P8106-HW3-yz4184

Yunlin Zhou

Contents

```r
library(caret)
library(glmnet)
library(pROC)
library(pdp)
library(vip)
library(AppliedPredictiveModeling)
library(earth)
library(tidyverse)
library(ggplot2)
library(patchwork)
library(MASS)
```

Data cleaning

```r
# import data
dat = read.csv("./auto.csv")%>%
  na.omit() %>%
  mutate(
    cylinders = as.factor(cylinders),
          year = as.factor(year),
          origin = as.factor(origin),
    mpg_cat = factor(mpg_cat, levels = c("low", "high")))
```

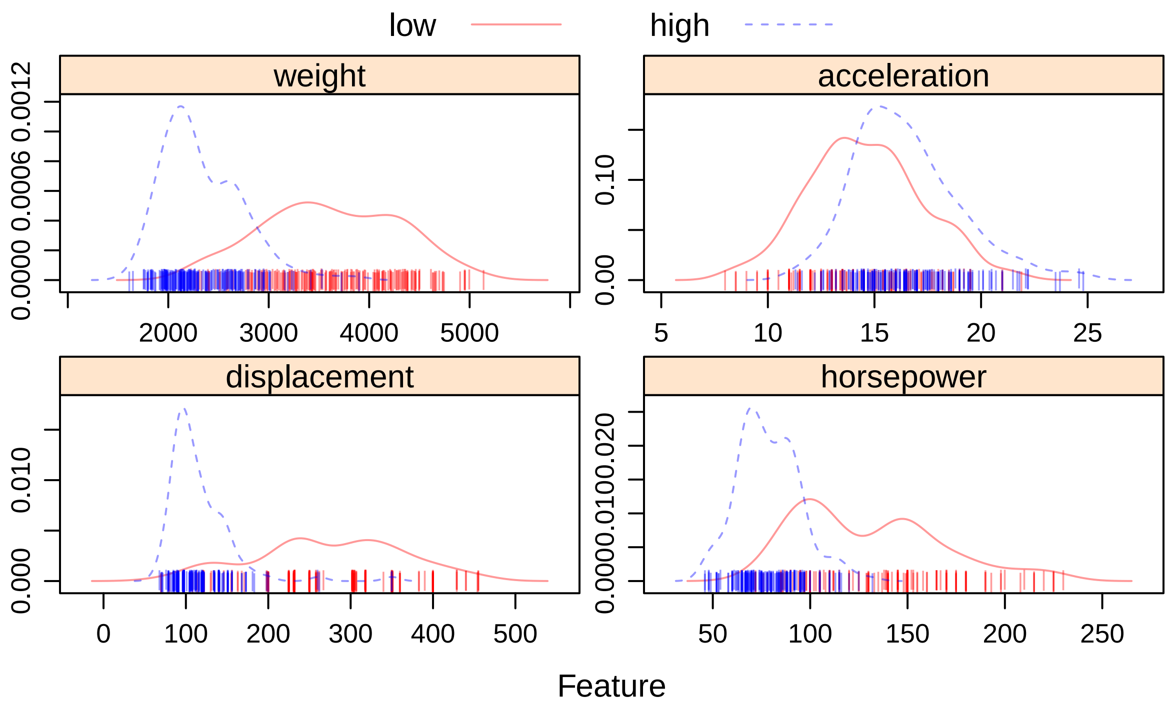
```r
# divide data into two parts (training and test)
set.seed(1)
rowTrain <- createDataPartition(y = dat$mpg_cat,
                                p = 0.7,
                                list = FALSE)
```

```r
train_df = dat[rowTrain,]
test_df = dat[-rowTrain,]
```

(a) Produce some graphical or numerical summaries of the data.

