p8106_hw3_wq2160

Wenshan Qu (wq2160)

3/18/2022

Contents

Import the data.

```
auto =
  read_csv("./auto.csv") %>%
  mutate(
    mpg_cat = as.factor(mpg_cat),
    origin = as.factor(origin)
  )

set.seed(33)
trainRows = createDataPartition(y = auto$mpg_cat, p = 0.7, list = FALSE)

train_data = auto[trainRows, ]
test_data = auto[-trainRows, ]

x = model.matrix(mpg_cat ~ ., train_data)[,-1]
y = train_data$mpg_cat
x_test = model.matrix(mpg_cat ~ ., test_data)[,-1]
y_test = test_data$mpg_cat
```

Data Summary

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

```
## numerical summary
summary(train_data)
```

```
##    cylinders      displacement     horsepower        weight      acceleration
##  Min.   :3.000   Min.   : 70.0   Min.   : 46.0   Min.   :1613   Min.   : 8.00
##  1st Qu.:4.000   1st Qu.:100.2   1st Qu.: 75.0   1st Qu.:2225   1st Qu.:13.60
##  Median :4.000   Median :145.0   Median : 94.0   Median :2782   Median :15.40
##  Mean   :5.482   Mean   :195.8   Mean   :105.0   Mean   :2974   Mean   :15.53
##  3rd Qu.:8.000   3rd Qu.:302.0   3rd Qu.:129.2   3rd Qu.:3654   3rd Qu.:17.30
##  Max.   :8.000   Max.   :455.0   Max.   :225.0   Max.   :4997   Max.   :24.60
##       year       origin  mpg_cat
##  Min.   :70.00   1:175   high:138
##  1st Qu.:73.00   2: 48   low :138
##  Median :76.00   3: 53
##  Mean   :75.93
##  3rd Qu.:79.00
##  Max.   :82.00
```

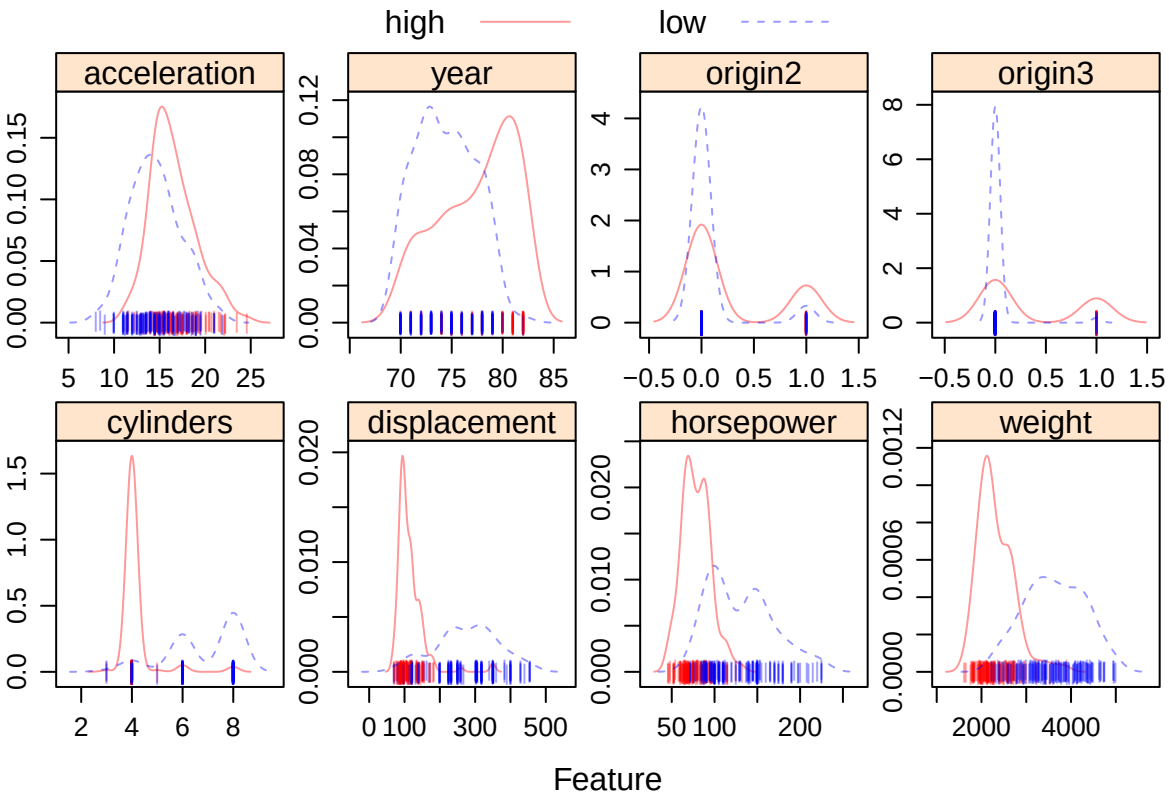
There are 276 observations in the training data set, 116 observations in the test data set, thus there are 392 observations in the whole data set. And there are 7 predictors including `cylinders`, `displacement`, `horsepower`, `weight`, `acceleration`, `year` and `origin`. I treat `origin` as dummy variable.

we can also find out the mean and median of each predictor in the above summary results, as well as the observed outcome, which is: within 276 observations in the training data, 138 belong to `high` class, and 138 belong to `low` class.

```
## Feature plot
theme1 <- transparentTheme(trans = .4)
```

```
trellis.par.set(theme1)

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




