P8106 - Homework 3

Joe LaRocca

April 1, 2025

Upload and Partition Data

```
set.seed(2025)

auto = read_csv("./auto.csv") |>
  mutate(mpg_cat = factor(mpg_cat))
```

```
## Rows: 392 Columns: 8
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (1): mpg_cat
## dbl (7): cylinders, displacement, horsepower, weight, acceleration, year, or...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
split = sample.split(auto$cylinders, SplitRatio = 0.7)
auto_train = subset(auto, split == TRUE)
auto_test = subset(auto, split == FALSE)
```

Part (a)

Build the Logistic Regression Model

```
auto_logr = glm(mpg_cat ~ ., family = binomial(link = "logit"), data = auto_train)
summary(auto_logr)
```

```
##
## Call:
## glm(formula = mpg_cat ~ ., family = binomial(link = "logit"),
##     data = auto_train)
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  23.085557   7.346375   3.142 0.001675 **
## cylinders     0.647099   0.592812   1.092 0.275019
## displacement -0.020080   0.016526  -1.215 0.224353
## horsepower    0.033249   0.028980   1.147 0.251249
## weight        0.006104   0.001657   3.683 0.000231 ***
## acceleration  0.003765   0.160876   0.023 0.981327
## year         -0.553806   0.111894  -4.949 7.44e-07 ***
## origin       -0.718148   0.486237  -1.477 0.139689
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 379.83  on 273  degrees of freedom
## Residual deviance: 101.61  on 266  degrees of freedom
## AIC: 117.61
##
## Number of Fisher Scoring iterations: 8
```

Evaluate Model Performance

```r
# predict probabilities for test data

auto_logr_prob <- predict(auto_logr, newdata = auto_test, type = "response")

# convert probabilities to binary predictions using a 0.5 cutoff

auto_logr_pred = rep("high", length(auto_logr_prob))
auto_logr_pred[auto_logr_prob > 0.5] = "low"

# generate confusion matrix

confusionMatrix(data = as.factor(auto_logr_pred),
                reference = auto_test$mpg_cat,
                positive = "high")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction high low
##       high   52   6
##       low     8  52
##
##                Accuracy : 0.8814
##                  95% CI : (0.809, 0.9336)
##     No Information Rate : 0.5085
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.7628
##
##  Mcnemar's Test P-Value : 0.7893
##
##             Sensitivity : 0.8667
##             Specificity : 0.8966
##          Pos Pred Value : 0.8966
##          Neg Pred Value : 0.8667
##              Prevalence : 0.5085
##          Detection Rate : 0.4407
##    Detection Prevalence : 0.4915
##       Balanced Accuracy : 0.8816
```

```
##
##        'Positive' Class : high
##
```

Create Correlation Matrix to Determine if Predictors are Redundant

```r
cor(auto_train[1:7])
```

```
##                cylinders displacement horsepower     weight acceleration
## cylinders      1.0000000    0.9518117  0.8456689  0.8923675   -0.5146215
## displacement   0.9518117    1.0000000  0.8970408  0.9263119   -0.5480396
## horsepower     0.8456689    0.8970408  1.0000000  0.8565627   -0.6866462
## weight         0.8923675    0.9263119  0.8565627  1.0000000   -0.4075962
## acceleration -0.5146215   -0.5480396 -0.6866462 -0.4075962    1.0000000
## year         -0.3898641   -0.4121672 -0.4474687 -0.3402096    0.3049474
## origin       -0.5635692   -0.6023842 -0.4451292 -0.5675503    0.2437979
##                    year     origin
## cylinders    -0.3898641 -0.5635692
## displacement -0.4121672 -0.6023842
## horsepower   -0.4474687 -0.4451292
## weight       -0.3402096 -0.5675503
## acceleration  0.3049474  0.2437979
## year          1.0000000  0.2406388
## origin        0.2406388  1.0000000
```

From the correlation matrix, we can see that cylinders, displacement, horsepower, and weight are all highly positively correlated, which makes sense given that engines with more cylinders are generally more powerful and heavier than engines with fewer cylinders. Therefore, one or more of these predictors are likely redundant.

Run Regularized Regression to Find Redundant Predictor(s)

```r
set.seed(2025)

cv_auto_lasso = cv.glmnet(x = as.matrix(auto_train[, 1:7]),
                          y = auto_train$mpg_cat,
                          alpha = 1,
                          family = "binomial")

auto_lasso = glmnet(x = auto_train[, 1:7],
                    y = auto_train$mpg_cat,
                    alpha = 1,
                    family = "binomial",
                    lambda = cv_auto_lasso$lambda.min)

coef(auto_lasso)
```

```
## 8 x 1 sparse Matrix of class "dgCMatrix"
##                        s0
## (Intercept)  17.502914631
## cylinders     0.209278931
## displacement  .
## horsepower    0.017603560
## weight        0.004004915
## acceleration  .
## year          -0.408067922
## origin        -0.238852182
```

By using L1-regularized regression (LASSO), we can see that both displacement and acceleration were eliminated through selection. Therefore, these predictors are likely redundant.

