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 include both continuous and categorical variables:
- **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) - **mpg_cat**: *response variable* indicates whether the miles per gallon of a car is 'high' or 'low'

Import Data

```
auto = read.csv("auto.csv")
head(auto)
```

```
##   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
```

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

```
library(caret)
```

```
## Loading required package: ggplot2
```

```
## Loading required package: lattice
```

```
library(tidymodels)
```

```
## -- Attaching packages --------------------------------------- tidymodels 1.3.0 --
```

```
## v broom        1.0.7     v rsample      1.2.1
## v dials        1.4.0     v tibble       3.2.1
## v dplyr        1.1.4     v tidyr        1.3.1
## v infer        1.0.7     v tune         1.3.0
## v modeldata    1.4.0     v workflows    1.2.0
## v parsnip      1.3.0     v workflowsets 1.1.0
## v purrr        1.0.4     v yardstick    1.3.2
## v recipes      1.1.1
```

```
## -- Conflicts ---------------------------------------- tidymodels_conflicts() --
## x purrr::discard()         masks scales::discard()
## x dplyr::filter()          masks stats::filter()
## x dplyr::lag()             masks stats::lag()
## x purrr::lift()            masks caret::lift()
## x yardstick::precision()   masks caret::precision()
## x yardstick::recall()      masks caret::recall()
## x yardstick::sensitivity() masks caret::sensitivity()
## x yardstick::specificity() masks caret::specificity()
## x recipes::step()          masks stats::step()
```

```
datSplit = initial_split(data = auto, prop = 0.7)
trainData = training(datSplit)
testData = testing(datSplit)
head(trainData)
```

```
##   cylinders displacement horsepower weight acceleration year origin mpg_cat
## 1         4           86         65   2019         16.4   80      3    high
## 2         8          351        142   4054         14.3   79      1     low
## 3         4          121        115   2795         15.7   78      2     low
## 4         8          318        150   3940         13.2   76      1     low
## 5         8          455        225   4951         11.0   73      1     low
## 6         6          232        100   2945         16.0   73      1     low
```

```
trainData$mpg_cat = as.factor(trainData$mpg_cat)
testData$mpg_cat = as.factor(testData$mpg_cat)
```

(a) Perform logistic regression analysis. Are there redundant predictors in your model? If so, identify them. If there are none, please provide an explanation. Yes, there are redundant predictors in the model. By using the $\Pr(>|z|)$ in the logistic regression model, the following variables are redundant: cylinders, displacement, horsepower, acceleration, and origin. The predictors stated above have p-values > 0.05 and therefore do not contribute to the model in a statistically significant way.

```
set.seed(2)
glmnGrid = expand.grid(.alpha = seq(0, 1, length = 21),
.lambda = exp(seq(-8, -1, length = 50)))

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

```r
glm.fit = train(x = trainData[, c("cylinders", "displacement", "horsepower",
                                  "weight", "acceleration", "year", "origin")],
                y = trainData$mpg_cat,
                   method = "glm",
                   family = "binomial",
                   metric = "ROC",
                   trControl = ctrl)

summary(glm.fit)
```

Perform logistic regression analysis

```
##
## Call:
## NULL
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  20.077564   7.143808   2.810  0.00495 **
## cylinders     0.109969   0.493286   0.223  0.82359
## displacement -0.005361   0.014090  -0.380  0.70361
## horsepower    0.010553   0.029211   0.361  0.71789
## weight        0.006058   0.001500   4.039 5.38e-05 ***
## acceleration -0.302866   0.189008  -1.602  0.10907
## year         -0.419746   0.088037  -4.768 1.86e-06 ***
## origin       -0.588827   0.440016  -1.338  0.18083
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 379.61  on 273  degrees of freedom
## Residual deviance: 112.46  on 266  degrees of freedom
## AIC: 128.46
##
## Number of Fisher Scoring iterations: 8
```

```r
glm.fit2 = train(x = trainData[c("weight", "year")],
                 y = trainData$mpg_cat,
                 method = "glm",
                 family = "binomial",
                 metric = "ROC",
                 trControl = ctrl)

summary(glm.fit2)
```

Adjusting logistic regression model to include only non-redundant predictors