graphical summaries of continuous variables

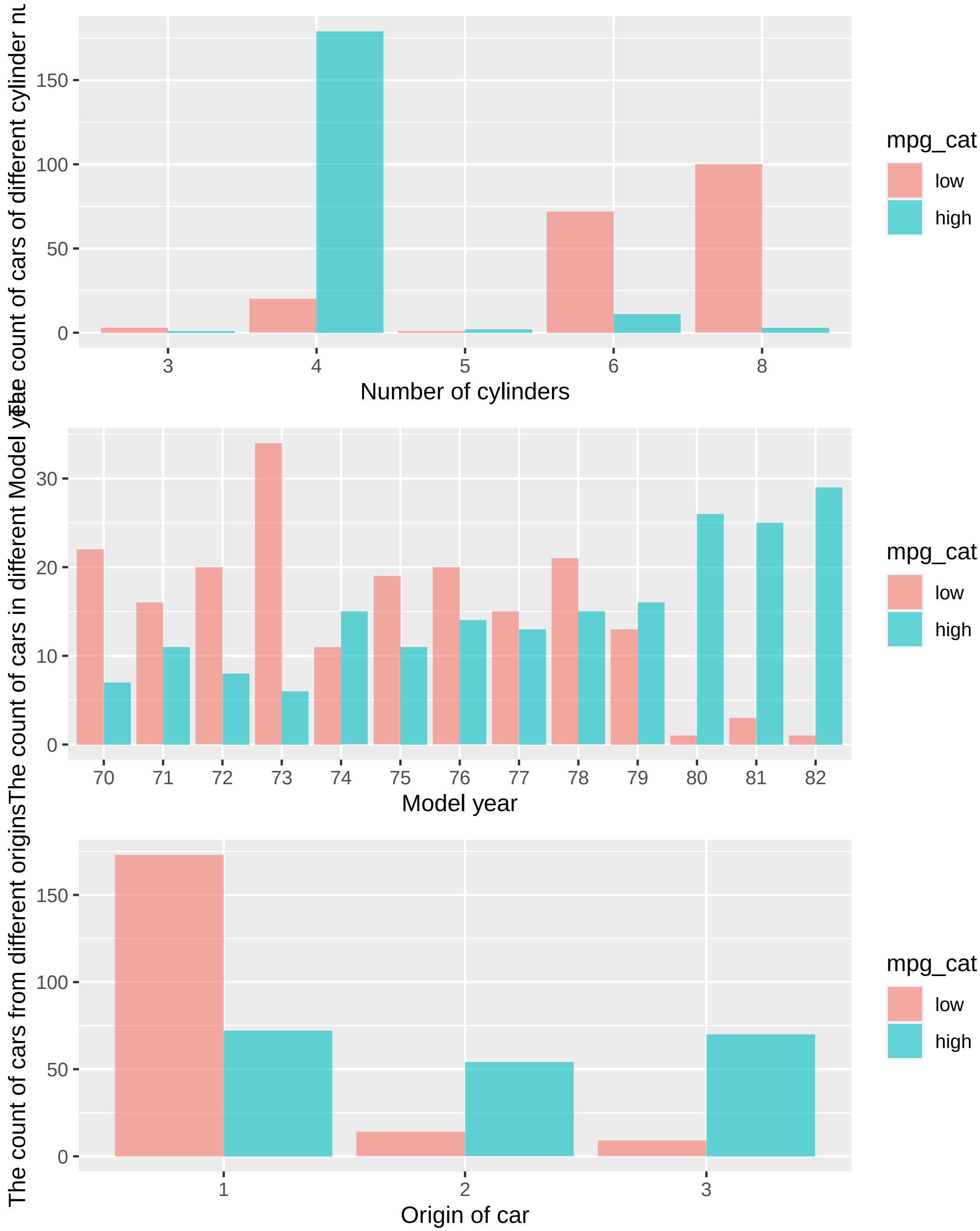
```
theme1 <- transparentTheme(trans = .4)
trellis.par.set(theme1)

featurePlot(x = dat[, 2:5],
            y = dat$mpg_cat,
            scales = list(x = list(relation = "free"),
                          y = list(relation = "free")),
            plot = "density", pch = "|",
            auto.key = list(columns = 2))
```



As shown in the density plots above, we can conclude that the cars with high miles per gallon are tending to have lower weights; larger time to accelerate from 0 to 60 mph; lower engine displacement and lowerhorse power.

graphical summaries of catagorical variables

```
p_cylinders = dat%>%
  ggplot(aes(x = dat[,1], fill = mpg_cat)) +
  geom_bar(stat = "count",
           position = position_dodge(),
           alpha = 0.6)+
  labs(
    x = "Number of cylinders",
    y = "The count of cars of different cylinder number"
  )

p_year = dat%>%
  ggplot(aes(x = dat[,6], fill = mpg_cat)) +
  geom_bar(stat = "count",
           position = position_dodge(),
           alpha = 0.6)+
  labs(
    x = "Model year",
    y = "The count of cars in different Model year"
  )

p_origin = dat%>%
  ggplot(aes(x = dat[,7], fill = mpg_cat)) +
  geom_bar(stat = "count",
           position = position_dodge(),
           alpha = 0.6)+
  labs(
    x = "Origin of car",
    y = "The count of cars from different origins"
  )

grid.arrange(p_cylinders, p_year,p_origin, nrow = 3)
```



As we can see from the plot above: 4 cylinders car are tending to have the high miles per gallon; as the time went by, the cars are tending to have high miles per gallon; Many American cars have low miles per gallon.

