P8106_HW3_yh3554

Yi Huang

2023-03-22

Contents

```
library(tidyverse)
library(knitr)
library(caret)
library(GGally)
library(glmnet)
library(MASS) #lda
library(pROC)
```

Data Science II Homework 3

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. The predictors are:

- cylinders: Number of cylinders between 4 and 8
- displacement: Engine displacement (cu. inches)
- horsepower: Engine horsepower
- weight: Vehicle weight (lbs.)
- acceleration: Time to accelerate from 0 to 60 mph (sec.)
- year: Model year (modulo 100)
- origin: Origin of car (1. American, 2. European, 3. Japanese)

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

Dara preprocessing

The "auto.csv" dataset contains 1 binary response variable `mpg_cat`, 1 categorical variable `origin`, and 5 continuous variables `displacement`, `horsepower`, `weight`, `acceleration`, and `year`. There are 392 observations and no missing data.

```
# load data
dat <- read.csv("data/auto.csv") %>% na.omit() %>%
  mutate(
    mpg_cat = factor(mpg_cat, levels = c("low", "high")),
    origin = factor(origin))
head(dat)
```

```
##   cylinders displacement horsepower weight acceleration year origin mpg_cat
## 1         8          307        130   3504         12.0   70      1     low
## 2         8          350        165   3693         11.5   70      1     low
## 3         8          318        150   3436         11.0   70      1     low
## 4         8          304        150   3433         12.0   70      1     low
## 5         8          302        140   3449         10.5   70      1     low
## 6         8          429        198   4341         10.0   70      1     low
```

```
summary(dat)
```

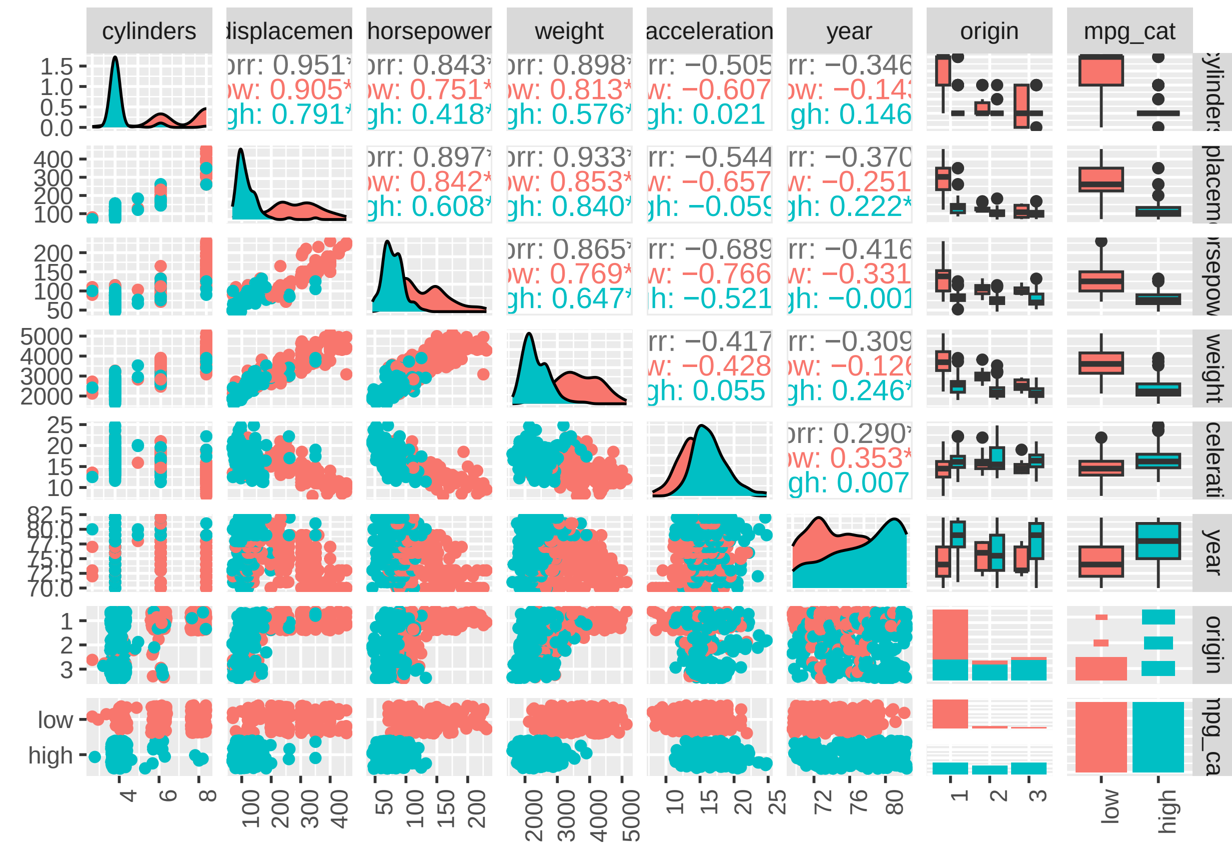
```
##    cylinders      displacement     horsepower        weight      acceleration
##  Min.   :3.000   Min.   : 68.0   Min.   : 46.0   Min.   :1613   Min.   : 8.00
##  1st Qu.:4.000   1st Qu.:105.0   1st Qu.: 75.0   1st Qu.:2225   1st Qu.:13.78
##  Median :4.000   Median :151.0   Median : 93.5   Median :2804   Median :15.50
##  Mean   :5.472   Mean   :194.4   Mean   :104.5   Mean   :2978   Mean   :15.54
##  3rd Qu.:8.000   3rd Qu.:275.8   3rd Qu.:126.0   3rd Qu.:3615   3rd Qu.:17.02
##  Max.   :8.000   Max.   :455.0   Max.   :230.0   Max.   :5140   Max.   :24.80
```

```
##       year         origin  mpg_cat
##  Min.   :70.00   1:245   low :196
##  1st Qu.:73.00   2: 68   high:196
##  Median :76.00   3: 79
##  Mean   :75.98
##  3rd Qu.:79.00
##  Max.   :82.00
```

```
contrasts(dat$mpg_cat)
```