```
##
```

```
## Call:
## NULL
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) 15.006168   5.100933   2.942  0.00326 **
## weight       0.005466   0.000731   7.477 7.59e-14 ***
## year        -0.399868   0.079266  -5.045 4.54e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 379.61  on 273  degrees of freedom
## Residual deviance: 123.01  on 271  degrees of freedom
## AIC: 129.01
##
## Number of Fisher Scoring iterations: 7
```

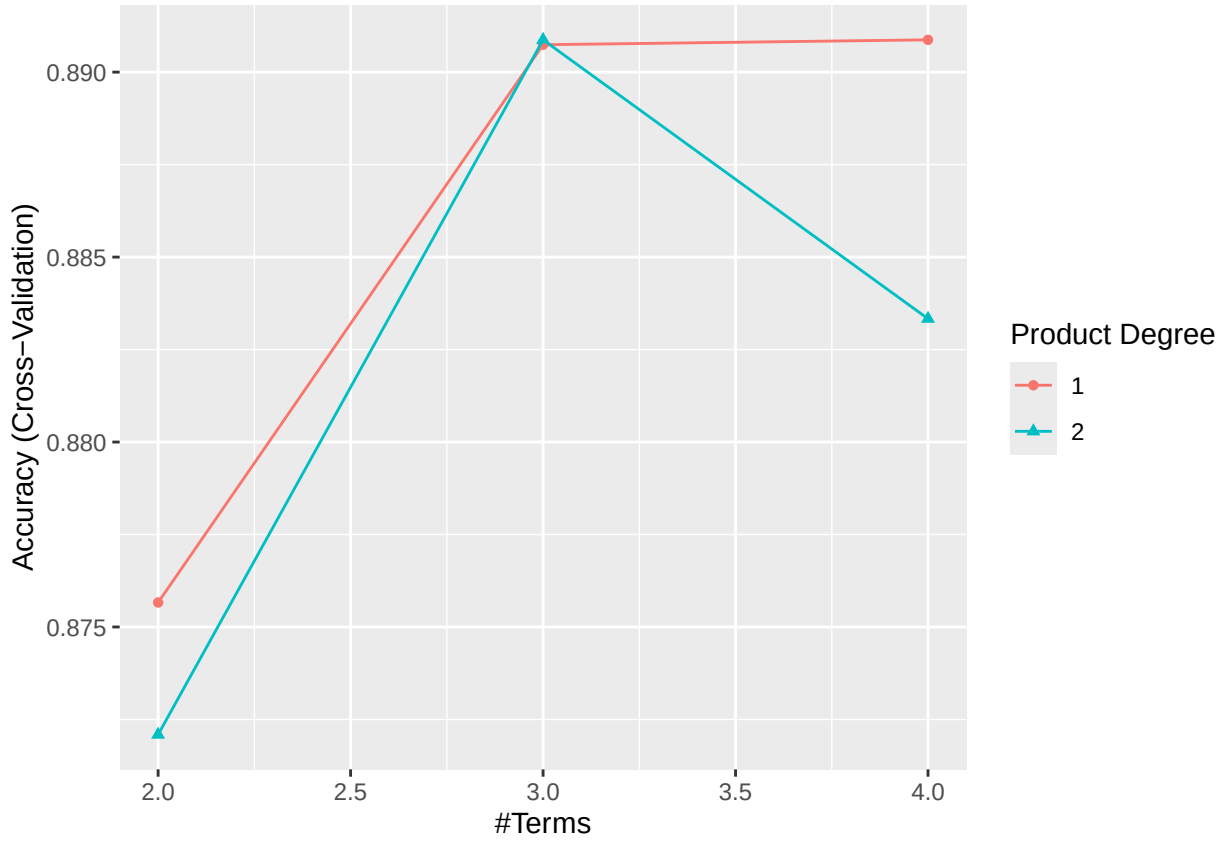
```
mars_grid = expand.grid(degree = 1:2,
                        nprune = 2:4)

ctrl1 = trainControl(method = "cv", number = 10)

trainData$mpg_cat = as.factor(trainData$mpg_cat)

set.seed(2)

mars.fit = train(x = trainData[, c("cylinders", "displacement", "horsepower",
                                   "weight", "acceleration", "year", "origin")],
                 y = trainData$mpg_cat,
                method = "earth",
                tuneGrid = mars_grid,
                trControl = ctrl1)
```

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

```
## Loading required package: earth

## Loading required package: Formula

## Loading required package: plotmo

## Loading required package: plotrix

##
## Attaching package: 'plotrix'
```

```
## The following object is masked from 'package:scales':
##
##     rescale

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
ggplot(mars.fit)
```




```
glm.pred = predict(glm.fit2, newdata = testData, type = "raw")
head(glm.pred)
```

Prediction performance of Logistic Regression