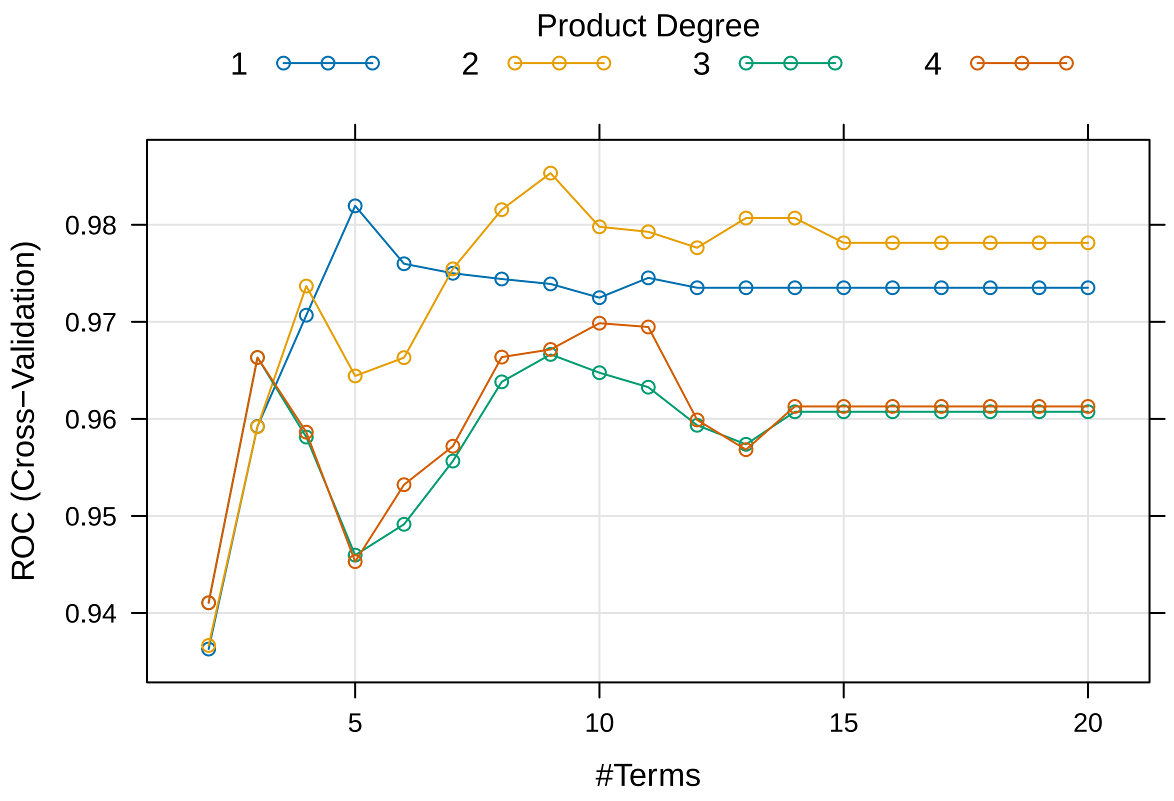
Part (b)

Build the MARS Model

```
set.seed(2025)

auto_train_mars = auto_train |>
    mutate(mpg_cat = case_match(
    mpg_cat,
    "low" ~ "No",
    "high" ~ "Yes"
    )
  )

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

mars_grid = expand.grid(
  degree = 1:4,
  nprune = 2:20
)

auto_mars = train(mpg_cat ~ .,
                  data = auto_train_mars,
                  method = "earth",
                  tuneGrid = mars_grid,
                  metric = "ROC",
                  trControl = ctrl
                  )
```

```
## 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
```

```
plot(auto_mars)
```



Evaluate Model Performance

```
# predict probabilities for test data

auto_mars_prob <- predict(auto_mars, newdata = auto_test, type = "raw")

auto_mars_pred = case_match(
  auto_mars_prob,
  "No" ~ "low",
  "Yes" ~ "high"
)

# generate confusion matrix

confusionMatrix(data = as.factor(auto_mars_pred),
                reference = auto_test$mpg_cat,
                positive = "high")
```