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

```
# correlation plot and boxplot
dat %>% ggpairs(., mapping = ggplot2::aes(colour = mpg_cat), lower = list(combo = 'dot_no_facet')) +
theme(axis.text.x = element_text(angle = 90, hjust = 1))
```




```
# set seed for reproducibility
set.seed(123)

# split the data into training data (70%) and test data (30%)
# specify rows of training data (70% of the dataset)
train_rows <- createDataPartition(dat$mpg_cat,
                                  p = 0.7,
                                  list = F)
dat2 <- model.matrix(mpg_cat~., dat)[,-1]

# training data
```

```r
x <- dat2[train_rows,]
y <- dat$mpg_cat[train_rows]

# test data
x2 <- dat2[-train_rows,]
y2 <- dat$mpg_cat[-train_rows]

# resampling method repeated cross-validation
ctrl <- trainControl(method = "repeatedcv",
                     repeats = 5,
                     summaryFunction = twoClassSummary,
                     classProbs = TRUE)
```

(a) Logistic regression

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.

Perform a logistic regression using the training data. Interpret the results.

```r
# set seed for reproducibility
set.seed(123)

# logistic regression using train data
glm.fit <- glm(mpg_cat ~ .,
               data = dat,
               subset = train_rows,
               family = binomial(link = "logit"))
summary(glm.fit)
```

```
##
## Call:
## glm(formula = mpg_cat ~ ., family = binomial(link = "logit"),
##     data = dat, subset = train_rows)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.1240  -0.1040   0.0054   0.1866   2.9500
##
## Coefficients:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -19.103628   7.290230  -2.620 0.008782 **
## cylinders     -0.302420   0.500722  -0.604 0.545865
## displacement   0.020019   0.015628   1.281 0.200213
## horsepower    -0.039326   0.029745  -1.322 0.186136
## weight        -0.006103   0.001665  -3.666 0.000246 ***
## acceleration   0.073445   0.174583   0.421 0.673981
## year           0.476157   0.103035   4.621 3.81e-06 ***
## origin2        2.266214   0.912275   2.484 0.012987 *
## origin3        1.710709   0.868194   1.970 0.048790 *
## ---
## 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: 114.63  on 267  degrees of freedom
## AIC: 132.63
##
## Number of Fisher Scoring iterations: 8
```

```
# Using caret
model.glm <- train(x,
                   y,
                   method = "glm",
                   metric = "ROC",
                   trControl = ctrl)
```

