Homework 3

Yiying Wu (yw3996)

R packages

```
library(tidyverse)
library(caret)
library(tidymodels)
library(MASS) # for LDA and QDA
library(pROC) # ROC curve
```

Input dataset

```
dat<-read_csv("./data/auto.csv")%>%
  mutate(
    mpg_cat = as.factor(mpg_cat),
    origin = as.factor(origin))
dat <- dat%>%
  na.omit()
```

Response: mpg_cat

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

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

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

```
set.seed(1)
data_split <- initial_split(dat, prop = 0.7)

# Extract the training and test data
training_data <- training(data_split)
testing_data <- testing(data_split)

ctrl <- trainControl(method = "cv", number = 10,
                     summaryFunction = twoClassSummary,
                     classProbs = TRUE)
```

(a) Perform a logistic regression analysis using the training data. Are there redundant predictors in your model? If so, identify them. If none is present, please provide an explanation.

Use Penalized logistic regression

```
glmnGrid <- expand.grid(.alpha = seq(0, 1, length = 21),
                        .lambda = exp(seq(-8, 1, length = 50)))
set.seed(1)
model.glmn <- train(x = training_data[,1:7],
                    y = training_data$mpg_cat,
                    method = "glmnet",
                    tuneGrid = glmnGrid,
                    metric = "ROC",
                    trControl = ctrl)

model.glmn$bestTune
```

```
##     alpha      lambda
## 617   0.6 0.006337794
```

```
# if the lambda is selected at the boundary, expand the boundary.
# If alpha is 0 or 1, it's okay since the range is from [0,1]

#Coefficients
coef(model.glmn$finalModel, model.glmn$bestTune$lambda)
```

```
## 8 x 1 sparse Matrix of class "dgCMatrix"
##                          s1
## (Intercept)  10.680413233
## cylinders     0.117395142
## displacement  0.007891471
## horsepower    0.024224850
## weight        0.002590404
## acceleration  .
## year         -0.291904277
## origin       -0.073084785
```

acceleration is the redundant predictor in this model.

(b) Based on the model in (a), set a probability threshold to determine the class labels and compute the confusion matrix using the test data. Briefly interpret what the confusion matrix reveals about your model's performance.

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

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

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

```
                reference = testing_data$mpg_cat,
                positive = "low")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction high low
##       high   55   8
##       low     2  53
##
##                Accuracy : 0.9153
##                  95% CI : (0.8497, 0.9586)
##     No Information Rate : 0.5169
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.8309
##
##  Mcnemar's Test P-Value : 0.1138
##
##             Sensitivity : 0.8689
##             Specificity : 0.9649
##          Pos Pred Value : 0.9636
##          Neg Pred Value : 0.8730
##              Prevalence : 0.5169
##          Detection Rate : 0.4492
##    Detection Prevalence : 0.4661
##       Balanced Accuracy : 0.9169
##
##        'Positive' Class : low
##
```