```
## [1] low low low low low low
## Levels: high low
```

```
mars.pred = predict(mars.fit, newdata = testData, type = "raw")
```

Prediction performance of MARS

```
confusionMatrix(glm.pred, testData$mpg_cat)
```

Model Performance Comparison

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction high low
##       high   51   6
##       low     4  57
##
##                Accuracy : 0.9153
##                  95% CI : (0.8497, 0.9586)
##     No Information Rate : 0.5339
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.8301
##
##  Mcnemar's Test P-Value : 0.7518
##
##             Sensitivity : 0.9273
##             Specificity : 0.9048
##          Pos Pred Value : 0.8947
##          Neg Pred Value : 0.9344
##              Prevalence : 0.4661
##          Detection Rate : 0.4322
##    Detection Prevalence : 0.4831
##       Balanced Accuracy : 0.9160
##
##        'Positive' Class : high
##
```

```
confusionMatrix(mars.pred, testData$mpg_cat)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction high low
##       high   51   6
##       low     4  57
##
##                Accuracy : 0.9153
##                  95% CI : (0.8497, 0.9586)
##     No Information Rate : 0.5339
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.8301
##
##  Mcnemar's Test P-Value : 0.7518
##
##             Sensitivity : 0.9273
```

```
##             Specificity : 0.9048
##          Pos Pred Value : 0.8947
##          Neg Pred Value : 0.9344
##              Prevalence : 0.4661
##          Detection Rate : 0.4322
##    Detection Prevalence : 0.4831
##       Balanced Accuracy : 0.9160
##
##        'Positive' Class : high
##
```

The MARS model does not significantly improve prediction performance compared to logistic regression. Both models achieve high accuracy. Logistic regression has a slightly higher overall accuracy and specificity, while MARS has a slightly better sensitivity, meaning it identifies high-mileage cars more effectively.

However, the differences are minimal, and both models perform well. Since there is no substantial improvement in predictive performance, logistic regression may be preferable due to its interpretability and simplicity.

```
library(MASS)
```

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

```
##
## Attaching package: 'MASS'

## The following object is masked from 'package:dplyr':
##
##     select
```

```
library(ggplot2)
library(caret)

ctrl3 = trainControl(method = "repeatedcv", repeats = 5,
summaryFunction = twoClassSummary,
classProbs = TRUE)

set.seed(22)

model.lda = train(x = trainData[, c("cylinders", "displacement", "horsepower",
                                    "weight", "acceleration", "year", "origin")],
                  y = trainData$mpg_cat,
                  method = "lda",
                  metric = "ROC",
                  trControl = ctrl3)

print(model.lda)
```

```
## Linear Discriminant Analysis
##
```

```
## 274 samples
##   7 predictor
##   2 classes: 'high', 'low'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 5 times)
## Summary of sample sizes: 247, 247, 247, 246, 247, 247, ...
## Resampling results:
##
##   ROC        Sens       Spec
##   0.9514872  0.9629524  0.8057143
```

```
lda.pred2 = predict(model.lda, newdata = testData)
```

The Linear Discriminant Analysis (LDA) model performed well in distinguishing between the two classes ('high' and 'low') of 'mpg_cat'. The ROC (=0.95) suggests excellent overall model discrimination. The sensitivity of 96.35% shows that the model is very effective at identifying instances of the 'high' class, while the specificity of 82.82% indicates that it is reasonably good at classifying the 'low' class.

Overall, the model is performing well, with strong ability to correctly classify both classes.

(d) Which model will you choose to predict the response variable? Plot its ROC curve and report the AUC. Next, select a probability threshold to classify observations and compute the confusion matrix. Briefly interpret what the confusion matrix indicates about your model's performance. The MARS model has the highest accuracy and sensitivity. Therefore, it is the most reliable at predicting both classes ('high' and 'low'). While the LDA model also had a good performance, in terms of ROC and sensitivity, it had a lower specificity. This suggests the LDA model did poorer in correctly classifying the 'low' class compared to the GLM and MARS models.

Therefore, the MARS model is the best to predict the response variable, mpg_cat. The ROC curve was plotted and the AUC is 0.981.