Based on the summary table, the predictors **weight**, **year**, **origin** are statistically significant since their p-value are relatively smaller than $\alpha = 0.05$.
Interpretation for statistically significant predictors
* β_{weight}: the log odds of high car gas mileage for one lbs increase in vehicle weight is -0.0061
* β_{year}: the log odds of high car gas mileage for one lbs increase in vehicle year is 0.4762
* $\beta_{origin2}$: the log odds ratio of high car gas mileage comparing European model to American model is 2.2662
* $\beta_{origin3}$: the log odds ratio of high car gas mileage comparing Japanese model to American model is 1.7107

Set a probability threshold and compute the confusion matrix using test data.

Set probability threshold the classifier cut-off = 0.5

```
set.seed(123)
test.pred.prob <- predict(glm.fit, newdata = dat[-train_rows,],
                          type = "response")
test.pred <- rep("low", length(test.pred.prob))
test.pred[test.pred.prob>0.5] <- "high"
confusionMatrix(data = as.factor(test.pred),
                reference = dat$mpg_cat[-train_rows],
                positive = "high")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction low high
##       low   54    3
##       high   4   55
##
##                Accuracy : 0.9397
##                  95% CI : (0.8796, 0.9754)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.8793
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.9483
```

```
##             Specificity : 0.9310
##          Pos Pred Value : 0.9322
##          Neg Pred Value : 0.9474
##              Prevalence : 0.5000
##          Detection Rate : 0.4741
##    Detection Prevalence : 0.5086
##       Balanced Accuracy : 0.9397
##
##        'Positive' Class : high
##
```

Briefly explain what the confusion matrix.

Based on the confusion matrix, the correction prediction or accuracy can be calculated as $(54+55)/(54+3+4+55) = 0.9397$. The sensitivity is $55/58 = 0.9483$, and specificity is $54/58 = 0.9310$. The balanced accuracy is the average of sensitivity and specificity that is 0.9397. Since the accuracy is close to 1, the prediction is pretty good.