Interpretation

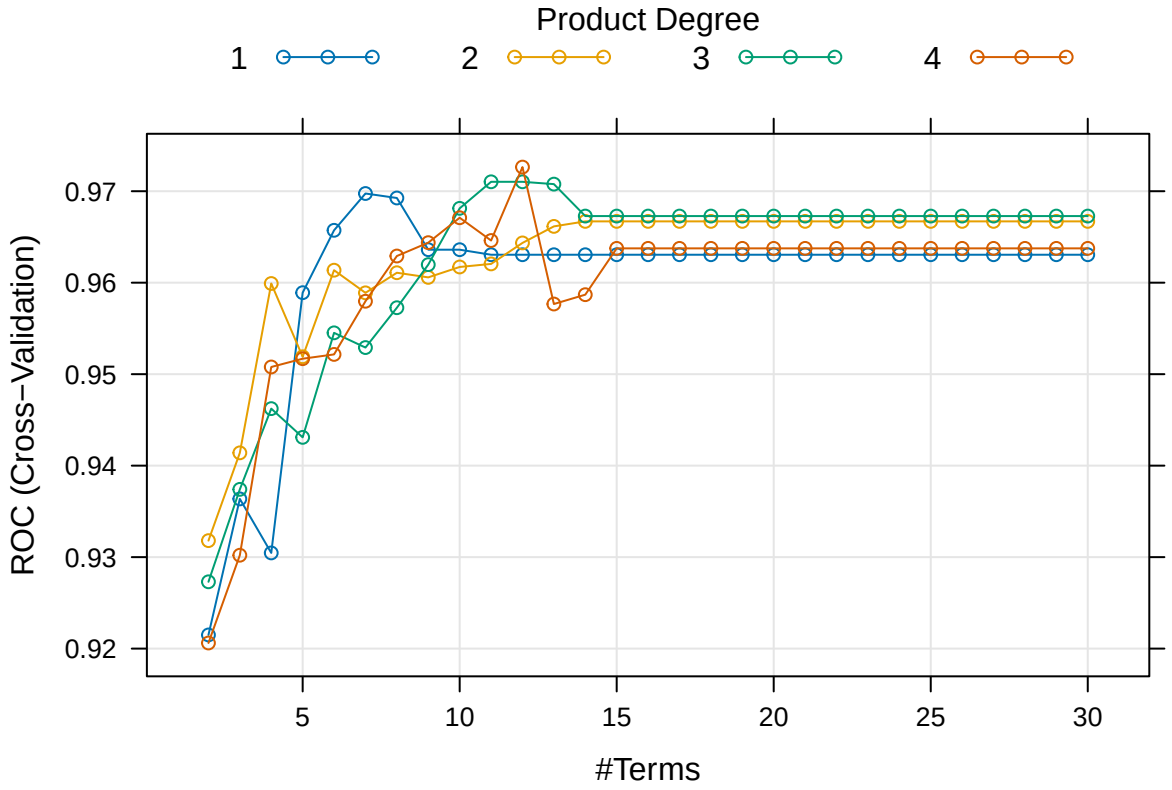
The confusion matrix and accompanying statistics reveal that the model performs well in classifying instances into "high" and "low" categories, with an overall accuracy of 91.53% (CI: 84.97% - 95.86%). The model's performance significantly surpasses the No Information Rate, indicating effective learning beyond mere chance, as evidenced by a p-value of less than 2e-16. The Cohen's Kappa score of 0.8309 further reinforces the model's strong agreement between predictions and actual values, adjusting for chance agreement. Sensitivity and specificity stand at 86.89% and 96.49%, respectively, showcasing the model's ability to accurately identify both "high" and "low" cases. Positive and Negative Predictive Values of 96.36% and 87.30% indicate high probabilities of correct predictions.

(c) Train a multivariate adaptive regression spline (MARS) model. Does the MARS model improve the prediction performance compared to logistic regression?

```
set.seed(1)
model.mars <- train(x = training_data[,1:7],
                    y = training_data$mpg_cat,
                    method = "earth", # earth is for mars
                    tuneGrid = expand.grid(degree = 1:4,
```

```
                                        # degree from 1~4 is sufficient
                                        nprune = 2:30),
                    #nprune can be larger than the number of predictors, make it as large as possible
                    metric = "ROC",
                    trControl = ctrl)

plot(model.mars)
```




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

```
##                                           (Intercept)
##                                          5.406892e+00
##                                  h(232-displacement)
##                                         -4.975136e-02
##                     h(4-cylinders) * h(232-displacement)
##                                          1.698865e-01
##                         h(155-displacement) * h(year-72)
##                                          2.334079e-02
##                     h(232-displacement) * h(weight-2670)
##                                          3.082964e-04
## h(232-displacement) * h(90-horsepower) * h(weight-2670)
##                                         -5.357612e-05
##               h(displacement-232) * h(acceleration-14.5)
##                                          5.887283e+01
##       h(displacement-232) * h(acceleration-14.5) * year
##                                         -7.457330e-01
##                         h(232-displacement) * h(year-72)
```