```
## We can see that 0 is "high" and 1 is "low"
contrasts(auto$mpg_cat)
```

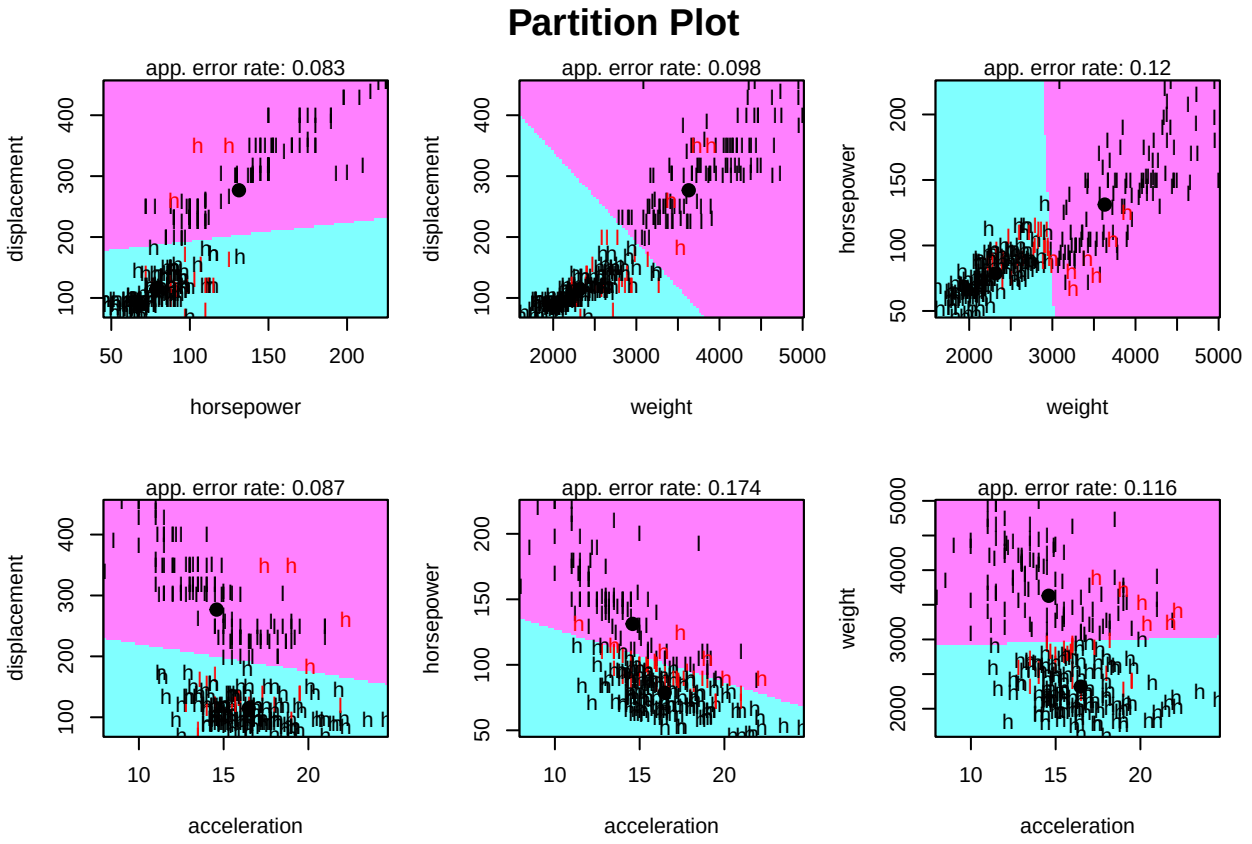
```
##      low
## high   0
## low    1
```

Feature plots can give us a general idea about whether each variable could provide useful information to the classification.

- `acceleration`: the distribution of data points in this plot actually overlaps a lot between two response classes, which means this predictor may not provide useful information in prediction the class.
- `year`: we can see that in terms of `year`, data points separate quite well, though still there is some overlap of classes exists around year = 80, this predictor may be significant.
- `origin2`: data points have severe overlap, may not provide useful information in classification.
- `origin3`: may be a influential predictor.

- `cylinders`: overlap is not a big problem in this case, while the information provided by this predictor is still questionable, since most data points take the value of 4 in this case, which is quite reasonable because most of the cars in the real world has 4 cylinders, and those with more than 4 cylinders are less common. Only when enough "rare" data points ($>$ 4 cylinders) included in this data set could make this predictor valuable in our prediction.
- `displacement`: the situation of this predictor is quite similar with `cylinders`: less overlap, and one class has more data points than the other. While this predictor may have contribution to our prediction, since the data points in the "low" class is not that rare.
- `horsepower`: this predictor may have less significant contribution to prediction, since generally the data points separate good, but the overlap around 75 $\sim$ 125 seems to be severe, more statistical analysis is needed.
- `weight`: this predictor has pretty perfect separation, has large potential to be statistically significant.

```
## LDA based on every combination of two variables
partimat(mpg_cat ~ displacement + horsepower + weight +acceleration,
         data = auto, subset = trainRows, method = "lda")
```



Logistic Regression

(b) Perform a logistic regression using the training data. Do any of the predictors appear to be statistically significant? If so, which ones? Compute the confusion matrix and overall fraction of correct predictions using the test data. Briefly explain what the confusion matrix is telling you.

GLM model - without penalization

Method 1: Using `glm` function

```
glm.fit = glm(mpg_cat ~ .,
              data = train_data,
              subset = trainRows,
              family = binomial(link = "logit"))

summary(glm.fit)
```

```
##
## Call:
## glm(formula = mpg_cat ~ ., family = binomial(link = "logit"),
##     data = train_data, subset = trainRows)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.57809  -0.12944  -0.00511   0.06715   2.25924
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  27.538712   9.574398   2.876  0.00402 **
## cylinders     0.698766   0.595644   1.173  0.24075
## displacement -0.041030   0.017621  -2.328  0.01989 *
## horsepower    0.039301   0.032471   1.210  0.22614
## weight        0.008285   0.002121   3.906 9.38e-05 ***
## acceleration -0.097118   0.224977  -0.432  0.66597
## year         -0.645769   0.153143  -4.217 2.48e-05 ***
## origin2      -1.908817   1.052223  -1.814  0.06967 .
## origin3      -2.178285   1.106426  -1.969  0.04898 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 273.054  on 196  degrees of freedom
## Residual deviance:  71.003  on 188  degrees of freedom
##   (79 observations deleted due to missingness)
## AIC: 89.003
##
## Number of Fisher Scoring iterations: 8
```

From the summary results of glm model, we can see that predictors `displacement`, `weight`, `year` and `origin3` appear to be statistically significant, while others are not.

We consider the simple classifier with a cut-off of 0.5 and evaluate its performance on test data.