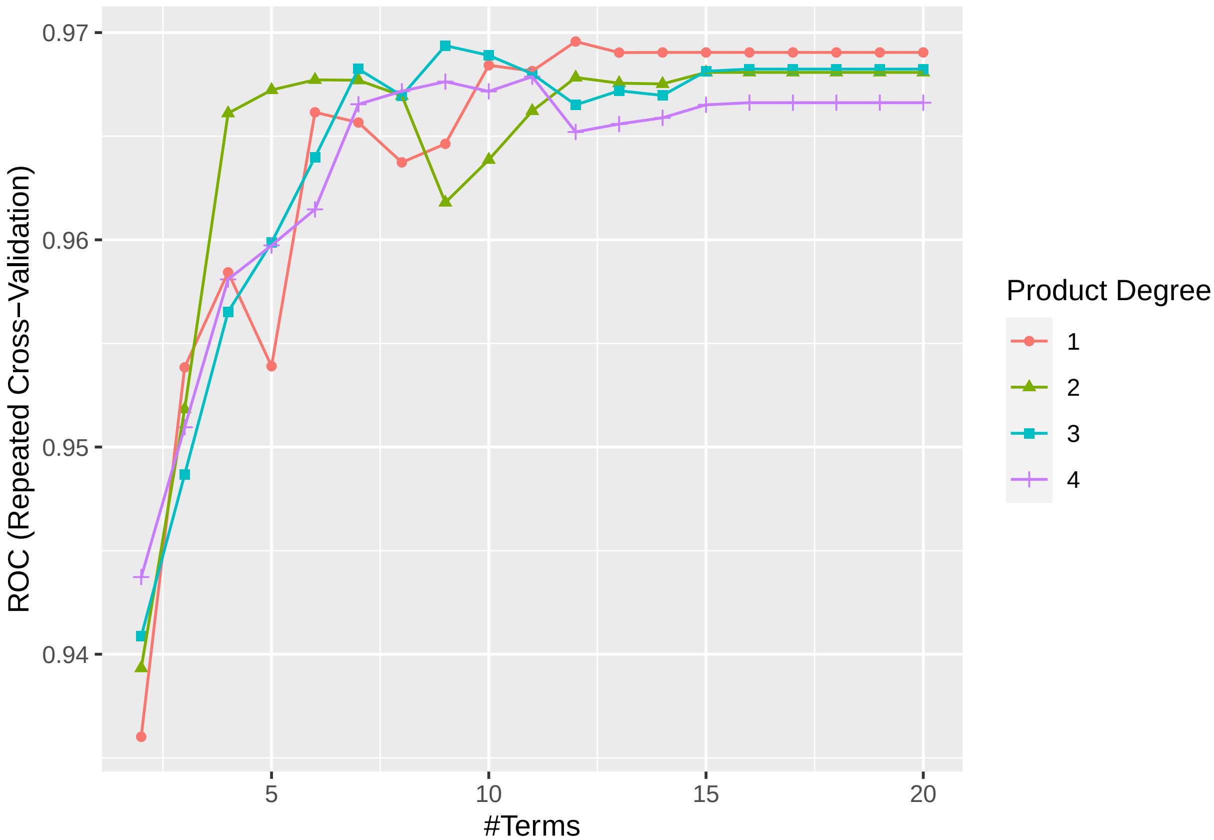
(b) MARS

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

```r
set.seed(123)

model.mars <- train(x, # train x
                    y, # train y
                    method = "earth",
                    tuneGrid = expand.grid(degree = 1:4,
                                           nprune = 2:20),
                    metric = "ROC",
                    trControl = ctrl)

ggplot(model.mars)
```




```
model.mars$bestTune
```

```
##    nprune degree
## 11     12      1
```

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

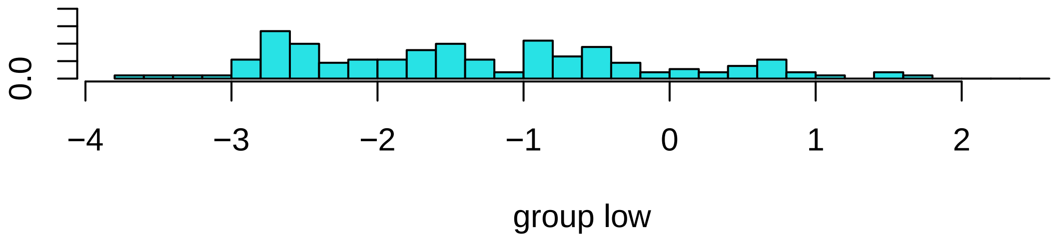
```
##         (Intercept)          h(year-72)          h(72-year)    h(horsepower-81)
##        29.591075644         0.616396423         1.286110312        -0.091736891
##       h(3282-weight) h(displacement-156)       h(cylinders-6)       h(6-cylinders)
##         0.004625826         0.453048820        19.696009066       -16.378606625
##       h(cylinders-4) h(displacement-173) h(displacement-200)
##       -17.622496351        -0.776921866         0.331512444
```