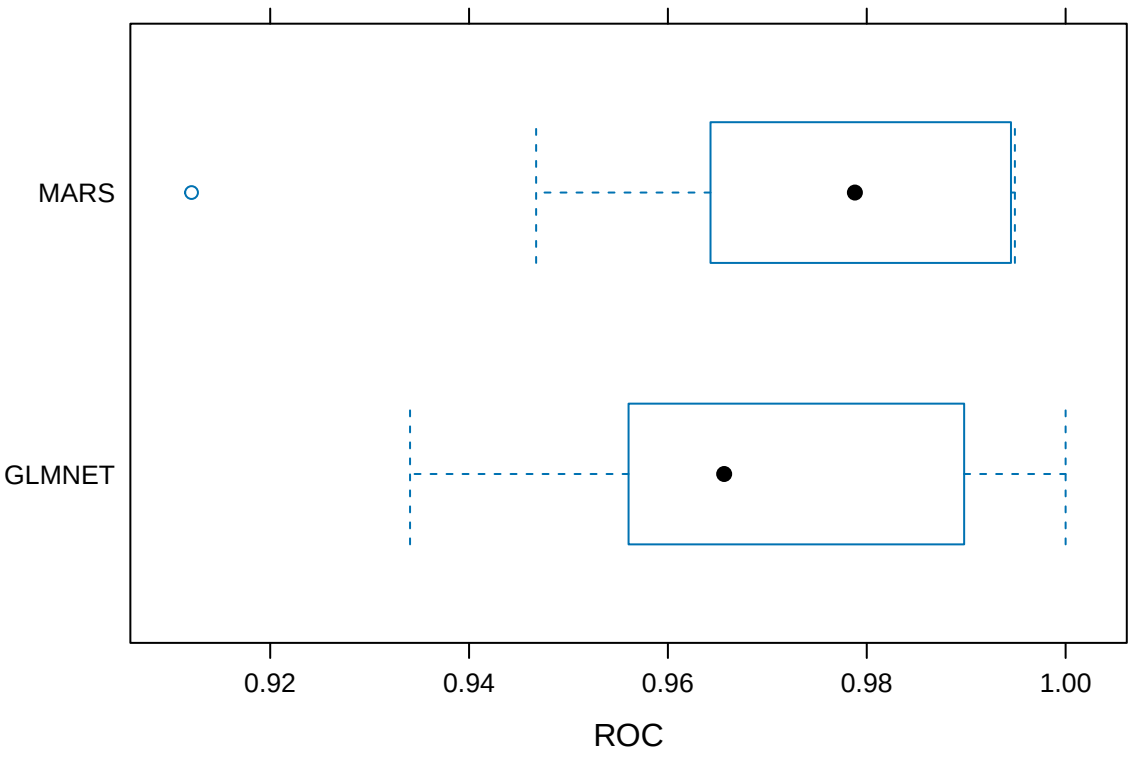
```
##                                          -1.388563e-02
##                       h(232-displacement) * h(72-year)
##                                          -3.166070e-02
```

ROC comparison

```
res <- resamples(list(GLMNET = model.glmn, MARS = model.mars))
summary(res)
```

```
##
## Call:
## summary.resamples(object = res)
##
## Models: GLMNET, MARS
## Number of resamples: 10
##
## ROC
##             Min.   1st Qu.    Median      Mean   3rd Qu.     Max. NA's
## GLMNET 0.9340659 0.9568289 0.9656593 0.9684911 0.9872449 1.000000    0
## MARS   0.9120879 0.9655612 0.9788069 0.9726377 0.9939659 0.994898    0
##
## Sens
##             Min.   1st Qu.    Median      Mean   3rd Qu. Max. NA's
## GLMNET 0.8461538 0.9285714 0.9285714 0.9346154 0.9821429    1    0
## MARS   0.9230769 0.9285714 0.9285714 0.9494505 0.9821429    1    0
##
## Spec
##             Min.   1st Qu.    Median      Mean   3rd Qu. Max. NA's
## GLMNET 0.6923077 0.8461538 0.9285714 0.8873626 0.9285714    1    0
## MARS   0.7692308 0.9230769 0.9285714 0.9104396 0.9285714    1    0
```