```
## Confusion Matrix
test.pred.prob = predict(glm.fit, test_data, type = "response")
test.pred = rep("high", length(test.pred.prob))
test.pred[test.pred.prob > 0.5] <- "low"

confusionMatrix(data = as.factor(test.pred),
```

```
                reference = auto$mpg_cat[-trainRows],
                positive = "low")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction high low
##       high   54   8
##       low     4  50
##
##                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 : 0.3865
##
##             Sensitivity : 0.8621
##             Specificity : 0.9310
##          Pos Pred Value : 0.9259
##          Neg Pred Value : 0.8710
##              Prevalence : 0.5000
##          Detection Rate : 0.4310
##    Detection Prevalence : 0.4655
##       Balanced Accuracy : 0.8966
##
##        'Positive' Class : low
##
```

The **overall fraction of correct predictions** using test data, which is actually the `accuracy` in the above results, is 0.8966, which is calculated by $\frac{54+50}{116}$. And based on this fraction, we can say that the model predicts (or classifies) pretty well since larger accuracy close to 1 is preferred.

For the illustration of confusion matrix, we first define: TN is 54, presented by **a**, TP is 50, presented by **b**, FN is 8, presented by **c**, FP is 4, presented by **d**.

The information given by confusion matrix includes:

- The `95% CI` of the accuracy is (0.8263, 0.9454), which further support our assumption of the goodness of model.
- The `no information rate`, defined by $max(\frac{a+c}{n}, \frac{b+d}{n})$ is 0.5.
- The `P-Value [Acc > NIR]`, which is the p-value of `accuracy` > `no information rate` is significantly less than 0.05, thus reject the null hypothesis and conclude that our model is significant.
- The `Kappa` is calculated by $\frac{P_0-P_e}{1-P_e}$, where $P_0 = \frac{a+d}{n}$, and $P_e = \frac{a+c}{n}\frac{a+b}{n} + \frac{b+d}{n}\frac{c+d}{n}$.The kappa coefficient measures the agreement between classification and truth values. A kappa value close to 1 is preferred, thus the kappa = 0.7931 in this case shows the model is quite good.
- The `sensitivity` is $\frac{d}{b+d} = 0.8621$, is the rate of cars with low gas mileage been correctly classified as low by the model.

- The `specificity` is $\frac{a}{a+c} = 0.9310$, is the rate of cars with high gas mileage been correctly classified as high by the model.
- The `PPV` is $\frac{d}{d+c} = 0.9259$, is the rate of the predicted low gas mileage cars are actually in the low class.
- The `NPV` is $\frac{a}{a+b} = 0.8710$, is the rate of the predicted high gas mileage cars are actually in the high class.
- The `prevalence` is $\frac{b+d}{n} = 0.5$, is the percentage of low gas mileage cars in the test data set, which is not meaningful.
- The `detection rate` is $\frac{d}{n} = 0.4310$.
- The `detection prevalence` is $\frac{c+d}{n} = 0.4655$.
- The `balance accuracy` is $\frac{Sensitivity+Specificity}{2} = 0.8966$.

Method 2: Using `caret` package

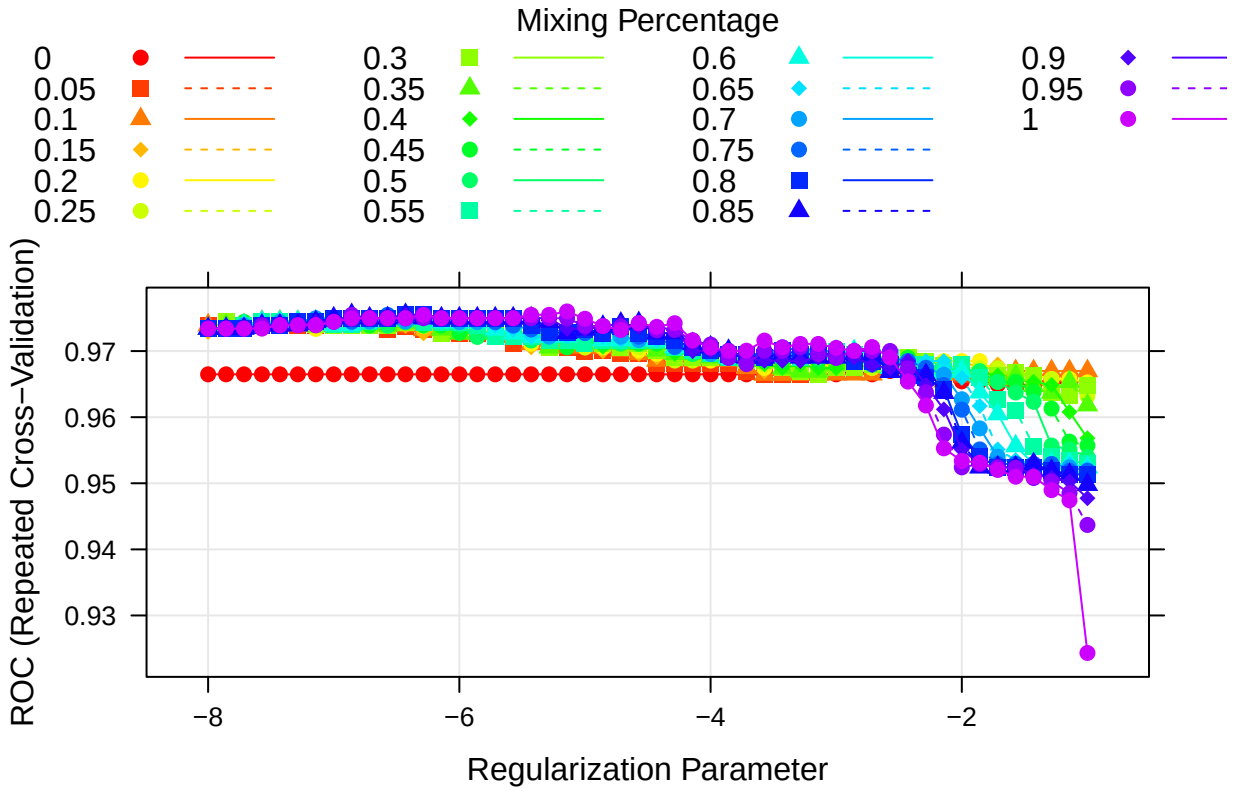
This is to compare the cv performance with other models, rather than tuning the model.