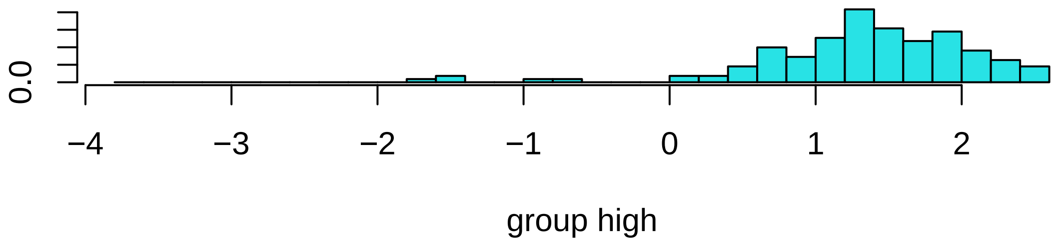
(c) LDA

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

```
set.seed(123)
lda.fit <- lda(mpg_cat~., data = dat,
               subset = train_rows)

plot(lda.fit)
```






```
lda.fit$scaling
```

```
##                       LD1
## cylinders    -0.497878114
## displacement  0.001542615
## horsepower    0.009489176
## weight       -0.001167187
## acceleration  0.018942416
## year          0.122161738
## origin2       0.509913227
## origin3       0.477107038
```

```
# Using caret
model.lda <- train(x, #train x
                   y, #train y
                   method = "lda",
                   metric = "ROC",
                   trControl = ctrl)
summary(model.lda)
```

```
##             Length Class      Mode
## prior        2     -none-     numeric
## counts       2     -none-     numeric
## means       16     -none-     numeric
## scaling      8     -none-     numeric
## lev          2     -none-     character
## svd          1     -none-     numeric
## N            1     -none-     numeric
## call         3     -none-     call
## xNames       8     -none-     character
## problemType  1     -none-     character
## tuneValue    1     data.frame list
## obsLevels    2     -none-     character
```

```
## param        0      -none-     list
```

```
# lda model with continues predictor only because
# lda model does not capture categorical predictors well
model.lda.cont <- train(x[,1:6], #continues train x
                   y, #train y
                   method = "lda",
                   metric = "ROC",
                   trControl = ctrl)
summary(model.lda.cont)
```

```
##             Length Class      Mode
## prior        2     -none-     numeric
## counts       2     -none-     numeric
## means       12     -none-     numeric
## scaling      6     -none-     numeric
## lev          2     -none-     character
## svd          1     -none-     numeric
## N            1     -none-     numeric
## call         3     -none-     call
## xNames       6     -none-     character
## problemType  1     -none-     character
## tuneValue    1     data.frame list
## obsLevels    2     -none-     character
## param        0     -none-     list
```

(d) Comparing models

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.