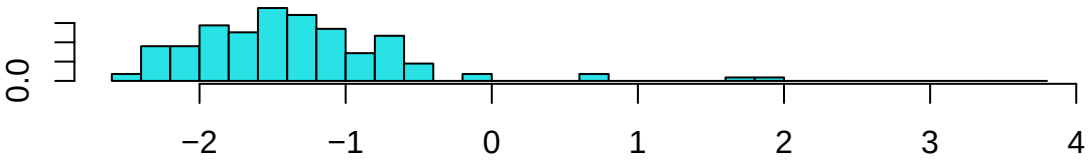
```
bwplot(res, metric = "ROC")
```



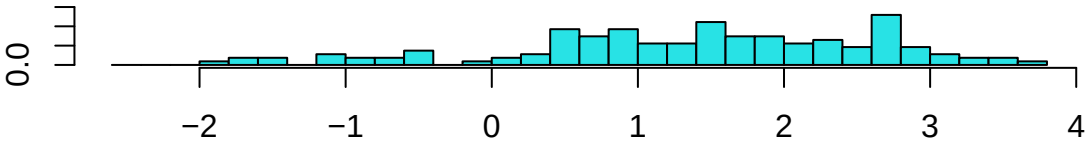
MARS shows a slightly better mean ROC value than penalized logistic regression, suggesting it might improve the prediction performance compared to logistic regression.

(d) Perform linear discriminant analysis using the training data. Plot the linear discriminant variable(s).

```
lda.fit <- lda(mpg_cat~., data = training_data)
plot(lda.fit) # histogram for z variables: the variable to do classification
```



group high



group low

```
set.seed(1)
model.lda <- train(mpg_cat~.,
                   data = training_data,
                   method = "lda",
                   metric = "ROC",
                   trControl = ctrl)
```

(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(GLMNET = model.glmn,
                      MARS = model.mars,
                      LDA = model.lda))
summary(res)
```