```
ctrl1 = trainControl(method = "repeatedcv",
                     summaryFunction = twoClassSummary,
                     classProbs = TRUE)

set.seed(33)
model.glm = train(x, y,
                  method = "glm",
                  metric = "ROC",
                  trControl = ctrl1)
```

GLMNET model - with penalization (Penalized Logistic Regression)

```
glmnGrid = expand.grid(.alpha = seq(0, 1, length = 21),
                       .lambda = exp(seq(-8, -1, length = 50)))
set.seed(33)
model.glmn = train(x,
                   y,
                   method = "glmnet",
                   tuneGrid = glmnGrid,
                   metric = "ROC",
                   trControl = ctrl1)
## Rainbow Plot
myCol = rainbow(25)
myPar = list(superpose.symbol = list(col = myCol),
             superpose.line = list(col = myCol))
plot(model.glmn, par.settings = myPar, xTrans = function(x) log(x))
```




```
## Best Tune
model.glmn$bestTune
```

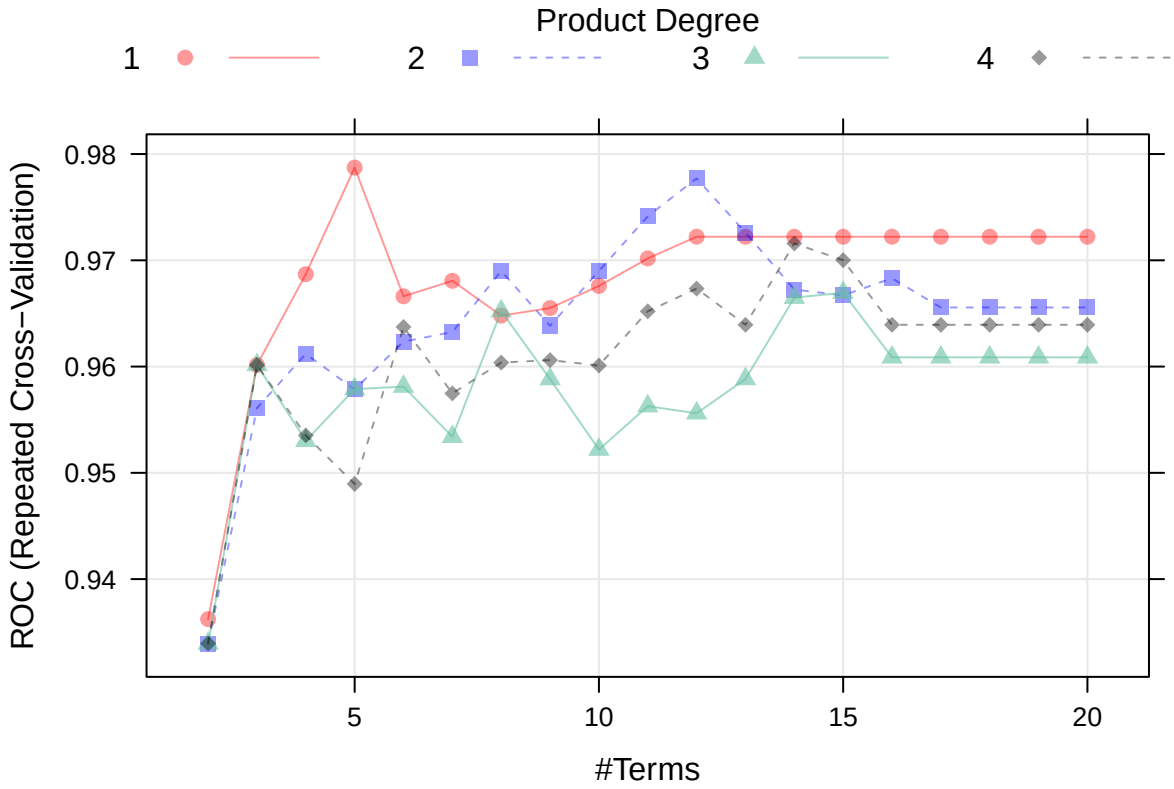
```
##      alpha      lambda
## 1021     1 0.005840977
```

MARS model

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

```
set.seed(33)
model.mars <- train(x,
                    y,
                    method = "earth",
                    tuneGrid = expand.grid(degree = 1:4,
                                           nprune = 2:20),
                    metric = "ROC",
                    trControl = ctrl1)