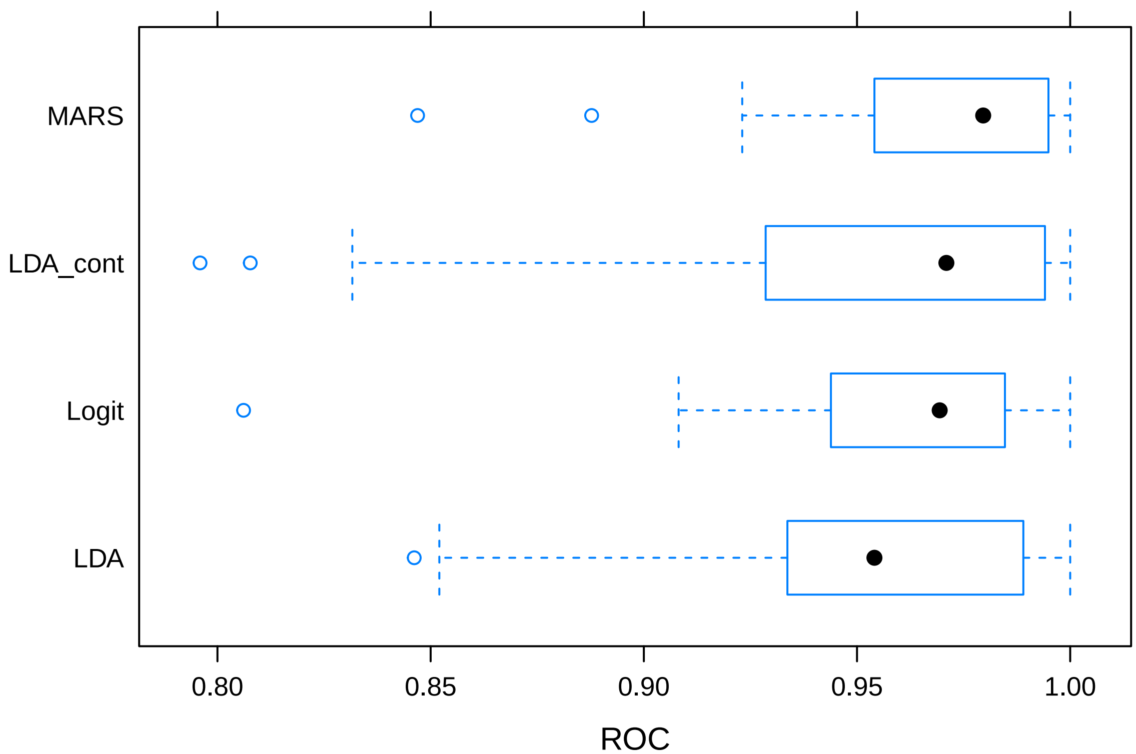
Which model will you use to predict the response variable?

```
set.seed(123)
resamp <- resamples(list(Logit = model.glm,
                         MARS = model.mars,
                         LDA = model.lda,
                         LDA_cont = model.lda.cont
                         ))
summary(resamp)
```

```
##
## Call:
## summary.resamples(object = resamp)
##
## Models: Logit, MARS, LDA, LDA_cont
## Number of resamples: 50
##
## ROC
##               Min.   1st Qu.    Median      Mean   3rd Qu. Max. NA's
## Logit    0.8061224 0.9438776 0.9693878 0.9622214 0.9846939    1    0
## MARS     0.8469388 0.9545722 0.9795918 0.9695683 0.9947998    1    0
## LDA      0.8461538 0.9337716 0.9540816 0.9509184 0.9890110    1    0
## LDA_cont 0.7959184 0.9298469 0.9709576 0.9539766 0.9930111    1    0
```

```
##
## Sens
##               Min.   1st Qu.    Median      Mean   3rd Qu. Max. NA's
## Logit    0.6428571 0.7857143 0.8571429 0.8745055 0.9285714    1    0
## MARS     0.7857143 0.8571429 0.9230769 0.8991209 0.9285714    1    0
## LDA      0.6428571 0.7857143 0.8571429 0.8443956 0.9230769    1    0
## LDA_cont 0.5000000 0.7280220 0.8571429 0.8387912 0.9285714    1    0
##
## Spec
##               Min.   1st Qu.    Median      Mean   3rd Qu. Max. NA's
## Logit    0.7142857 0.8571429 0.9285714 0.8956044 0.9285714    1    0
## MARS     0.7142857 0.9230769 0.9285714 0.9274725 1.0000000    1    0
## LDA      0.8461538 0.9285714 0.9285714 0.9550549 1.0000000    1    0
## LDA_cont 0.7857143 0.9244505 0.9285714 0.9491209 1.0000000    1    0
```

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



While comparing logistic, MARS, and lda through resampling method, MARS model has the highest mean ROC value 0.9700 compare to other three models. Thus I would prefer using MARS model to predict the response variable. On the other hand, the mean ROC value of lda.cont model is 0.9540, lda model is 0.9510, implies the mean ROC value increases after removing categorical predictor `origin`, lda_cont indeed better than the lda model.

Plot its ROC curve using the test data. Report the AUC and the misclassification error rate.

```
set.seed(123)
# plot roc using test data for all models
glm.pred.p <- predict(model.glm, newdata = x2, type = "prob")[,2]
mars.pred.p <- predict(model.mars, newdata = x2, type = "prob")[,2]
```