```
##
## Call:
## summary.resamples(object = res)
##
## Models: GLMNET, MARS, LDA
## Number of resamples: 10
##
## ROC
##             Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## GLMNET 0.9340659 0.9568289 0.9656593 0.9684911 0.9872449 1.000000    0
## MARS   0.9120879 0.9655612 0.9788069 0.9726377 0.9939659 0.994898    0
## LDA    0.8571429 0.9311224 0.9423530 0.9431409 0.9650706 1.000000    0
##
## Sens
```

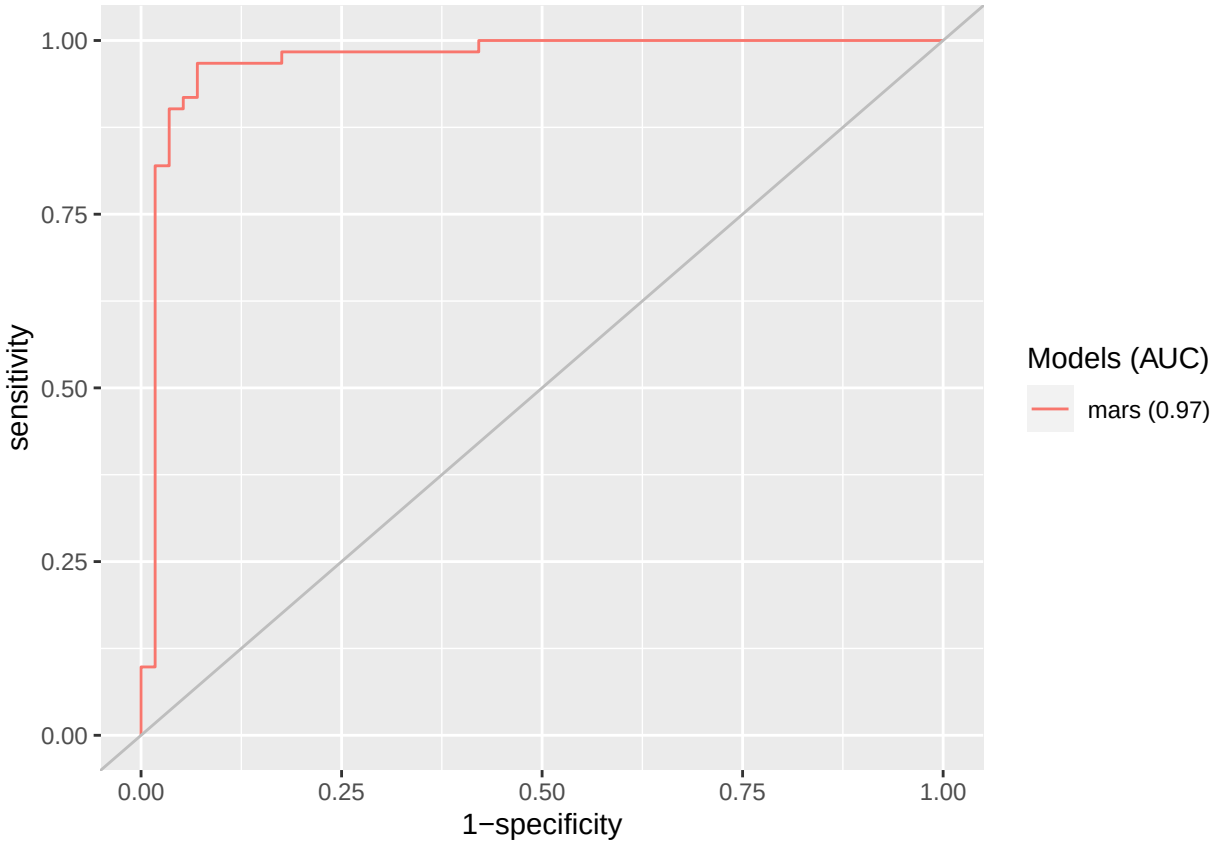
```
##             Min.   1st Qu.    Median      Mean   3rd Qu. Max. NA's
## GLMNET 0.8461538 0.9285714 0.9285714 0.9346154 0.9821429    1    0
## MARS   0.9230769 0.9285714 0.9285714 0.9494505 0.9821429    1    0
## LDA    0.8461538 0.9285714 1.0000000 0.9631868 1.0000000    1    0
##
## Spec
##             Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## GLMNET 0.6923077 0.8461538 0.9285714 0.8873626 0.9285714 1.0000000    0
## MARS   0.7692308 0.9230769 0.9285714 0.9104396 0.9285714 1.0000000    0
## LDA    0.6923077 0.8461538 0.8846154 0.8653846 0.9285714 0.9285714    0
```

MARS model will be used since it has the largest mean ROC value.

Plot the ROC curve using the test data

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

roc.mars <- roc(testing_data$mpg_cat, mars.pred)
auc <- c(roc.mars$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")
```



The **AUC** is 0.97.

confusion matrix

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

confusionMatrix(data = as.factor(test.pred),
                reference = testing_data$mpg_cat,
                positive = "low")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction high low
##       high   53   5
##       low     4  56
##
##                Accuracy : 0.9237
##                  95% CI : (0.8601, 0.9645)
##     No Information Rate : 0.5169
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.8474
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.9180
##             Specificity : 0.9298
##          Pos Pred Value : 0.9333
##          Neg Pred Value : 0.9138
##              Prevalence : 0.5169
##          Detection Rate : 0.4746
##    Detection Prevalence : 0.5085
##       Balanced Accuracy : 0.9239
##
##        'Positive' Class : low
##
```

Accuracy: 0.9237

Misclassification error rate $= 1 - Accuracy = 1 - 0.9237 = 0.0763$

The **misclassification error rate** is 7.63%.