plot(model.mars)
```




```
model.mars$bestTune
```

```
##   nprune degree
## 4      5      1
```

Higher AUC is preferred, and based on the plots, we can see that when degree = 1 with 5 terms, the curve achieve its highest point, which is actually the same result with the extracted best tunes from the final MARS model.

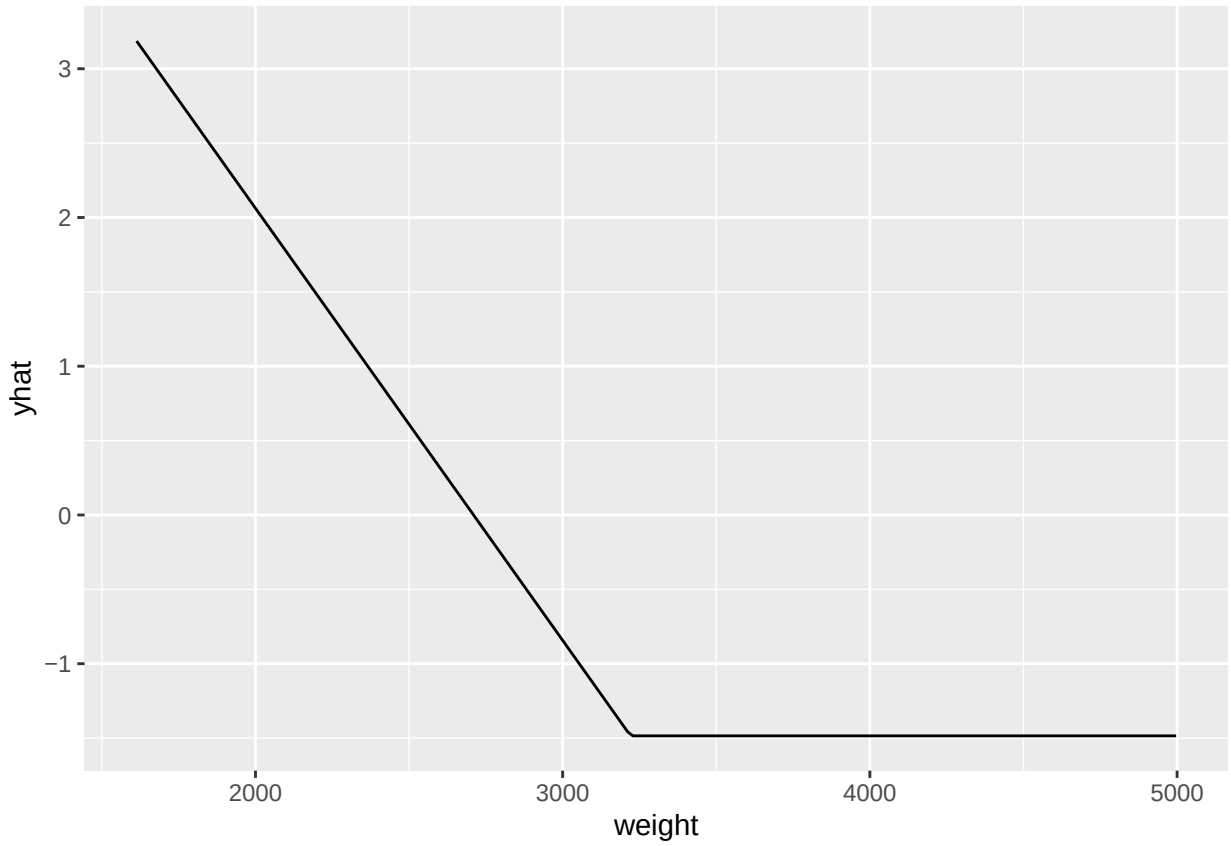
The coefficients of the MARS model are:

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

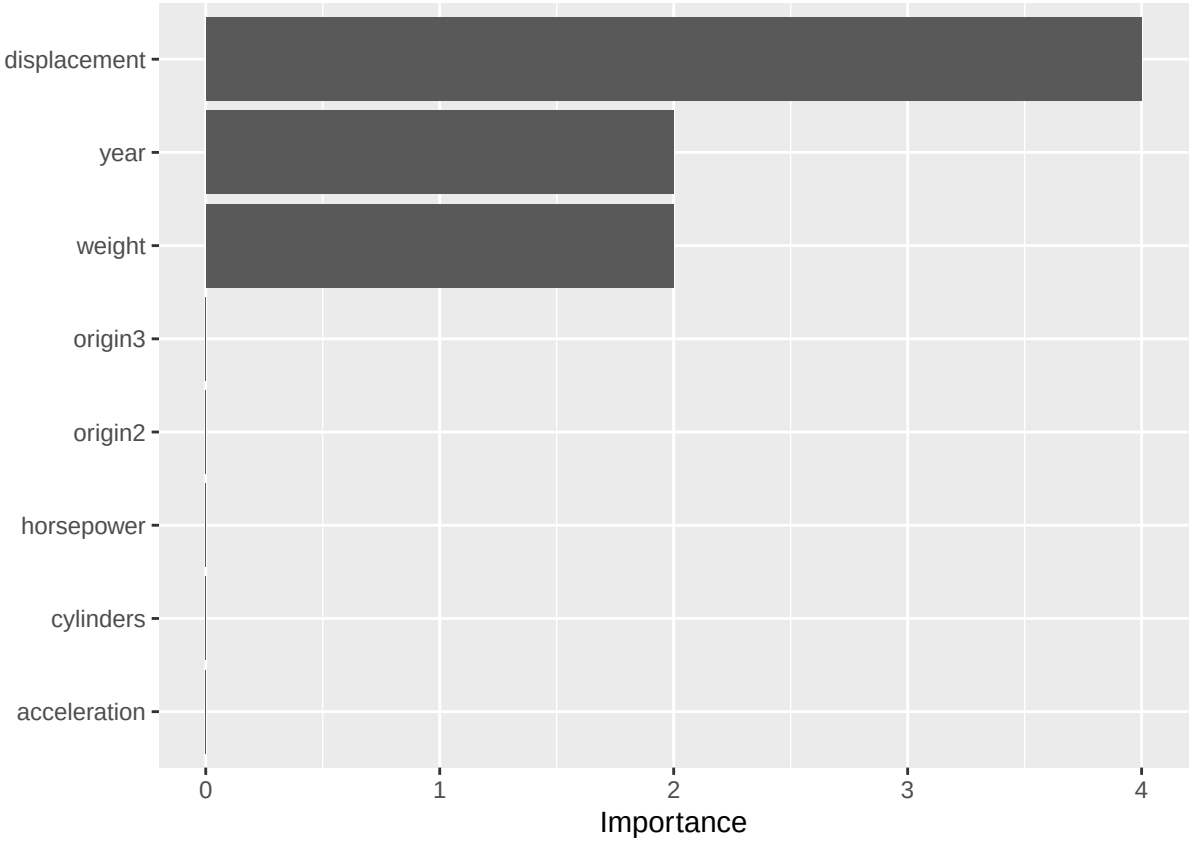
```
##          (Intercept)          h(year-72)      h(3221-weight) h(displacement-181)
##          4.474781653         -0.625384225         -0.005810691          0.231361514
## h(displacement-200)
##         -0.249849284
```

Now generate a partial dependency plot (PDP) of `weight` as well as the variable importance measurement plot (VIP), and the results can be shown that only `displacement`, `year` and `weight` contributes to the model. This result echos the coefficient significance conclusion we drawn in question (b), and the only difference is that `origin3` is not included in MARS model.