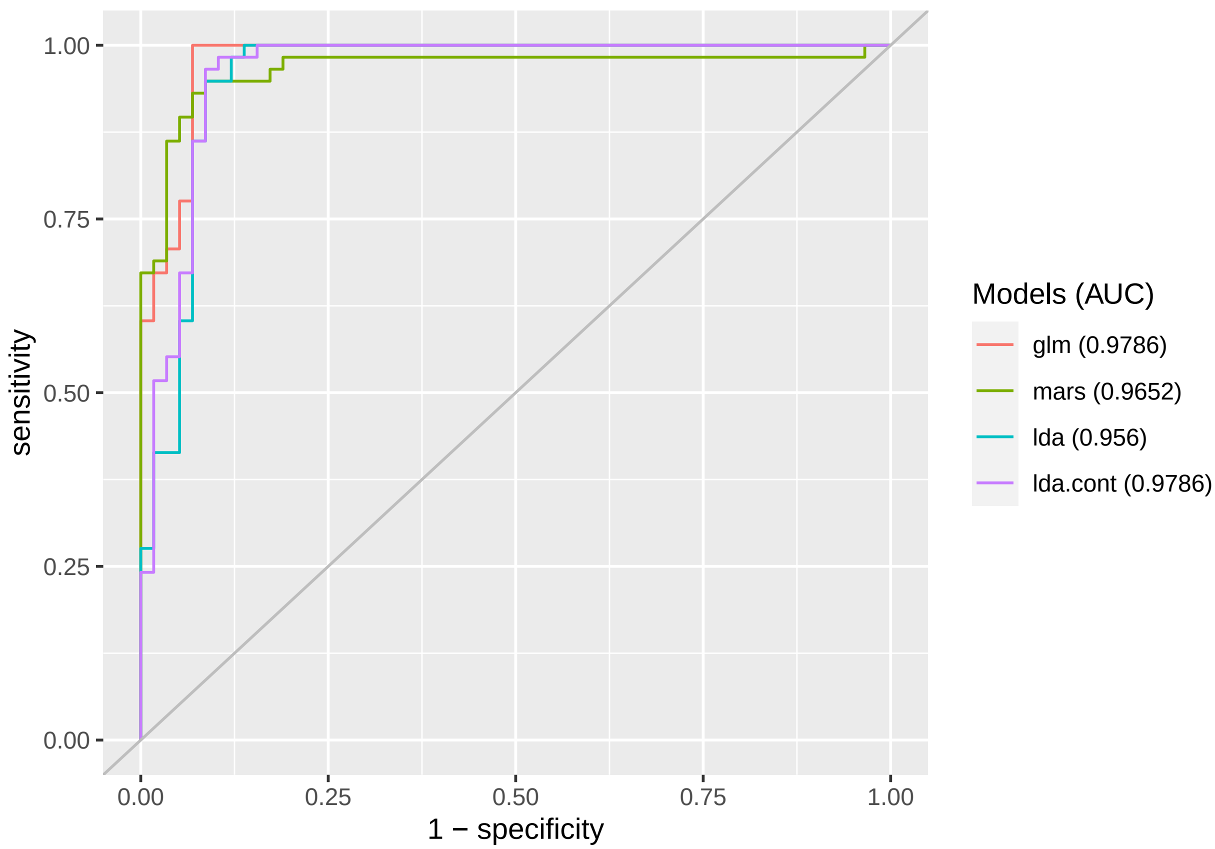
```
lda.pred.p <- predict(model.lda, newdata = x2, type = "prob")[,2]
lda.cont.pred.p <- predict(model.lda.cont, newdata = x2[,1:6], type = "prob")[,2] #remove origin from t

roc.glm <- roc(dat$mpg_cat[-train_rows], glm.pred.p)
roc.mars <- roc(dat$mpg_cat[-train_rows], mars.pred.p)
roc.lda <- roc(dat$mpg_cat[-train_rows], lda.pred.p)
roc.lda.cont <- roc(dat$mpg_cat[-train_rows], lda.cont.pred.p)

# auc
auc <- c(roc.glm$auc[1],
         roc.mars$auc[1],
         roc.lda$auc[1],
         roc.lda$auc.cont[1])

modelNames <- c("glm","mars","lda","lda.cont")

ggroc(list(roc.glm, roc.mars, roc.lda, roc.lda.cont), legacy.axes = TRUE) +
  scale_color_discrete(labels = paste0(modelNames, " (", round(auc,4),")"),
                       name = "Models (AUC)") +
  geom_abline(intercept = 0, slope = 1, color = "grey")
```




```
# misclassification error rate
glm.pred <- rep("low", length(glm.pred.p))
glm.pred[glm.pred.p>0.5] <- "high"
glm_cm <- confusionMatrix(data = as.factor(glm.pred),
                reference = y2,
                positive = "high")
```

```
glm_cm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction low high
##       low   54    3
##       high   4   55
##
##                Accuracy : 0.9397
##                  95% CI : (0.8796, 0.9754)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.8793
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.9483
##             Specificity : 0.9310
##          Pos Pred Value : 0.9322
##          Neg Pred Value : 0.9474
##              Prevalence : 0.5000
##          Detection Rate : 0.4741
##    Detection Prevalence : 0.5086
##       Balanced Accuracy : 0.9397
##
##        'Positive' Class : high
##
```

```
mars.pred <- rep("low", length(mars.pred.p))
mars.pred[mars.pred.p>0.5] <- "high"
mars_cm <- confusionMatrix(data = as.factor(mars.pred),
                reference = y2,
                positive = "high")
mars_cm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction low high
##       low   54    5
##       high   4   53
##
##                Accuracy : 0.9224
##                  95% CI : (0.8578, 0.9639)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.8448
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.9138
```

```
##             Specificity : 0.9310
##          Pos Pred Value : 0.9298
##          Neg Pred Value : 0.9153
##              Prevalence : 0.5000
##          Detection Rate : 0.4569
##    Detection Prevalence : 0.4914
##       Balanced Accuracy : 0.9224
##
##        'Positive' Class : high
##
```

```
lda.pred <- rep("low", length(lda.pred.p))
lda.pred[lda.pred.p>0.5] <- "high"
lda_cm <- confusionMatrix(data = as.factor(lda.pred),