(b) Perform a logistic regression using the training data.

fit the logistic regression model using the training data

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

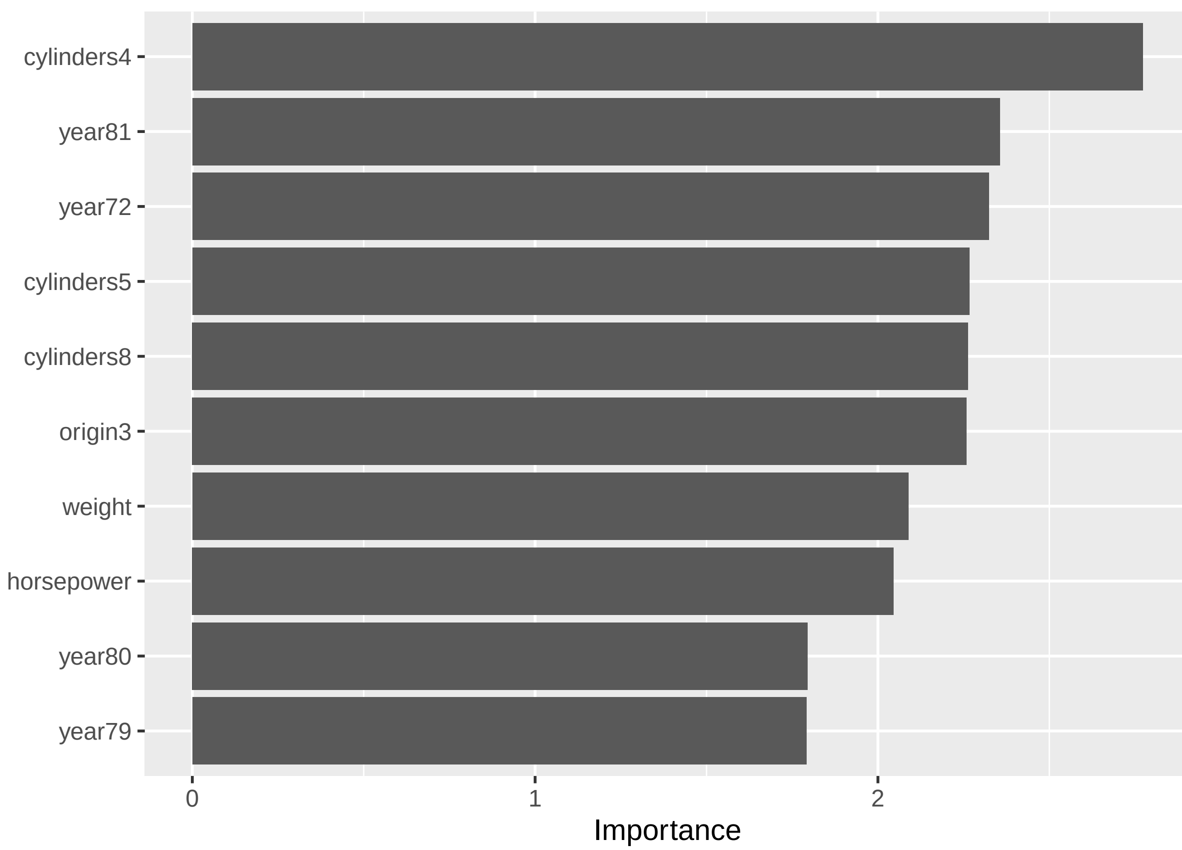
```
##      high
## low     0
## high    1
```

```
# Using caret
ctrl <- trainControl(method = "repeatedcv",
                     summaryFunction = twoClassSummary,
                     classProbs = TRUE)
set.seed(1)
model.glm <- train(mpg_cat ~ .,
                  data = train_df,
                   method = "glm",
                   metric = "ROC",
                   trControl = ctrl)

summary(model.glm)
```

```
##
## Call:
## NULL
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.9453  -0.0344   0.0000   0.0116   3.4974
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)   1.917e+01  9.727e+00   1.971  0.04874 *
## cylinders4    1.146e+01  4.133e+00   2.773  0.00556 **
## cylinders5    1.056e+01  4.659e+00   2.266  0.02343 *
## cylinders6    6.695e+00  3.959e+00   1.691  0.09079 .
## cylinders8    1.220e+01  5.389e+00   2.263  0.02363 *
## displacement  1.763e-02  2.501e-02   0.705  0.48086
## horsepower   -1.317e-01  6.437e-02  -2.046  0.04080 *
## weight       -6.143e-03  2.941e-03  -2.088  0.03676 *
## acceleration -2.191e-01  3.511e-01  -0.624  0.53252
## year71       -7.866e-01  3.573e+00  -0.220  0.82576
## year72       -4.829e+00  2.078e+00  -2.323  0.02016 *
## year73       -1.618e+00  2.305e+00  -0.702  0.48270
## year74        4.546e-01  5.102e+00   0.089  0.92899
## year75        7.168e-01  1.883e+00   0.381  0.70340
## year76        2.198e+00  2.352e+00   0.935  0.34997
## year77       -5.362e-01  2.284e+00  -0.235  0.81436
## year78        7.792e-02  2.379e+00   0.033  0.97387
## year79        4.322e+00  2.413e+00   1.792  0.07320 .
```

```
## year80        5.317e+00  2.962e+00   1.795  0.07262 .
## year81        5.313e+00  2.256e+00   2.355  0.01851 *
## year82        2.301e+01  1.712e+03   0.013  0.98928
## origin2       6.362e-01  1.529e+00   0.416  0.67736
## origin3       7.052e+00  3.123e+00   2.258  0.02392 *
## ---
## 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:  51.724  on 253  degrees of freedom
## AIC: 97.724
##
## Number of Fisher Scoring iterations: 18
```

```
vip(model.glm$finalModel)
```