```
pdp::partial(model.mars, pred.var = c("weight"), grid.resolution = 200) %>% autoplot()
```




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



LDA

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

`lda` function in `MASS`

```
## LDA fit
set.seed(33)
lda.fit = lda(mpg_cat~., data = auto,
              subset = trainRows)

## Shows the matrix of "A"
lda.fit$scaling
```

```
##                      LD1
## cylinders     0.497962453
## displacement -0.002214541
## horsepower   -0.009257410
## weight        0.001231058
## acceleration -0.009268365
## year         -0.130838571
## origin2      -0.530237856
## origin3      -0.664110527
```

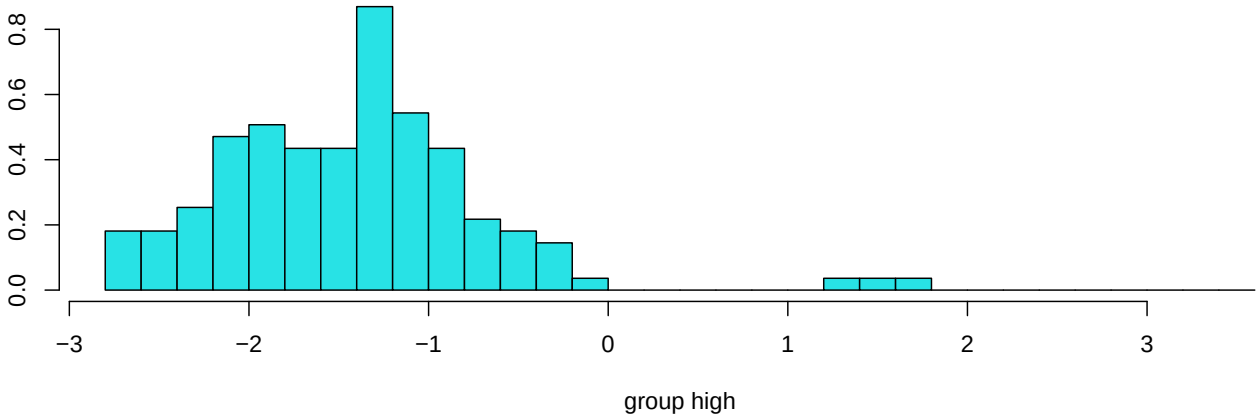
```
## More explore
head(predict(lda.fit)$x)
```

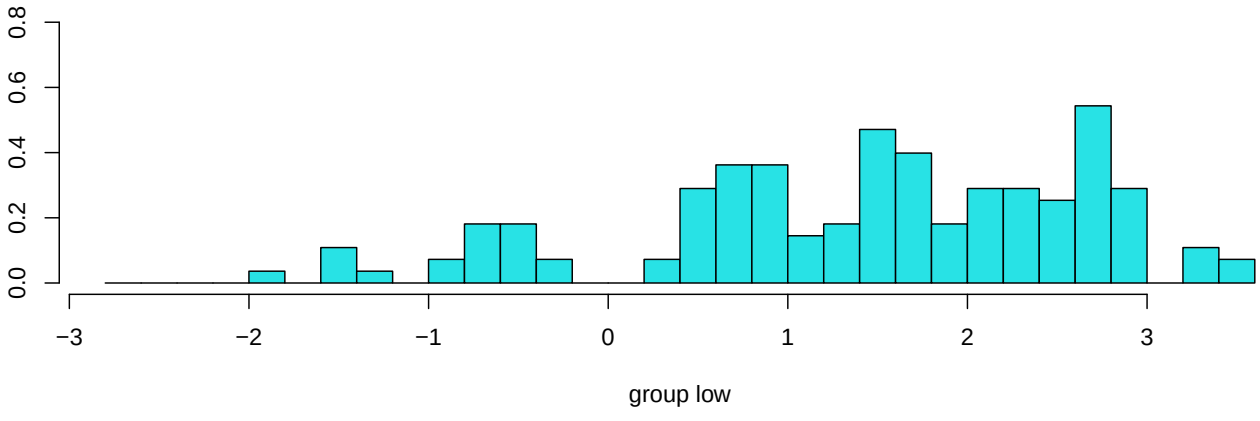
```
##         LD1
## 2  2.274620
## 4  2.190641
## 6  2.605805
## 7  2.372051
## 9  2.401686
## 10 2.175685
```

```
mean(predict(lda.fit)$x) ## seen as 0, means data is centered
```

```
## [1] 8.993119e-16
```

```
## Plot the linear discriminants in LDA
plot(lda.fit)
```





Use `caret`

For future model comparison.

```
set.seed(33)
model.lda = train(x,
                  y,
                  method = "lda",
                  metric = "ROC",
                  trControl = ctrl1)
```

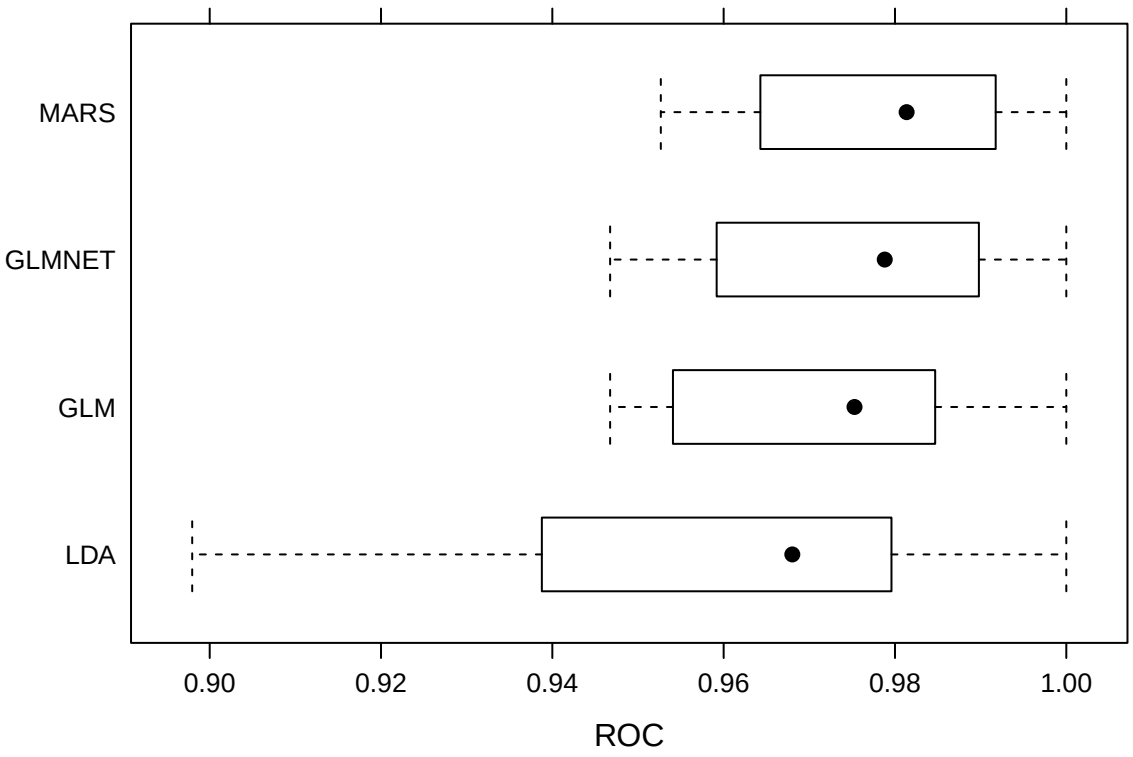
Model Comparison

(e) 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.