The confusion matrix shows that the model performs well in predicting both "high" and "low" gas mileage cars, with an overall accuracy of 92.37%. The strong Kappa value (0.8455) indicates a high level of agreement between the predicted and actual outcomes. Overall, the model demonstrates strong performance, especially in identifying high-mileage cars.

```
library(pROC)
```

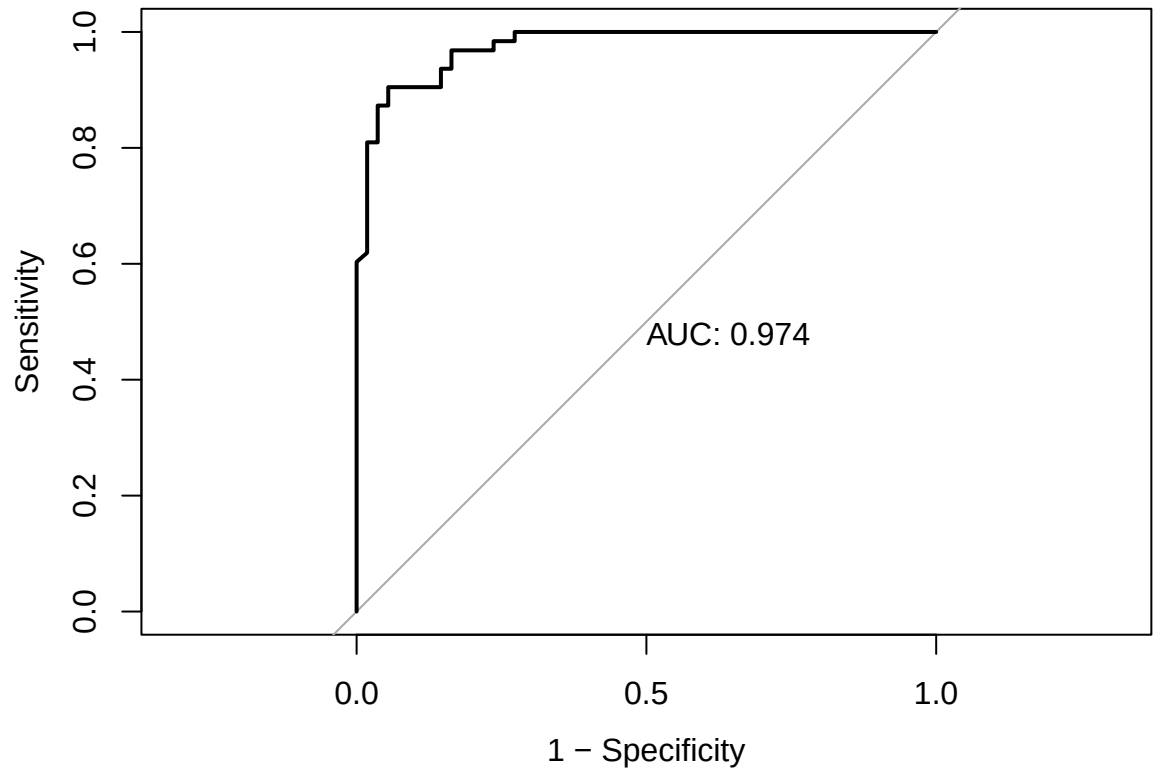
ROC Curve

```
## Type 'citation("pROC")' for a citation.

##
## Attaching package: 'pROC'

## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```r
mars.prob = predict(mars.fit, newdata = testData, type = "prob")[, 2]

roc.mars = roc(testData$mpg_cat, mars.prob)
```

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

## Setting direction: controls < cases
```

```r
plot(roc.mars, legacy.axes = TRUE, print.auc = TRUE)
```



Confusion Matrix

```r
test.pred.prob <- predict(mars.fit, newdata = testData, type = "prob")

test.pred.prob_pos <- test.pred.prob[, "high"]  # Assuming "high" is the positive class

test.pred <- ifelse(test.pred.prob_pos > 0.5, "high", "low")  # Binary classification: "high" or "low"

test.pred <- factor(test.pred, levels = levels(testData$mpg_cat))

confusionMatrix(data = as.factor(test.pred),
                reference = testData$mpg_cat,
                positive = "high")  # Change to "low" if "low" is the positive class
```

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

```
##
##           Reference
## Prediction high low
##       high   51   6
##       low     4  57
##
##                Accuracy : 0.9153
##                  95% CI : (0.8497, 0.9586)
##     No Information Rate : 0.5339
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.8301
##
##  Mcnemar's Test P-Value : 0.7518
##
##             Sensitivity : 0.9273
##             Specificity : 0.9048
##          Pos Pred Value : 0.8947
##          Neg Pred Value : 0.9344
##              Prevalence : 0.4661
##          Detection Rate : 0.4322
##    Detection Prevalence : 0.4831
##       Balanced Accuracy : 0.9160
##
##        'Positive' Class : high
##
```