                reference = y2,
                positive = "high")
lda_cm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction low high
##       low   50    0
##       high   8   58
##
##                Accuracy : 0.931
##                  95% CI : (0.8686, 0.9698)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.8621
##
##  Mcnemar's Test P-Value : 0.01333
##
##             Sensitivity : 1.0000
##             Specificity : 0.8621
##          Pos Pred Value : 0.8788
##          Neg Pred Value : 1.0000
##              Prevalence : 0.5000
##          Detection Rate : 0.5000
##    Detection Prevalence : 0.5690
##       Balanced Accuracy : 0.9310
##
##        'Positive' Class : high
##
```

```
lda.cont.pred <- rep("low", length(lda.cont.pred.p))
lda.cont.pred[lda.cont.pred.p>0.5] <- "high"
lda.cont_cm <- confusionMatrix(data = as.factor(lda.pred),
                reference = y2,
                positive = "high")
lda.cont_cm
```

```
## Confusion Matrix and Statistics
##
```

```
##           Reference
## Prediction low high
##       low   50    0
##       high   8   58
##
##                Accuracy : 0.931
##                  95% CI : (0.8686, 0.9698)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.8621
##
##  Mcnemar's Test P-Value : 0.01333
##
##             Sensitivity : 1.0000
##             Specificity : 0.8621
##          Pos Pred Value : 0.8788
##          Neg Pred Value : 1.0000
##              Prevalence : 0.5000
##          Detection Rate : 0.5000
##    Detection Prevalence : 0.5690
##       Balanced Accuracy : 0.9310
##
##        'Positive' Class : high
##
```

```r
# misclassification error rate
glm_er <- 1-glm_cm$byClass[["Balanced Accuracy"]]
round(glm_er, 4)
```

```
## [1] 0.0603
```

```r
mars_er <- 1-mars_cm$byClass[["Balanced Accuracy"]]
round(mars_er, 4)
```

```
## [1] 0.0776
```

```r
lda_er <- 1-lda_cm$byClass[["Balanced Accuracy"]]
round(lda_er, 4)
```

```
## [1] 0.069
```

```r
lda_cont_er <- 1-lda.cont_cm$byClass[["Balanced Accuracy"]]
round(lda_cont_er, 4)
```

```
## [1] 0.069
```

Test data performance:
The plot of ROC curve using test data shows logistic regression model and lda_cont model have the highest AUC values (0.9786) than mars and lda models. The prediction performance of both logistic regression model and lda_cont are very good.
Misclassficition error rate = 1 - accuracy. If set the classifier cut-off to be 0.5, the logistic regression model has the lowest misclassification error rate 0.0603 compare to all other models. However, final decision should not depend on misclassfication error rate because this is only at particular threshold, the result might be different if we change the threshold.