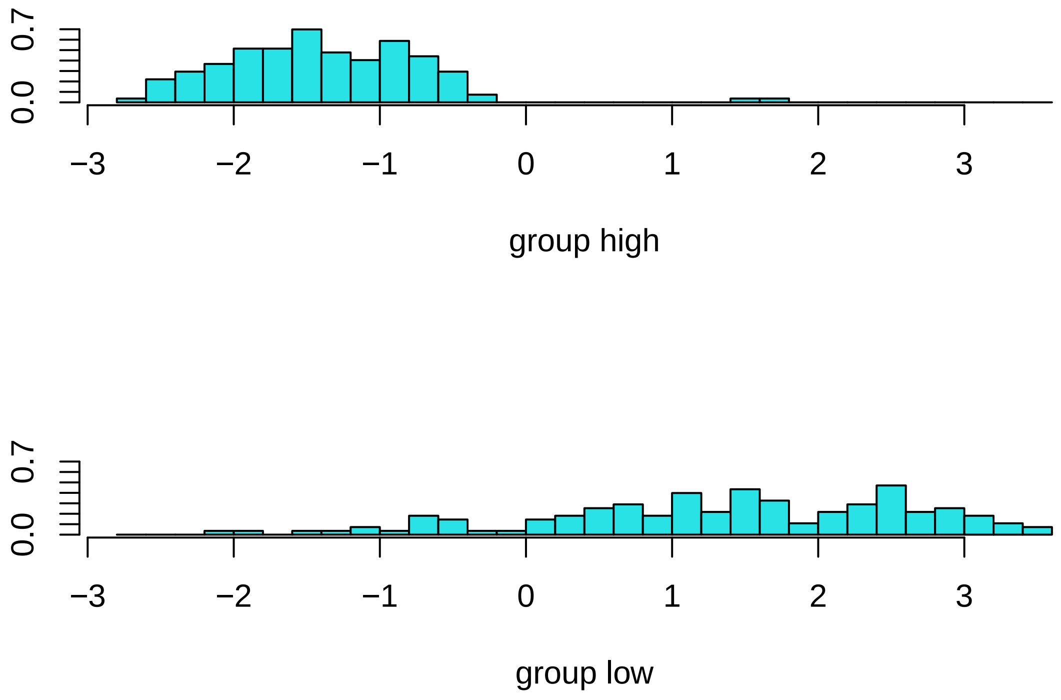
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction high low
##       high   54   6
##       low     6  52
##
##                Accuracy : 0.8983
##                  95% CI : (0.8291, 0.9463)
##     No Information Rate : 0.5085
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.7966
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.9000
##             Specificity : 0.8966
##          Pos Pred Value : 0.9000
##          Neg Pred Value : 0.8966
##              Prevalence : 0.5085
##          Detection Rate : 0.4576
##    Detection Prevalence : 0.5085
##       Balanced Accuracy : 0.8983
##
##        'Positive' Class : high
##
```

The MARS model has a higher classification accuracy (89.83%) compared to the logistic regression model (88.14%). One key reason could be the correlation between predictors explored in part (a), though the probability threshold (set to 0.5 for the logistic regression model) could also be important.

Part (c)

Plot Linear Discriminants

```r
auto_lda = lda(mpg_cat ~ ., data = auto_train)
plot(auto_lda)
```



Build LDA Model

```r
ctrl_lda <- trainControl(method = "repeatedcv", repeats = 5,
summaryFunction = twoClassSummary,
classProbs = TRUE)

# train LDA model with CV
set.seed(11)
model_lda <- train(x = auto_train[, 1:7],
                   y = auto_train$mpg_cat,
                   method = "lda",
                   metric = "ROC",
                   trControl = ctrl_lda)

auto_lda_pred_raw = predict(model_lda, newdata = auto_test, type = "prob")

auto_lda_pred = as.factor(auto_lda_pred_raw[, 1] > 0.5) |>
  case_match(
    "FALSE" ~ "low",
    "TRUE" ~ "high"
  )