According to the z-acore and vip plot, we can conclude that cylinders4, year 81, year 72, cylinder5, cylinders8, oringin3, weight, hoursepower, year 80 and year 79 are statistically significant.

Compute the confusion matrix and overall fraction of correct predictions using the test data

confusion matrix

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

test.pred.prob1 <- predict(glm.fit, newdata = test_df,
                           type = "response")
test.pred1 <- rep("low", length(test.pred.prob1))
test.pred1[test.pred.prob1>0.5] <- "high"
confusionMatrix(data = as.factor(test.pred1),
                reference = test_df$mpg_cat,
                positive = "high")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction low high
##       low   49    3
##       high   9   55
##
##                Accuracy : 0.8966
##                  95% CI : (0.8263, 0.9454)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.7931
##
##  Mcnemar's Test P-Value : 0.1489
##
##             Sensitivity : 0.9483
##             Specificity : 0.8448
##          Pos Pred Value : 0.8594
##          Neg Pred Value : 0.9423
##              Prevalence : 0.5000
##          Detection Rate : 0.4741
##    Detection Prevalence : 0.5517
##       Balanced Accuracy : 0.8966
##
##        'Positive' Class : high
##
```

The accuracy of this model is 0.8966. Since the P-Value [Acc > NIR] is small, we can conclude that the classification is good. The kappa is 0.7931 and it's large, which means our collected data is a good representative. Sensitivity and Specificity are both high.

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

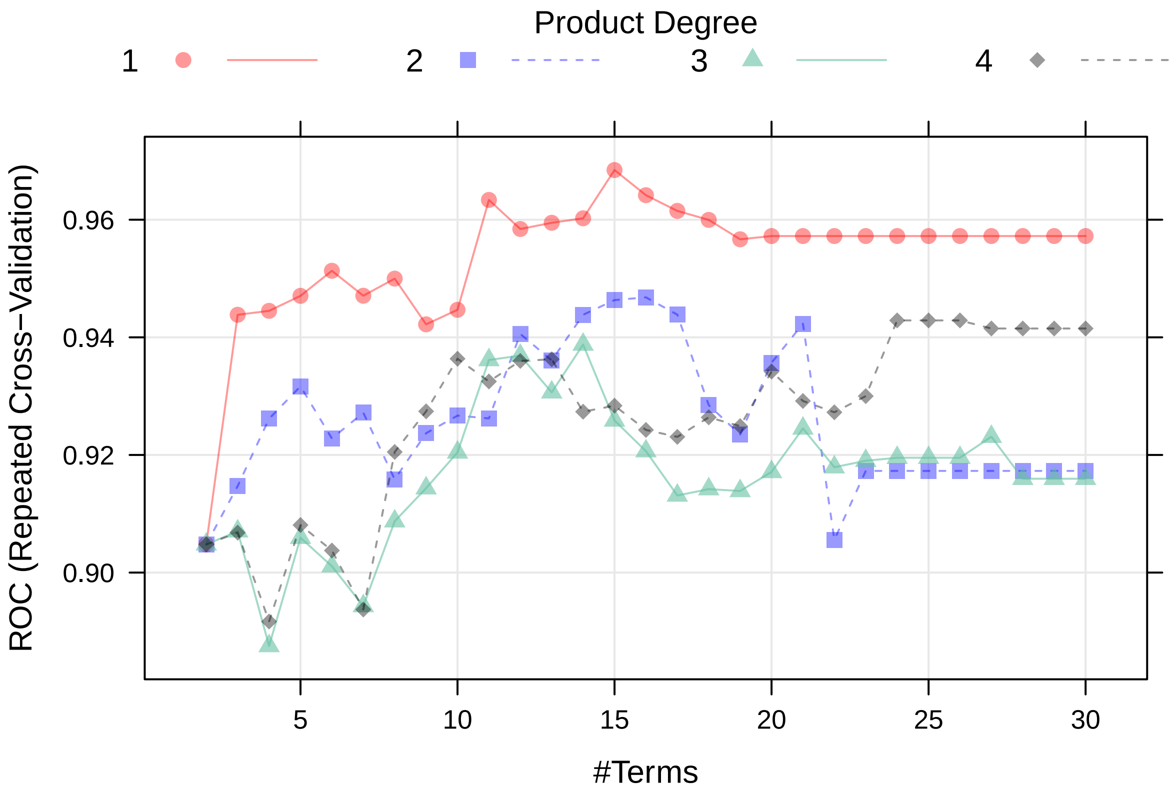
```
set.seed(1)
model.mars <- train(mpg_cat ~ .,
                  data = train_df,
                    method = "earth",
                    tuneGrid = expand.grid(degree = 1:4,
                                           nprune = 2:30),
                    metric = "ROC",
                    trControl = ctrl)