```
res = resamples(list(GLM = model.glm,
                     GLMNET = model.glmn,
                     MARS = model.mars,
                     LDA = model.lda))
summary(res)
```

```
##
## Call:
## summary.resamples(object = res)
##
## Models: GLM, GLMNET, MARS, LDA
## Number of resamples: 10
##
## ROC
##             Min.   1st Qu.    Median      Mean   3rd Qu. Max. NA's
## GLM    0.9467456 0.9566327 0.9752747 0.9723826 0.9846939    1    0
## GLMNET 0.9467456 0.9638932 0.9788069 0.9759932 0.9885204    1    0
## MARS   0.9526627 0.9668367 0.9813579 0.9787239 0.9899922    1    0
## LDA    0.8979592 0.9444662 0.9680141 0.9577587 0.9795918    1    0
##
## Sens
##             Min.   1st Qu.    Median      Mean   3rd Qu. Max. NA's
## GLM    0.8461538 0.8571429 0.8928571 0.9049451 0.9285714    1    0
## GLMNET 0.8461538 0.9244505 0.9285714 0.9340659 0.9821429    1    0
## MARS   0.8571429 0.9244505 0.9285714 0.9346154 0.9285714    1    0
## LDA    0.9230769 0.9285714 1.0000000 0.9708791 1.0000000    1    0
##
## Spec
##             Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## GLM    0.7857143 0.8571429 0.9258242 0.8983516 0.9285714 1.0000000    0
## GLMNET 0.7142857 0.8571429 0.9258242 0.8912088 0.9285714 1.0000000    0
## MARS   0.7142857 0.8736264 0.9285714 0.8983516 0.9285714 1.0000000    0
## LDA    0.6428571 0.8571429 0.8571429 0.8626374 0.9271978 0.9285714    0
```

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



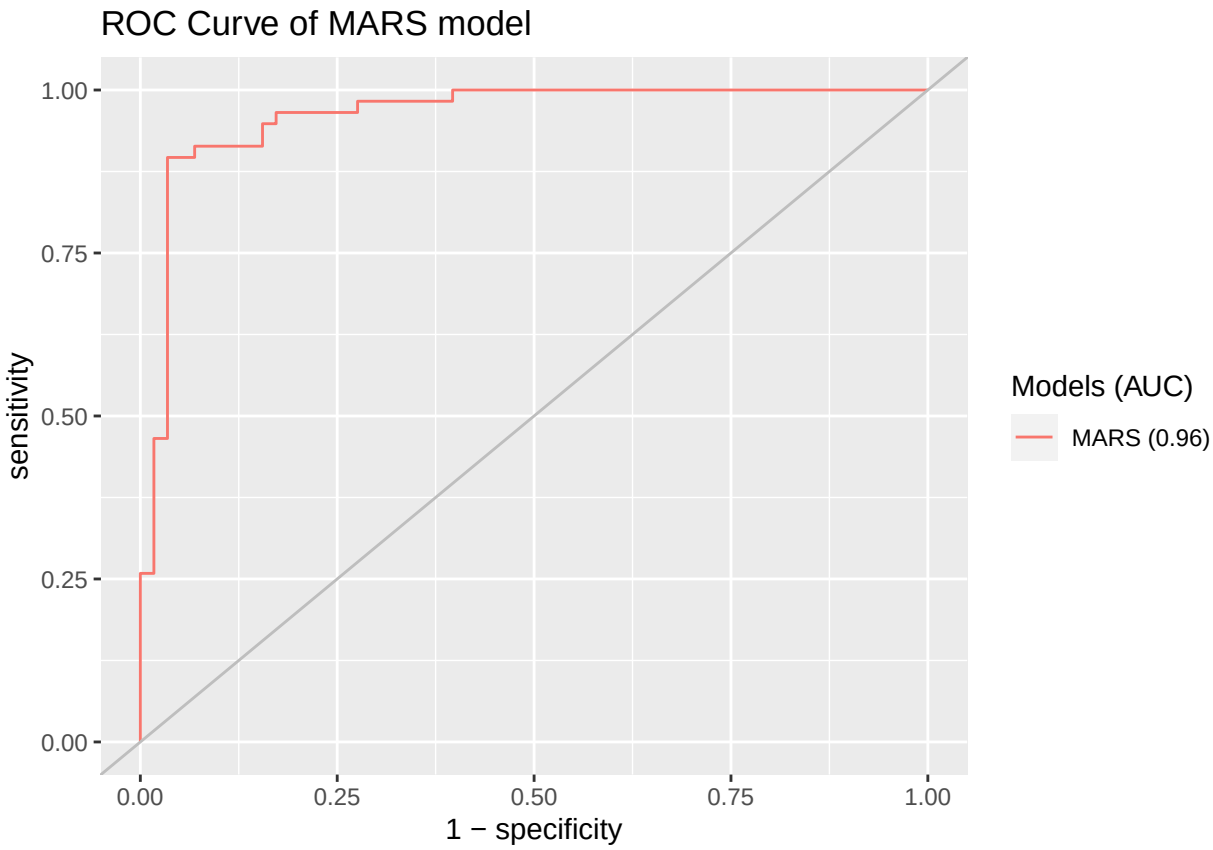
Based on the summary results and the boxplot, we prefer a model with the largest AUC (which named as "ROC" in the R output), and after comparing the AUC of 4 models, we can see that the performances of GLMNET, GLM and MARS model are quite similar, and better than LDA model. Generally, based on the mean and median of AUC, the MARS model has the best performance thus we choose **MARS model** to predict the response variable.

ROC curve

ROC curve using test data on MARS model

```
mars.pred = predict(model.mars, newdata = x_test, type = "prob")[,2]
roc.mars = roc(y_test, mars.pred)
auc = c(roc.mars$auc[1])
modelNames = c("MARS")

ggroc(list(roc.mars), legacy.axes = TRUE) +
  scale_color_discrete(labels = paste0(modelNames, " (", round(auc,3),")"),
                       name = "Models (AUC)") +
  geom_abline(intercept = 0, slope = 1, color = "grey") +
  labs(title = "ROC Curve of MARS model")
```



Report AUC and misclassification error rate

The AUC of `MARS` model is 0.96 as shown by the ROC curve above.

To get the misclassification error rate, we use confusion matrix for our previous `MARS` model and still consider 0.5 as classifier.