confusionMatrix(data = as.factor(auto_lda_pred),
                reference = auto_test$mpg_cat,
                positive = "high")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction high low
##       high   55  10
##       low     5  48
##
##                Accuracy : 0.8729
##                  95% CI : (0.799, 0.9271)
##     No Information Rate : 0.5085
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.7453
##
##  Mcnemar's Test P-Value : 0.3017
##
##             Sensitivity : 0.9167
##             Specificity : 0.8276
##          Pos Pred Value : 0.8462
##          Neg Pred Value : 0.9057
##              Prevalence : 0.5085
##          Detection Rate : 0.4661
##    Detection Prevalence : 0.5508
##       Balanced Accuracy : 0.8721
##
##        'Positive' Class : high
##
```

From the confusion matrix, we see that the prediction accuracy of the LDA model with a probability threshold of 0.5 is about 87.21%, lower than that of both the logistic regression model with a probability threshold of 0.5 and the MARS model.

Part (d)

It turns out that for different probability thresholds (such as 0.75), the logistic regression model can outperform the MARS model. For that reason, I'm choosing the logistic regression model.

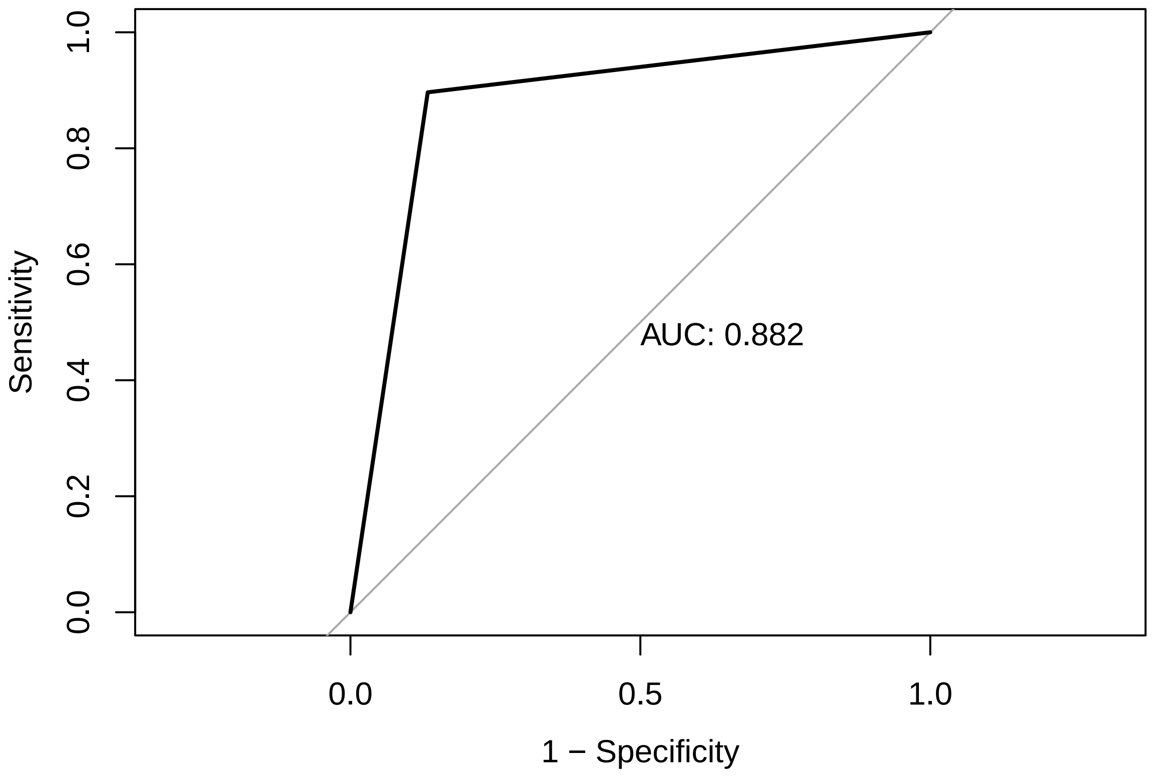
ROC Curve

```r
roc_auto_logr = roc(auto_test$mpg_cat, as.numeric(factor(auto_logr_pred)))
```

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

```
## Setting direction: controls < cases
```

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



The area under the curve (AUC) is 0.882.

Confusion Matrix

```r
# make new probability threshold of 0.75

auto_logr_pred_new = rep("high", length(auto_logr_prob))
auto_logr_pred_new[auto_logr_prob > 0.75] = "low"
```

```r
# generate confusion matrix

confusionMatrix(data = as.factor(auto_logr_pred_new),
                reference = auto_test$mpg_cat,
                positive = "high")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction high low
##       high   55   6
##       low     5  52
##
##                Accuracy : 0.9068
##                  95% CI : (0.8393, 0.9525)
##     No Information Rate : 0.5085
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.8135
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.9167
##             Specificity : 0.8966
##          Pos Pred Value : 0.9016
##          Neg Pred Value : 0.9123
##              Prevalence : 0.5085
##          Detection Rate : 0.4661
##    Detection Prevalence : 0.5169
##       Balanced Accuracy : 0.9066
##
##        'Positive' Class : high
##
```

The confusion matrix shows that the classification accuracy is about 90.68% (better than the MARS model from part (b)), with a 95% confidence interval of (83.93%, 95.25%). There are six "false positives" (i.e. model predicted high, reference was low) and five "false negatives" (i.e. model predicted low, reference was high). The sensitivity (true positive rate) of the model was about 91.67%,, while the specificity (true negative rate) of the model was about 89.66%, indicating that there were many more true positives than false negatives and true negatives than false positives, respectively.