model.mars$bestTune
```

```
##    nprune degree
## 14     15      1
```

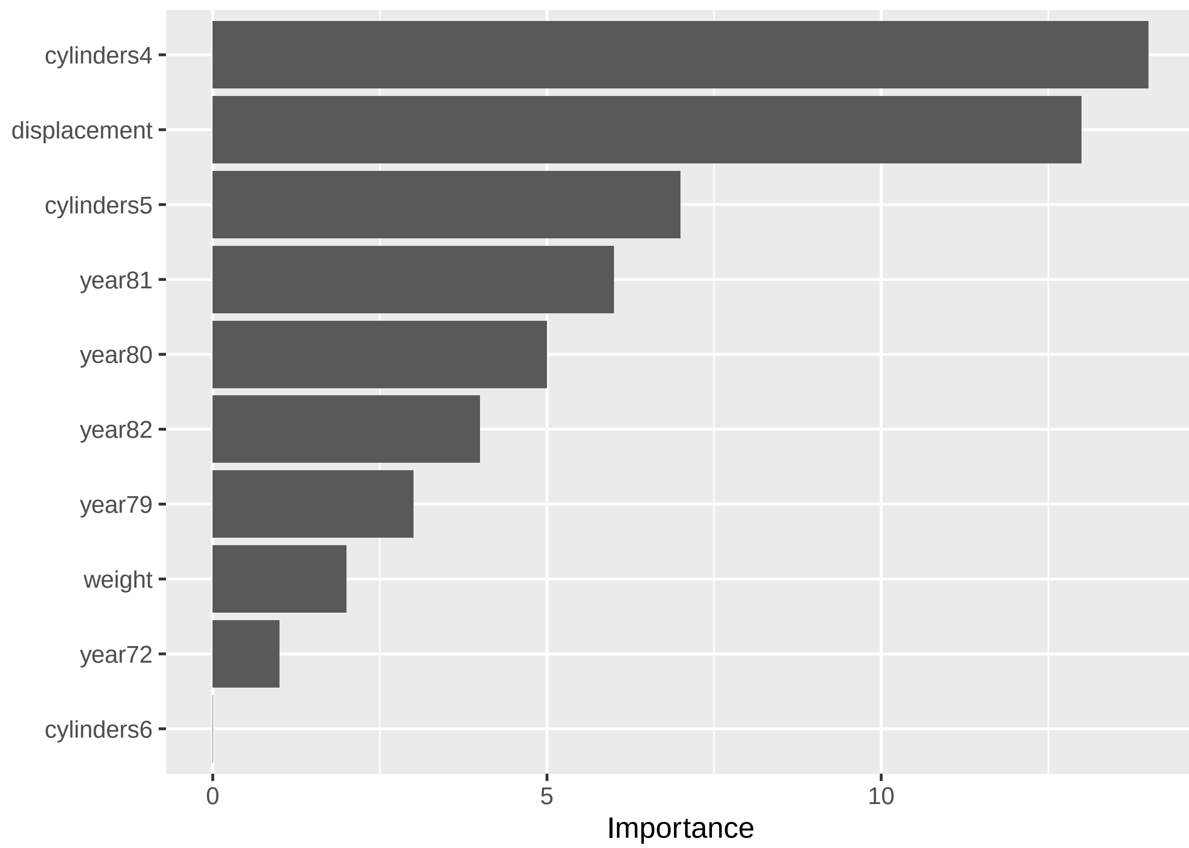
```
coef(model.mars$finalModel)
```

