference = auto$mpg_cat[-rowTrain],
                positive = "high")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction low high
##       low   47    6
##       high  11   52
##
##                Accuracy : 0.8534
##                  95% CI : (0.7758, 0.9122)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : 1.478e-15
##
##                   Kappa : 0.7069
##
##  Mcnemar's Test P-Value : 0.332
##
##             Sensitivity : 0.8966
##             Specificity : 0.8103
##          Pos Pred Value : 0.8254
##          Neg Pred Value : 0.8868
##              Prevalence : 0.5000
##          Detection Rate : 0.4483
##    Detection Prevalence : 0.5431
##       Balanced Accuracy : 0.8534
##
##        'Positive' Class : high
##
```

```r
# Plot ROC Curve
modelName = "LDA model"
pROC::ggroc(list(roc.lda), legacy.axes = TRUE) +
  scale_color_discrete(labels = paste0(modelName, " (", round(auc_lda, 2),")"),
                       name = "Model Type (AUC)") +
  geom_abline(intercept = 0, slope = 1, color = "gold")
```



- The LDA model has a misclassification rate of `1 - 0.8534 = 14.7%` , and when we apply threshold of 0.5 probability, and observe the area measured under the ROC curve(AUC). A higher value of AUC represents a better classifier. The AUC of the practical learner above is 90% (from the plot we know AUC=0.94) which is a pretty good score.