```
test.pred.prob.mars = predict(model.mars, newdata = x_test,
                              type = "prob")
test.pred.mars = rep("high", length(test.pred.prob))
test.pred.mars[test.pred.prob.mars[,2] > 0.5] <- "low"

confusionMatrix(data = as.factor(test.pred.mars),
                reference = auto$mpg_cat[-trainRows],
                positive = "low")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction high low
##       high   55   6
##       low     3  52
##
##                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 : 0.505
##
##             Sensitivity : 0.8966
##             Specificity : 0.9483
##          Pos Pred Value : 0.9455
##          Neg Pred Value : 0.9016
##              Prevalence : 0.5000
##          Detection Rate : 0.4483
##    Detection Prevalence : 0.4741
##       Balanced Accuracy : 0.9224
##
##        'Positive' Class : low
##
```

Given the accuracy = 0.9224, the misclassification error rate could be calculated by $1 - 0.9224 = 0.0776$, which is 7.76%.

Attachment (not the answer, just for illustration)

Here is the full ROC curve of 4 models, note that this part is just for illustration, NOT the answer of this question since the question only asked for the ROC curve for the chosen best model.

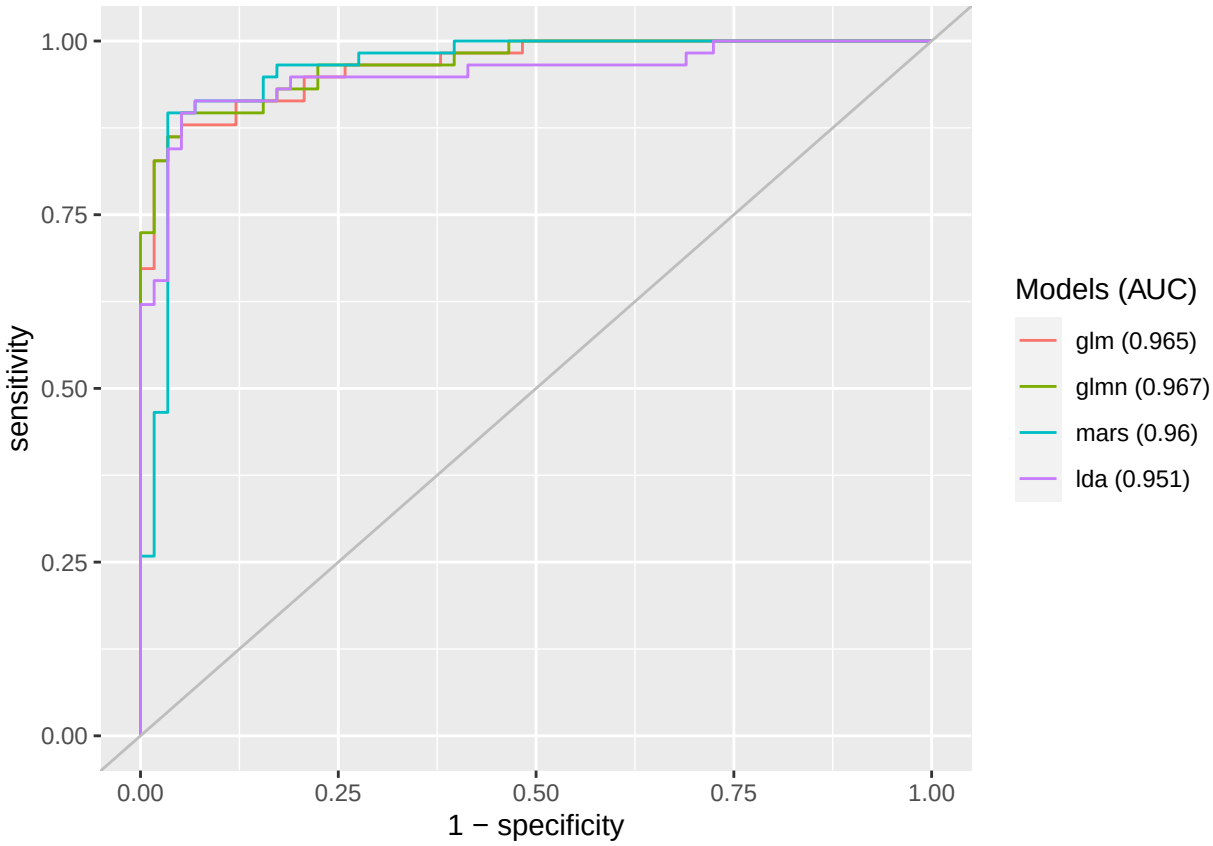
```
glm.pred <- predict(model.glm, newdata = x_test, type = "prob")[,2]
glmn.pred <- predict(model.glmn, newdata = x_test, type = "prob")[,2]
mars.pred <- predict(model.mars, newdata = x_test, type = "prob")[,2]
lda.pred <- predict(model.lda, newdata = x_test, type = "prob")[,2]

roc.glm <- roc(auto$mpg_cat[-trainRows], glm.pred)
roc.glmn <- roc(auto$mpg_cat[-trainRows], glmn.pred)
roc.mars <- roc(auto$mpg_cat[-trainRows], mars.pred)
roc.lda <- roc(auto$mpg_cat[-trainRows], lda.pred)

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

modelNames <- c("glm","glmn","mars","lda")

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


1.00
0.75
0.50
0.25
0.00
sensitivity
0.00
0.25
0.50
0.75
1.00
1 − specificity
Models (AUC)
glm (0.965)
glmn (0.967)
mars (0.96)
lda (0.951)