```
##          (Intercept)           cylinders4               year81               year80
##         -1.595167162          6.528127047          4.871466951          5.195997250
##               year82               year79     h(weight-3353) h(displacement-171)
##         18.511131974          3.869433119          0.002695681         -0.355514608
## h(displacement-122) h(displacement-119)           cylinders5 h(displacement-156)
##          3.157707427         -2.777811991          5.942441988          0.533711798
## h(displacement-146) h(displacement-200)     h(weight-2542)
##         -0.650154956          0.104929219         -0.004811159
```

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

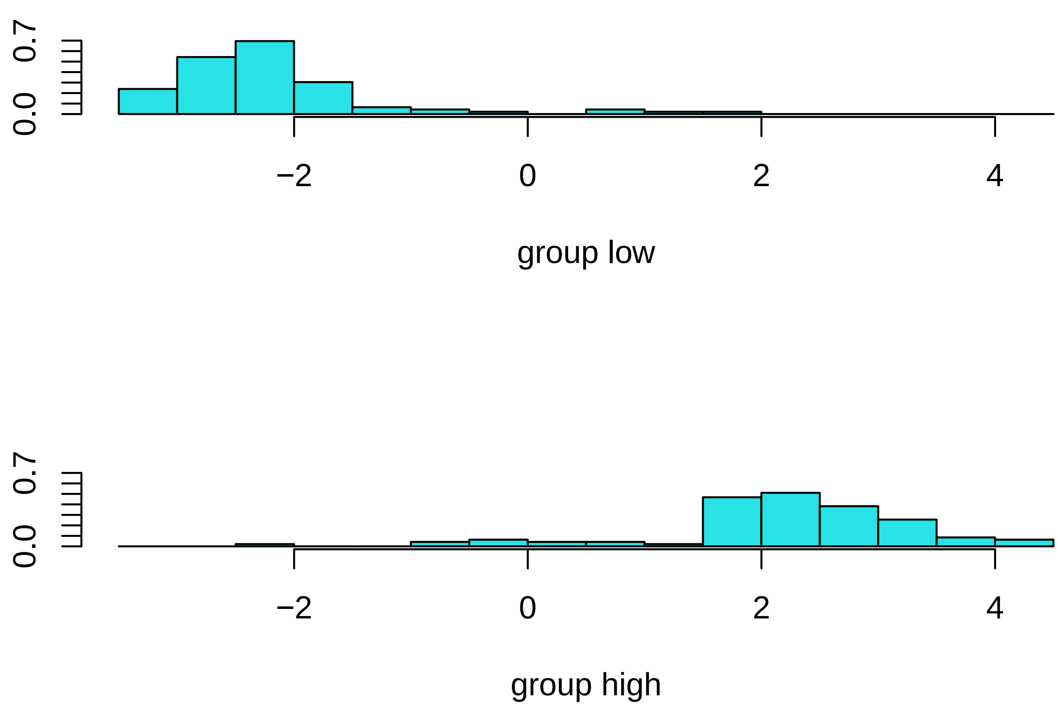


```
vip(model.mars$finalModel)
```



(d) Perform LDA using the training data. Plot the linear discriminants in LDA.

```r
lda.fit <- lda(mpg_cat~., data = train_df,
               subset = rowTrain)

plot(lda.fit)
```




```r
lda.fit$scaling
```

```
##                        LD1
## cylinders4    4.8867673343
## cylinders5    3.2144403810
## cylinders6    1.9017853625
## cylinders8    2.7929881179
## displacement -0.0061909365
## horsepower    0.0073989760
## weight       -0.0006595315
## acceleration  0.0504747270
## year71        0.4419528870
## year72       -0.1174207512
## year73        0.0850625020
## year74        0.5118653304
```

```
## year75        0.4829029801
## year76       -0.1959107737
## year77        0.4431627100
## year78       -0.1230282631
## year79        0.8287571259
## year80        1.3832241871
## year81        1.9812076901
## year82        1.0662148484
## origin2      -0.1980882046
## origin3       0.2290881919
```

```
head(predict(lda.fit)$x)
```

```
##          LD1
## 1  -2.372207
## 16 -2.228355
## 19  1.972383
## 23  1.543502
## 26 -2.703204
## 27 -2.277966
```

```
mean(predict(lda.fit)$x)
```

```
## [1] 1.278939e-16
```

Using caret

```
set.seed(1)
model.lda = train(mpg_cat ~ .,
                  data = train_df,
                  method = "lda",
                  metric = "ROC",
                  trControl = ctrl)
model.lda$bestTune
```

```
##   parameter
## 1      none
```

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

```
##                        LD1
## cylinders4    3.0544921070
## cylinders5    2.4951990604
## cylinders6    0.4593174150
## cylinders8    1.2492044367
## displacement -0.0026653700
## horsepower    0.0008400291
## weight       -0.0006320044
## acceleration  0.0214785874
```

```
## year71         0.1447224742
## year72        -0.8082829932
## year73        -0.2125876582
## year74         0.3247951606
## year75        -0.0009048596
## year76        -0.0283926888
## year77        -0.0398074545
## year78        -0.1600289333
## year79         0.6820353162
## year80         1.0644286784
## year81         1.3153908105
## year82         1.0782795295
## origin2       -0.0316863756
## origin3        0.4614236179
```

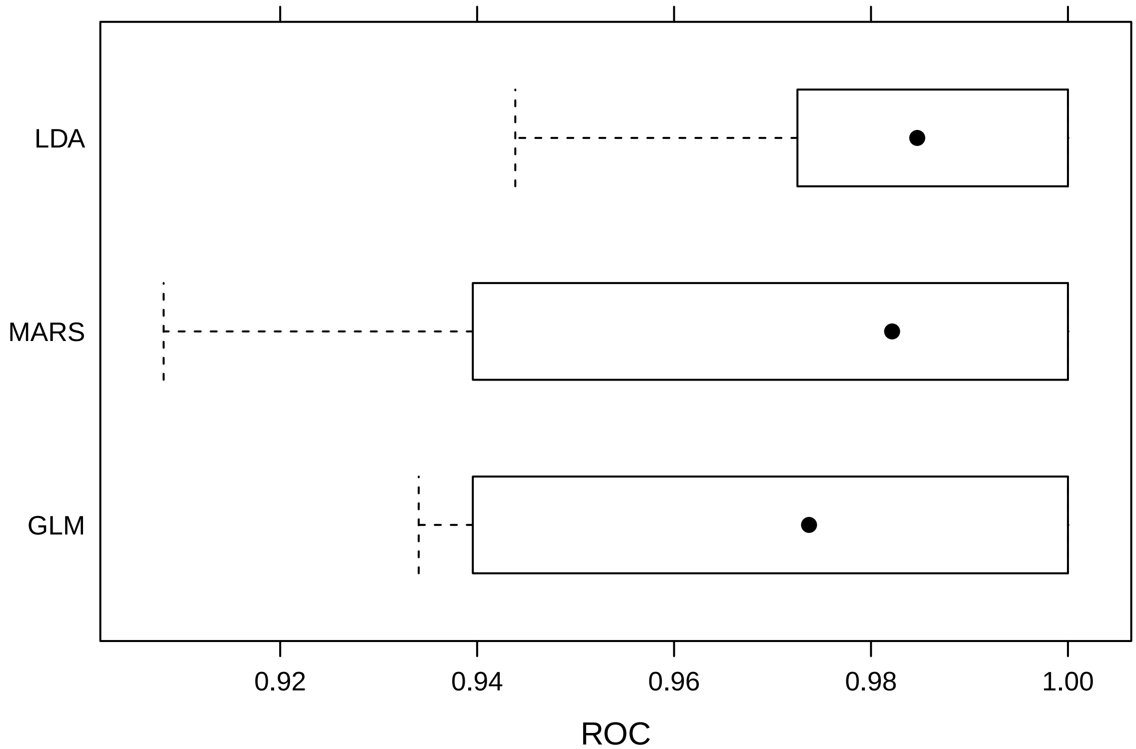
(e) Which model will you use to predict the response variable?

Using box plot to show the model with largest AUC

```
res <- resamples(list(GLM = model.glm,
                      MARS = model.mars,
                      LDA = model.lda))
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  0.9340659 0.9444662 0.9737049 0.9708006 0.9972527    1    0
## MARS 0.9081633 0.9419152 0.9821429 0.9684458 0.9987245    1    0
## LDA  0.9438776 0.9752747 0.9846939 0.9819859 0.9972527    1    0
##
## Sens
##           Min.   1st Qu.    Median      Mean   3rd Qu. Max. NA's
## GLM  0.7692308 0.8750000 0.9285714 0.8983516 0.9285714    1    0
## MARS 0.7857143 0.9244505 0.9285714 0.9351648 1.0000000    1    0
## LDA  0.8571429 0.8571429 0.8901099 0.9137363 0.9821429    1    0
##
## Spec
##           Min.   1st Qu.    Median      Mean 3rd Qu. Max. NA's
## GLM  0.7857143 0.8571429 0.9285714 0.9203297       1    1    0
## MARS 0.7857143 0.8736264 0.9285714 0.9208791       1    1    0
## LDA  0.6428571 0.8571429 0.9285714 0.9060440       1    1    0
```

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

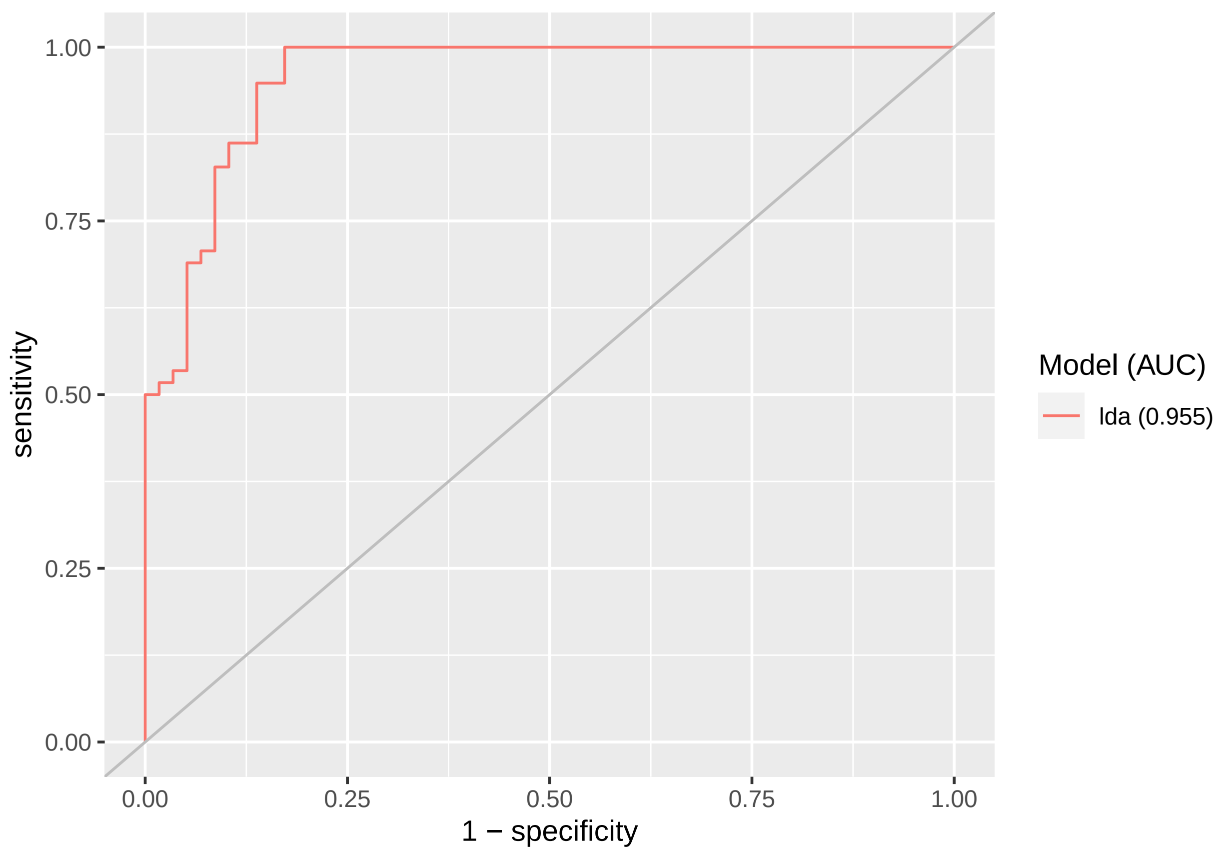


From the summary and the box-plot, we can conclude that LDA has the largest AUC, thus we choose LDA as our model.

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

ROC curve

```
lda.pred <- predict(model.lda, newdata = test_df, type = "prob")[,2]

roc.lda <- roc(test_df$mpg_cat, lda.pred)

auc <- roc.lda$auc[1]

modelName <- "lda"

ggroc(list(roc.lda), legacy.axes = TRUE) +
  scale_color_discrete(labels = paste0(modelName, " (", round(auc,3),")"),
                       name = "Model (AUC)") +
  geom_abline(intercept = 0, slope = 1, color = "grey")
```



From the plot above we can conclude that the AUC of LDA model is 0.955 which is very close to 1.

confusion Matrix

```
test.pred2 <- rep("low", length(lda.pred))
test.pred2[lda.pred >0.5] <- "high"


confusionMatrix(data = as.factor(test.pred2),
                reference = test_df$mpg_cat,
                positive = "high")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction low high
##       low   49    3
##       high   9   55
##
##                Accuracy : 0.8966
##                  95% CI : (0.8263, 0.9454)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.7931
##
##  Mcnemar's Test P-Value : 0.1489
##
##             Sensitivity : 0.9483
##             Specificity : 0.8448
##          Pos Pred Value : 0.8594
##          Neg Pred Value : 0.9423
##              Prevalence : 0.5000
##          Detection Rate : 0.4741
##    Detection Prevalence : 0.5517
##       Balanced Accuracy : 0.8966
##
##        'Positive' Class : high
##
```

The LDA model has a misclassification rate of 1 - 0.8966 = 0.1034.