P8106: Data Science II, Homework #3

Zachary Katz (UNI: zak2132)

3/25/2022

Contents

Set-Up and Data Preprocessing

```
set.seed(77)

# Load data, clean column names, eliminate rows containing NA entries
data = read_csv("./Data/auto.csv") %>%
  janitor::clean_names() %>%
  na.omit() %>%
  distinct() %>%
  mutate(
    cylinders = as.factor(cylinders),
    year = as.factor(year),
    origin = case_when(origin == "1" ~ "American",
                       origin == "2" ~ "European",
                       origin == "3" ~ "Japanese"),
    origin = as.factor(origin),
    mpg_cat = as.factor(mpg_cat),
    mpg_cat = fct_relevel(mpg_cat, "low")
  ) %>%
  as.data.frame()

# Partition data into training/test sets (70% split)
indexTrain = createDataPartition(y = data$mpg_cat,
                                 p = 0.7,
                                 list = FALSE)
```

Part (a): Exploratory Data Analysis

```r
# Summary statistics
summary(data)
```

```
##  cylinders  displacement     horsepower        weight      acceleration
##  3:  4     Min.   : 68.0   Min.   : 46.0   Min.   :1613   Min.   : 8.00
##  4:199     1st Qu.:105.0   1st Qu.: 75.0   1st Qu.:2225   1st Qu.:13.78
##  5:  3     Median :151.0   Median : 93.5   Median :2804   Median :15.50
##  6: 83     Mean   :194.4   Mean   :104.5   Mean   :2978   Mean   :15.54
##  8:103     3rd Qu.:275.8   3rd Qu.:126.0   3rd Qu.:3615   3rd Qu.:17.02
##            Max.   :455.0   Max.   :230.0   Max.   :5140   Max.   :24.80
##
##       year          origin    mpg_cat
##  73     : 40   American:245   low :196
##  78     : 36   European: 68   high:196
##  76     : 34   Japanese: 79
##  75     : 30
##  82     : 30
##  70     : 29
##  (Other):193
```

```r
skimr::skim_without_charts(data)
```

Table 1: Data summary

Name	data
Number of rows	392
Number of columns	8
Column type frequency:	
factor	4
numeric	4
Group variables	None

Variable type: factor

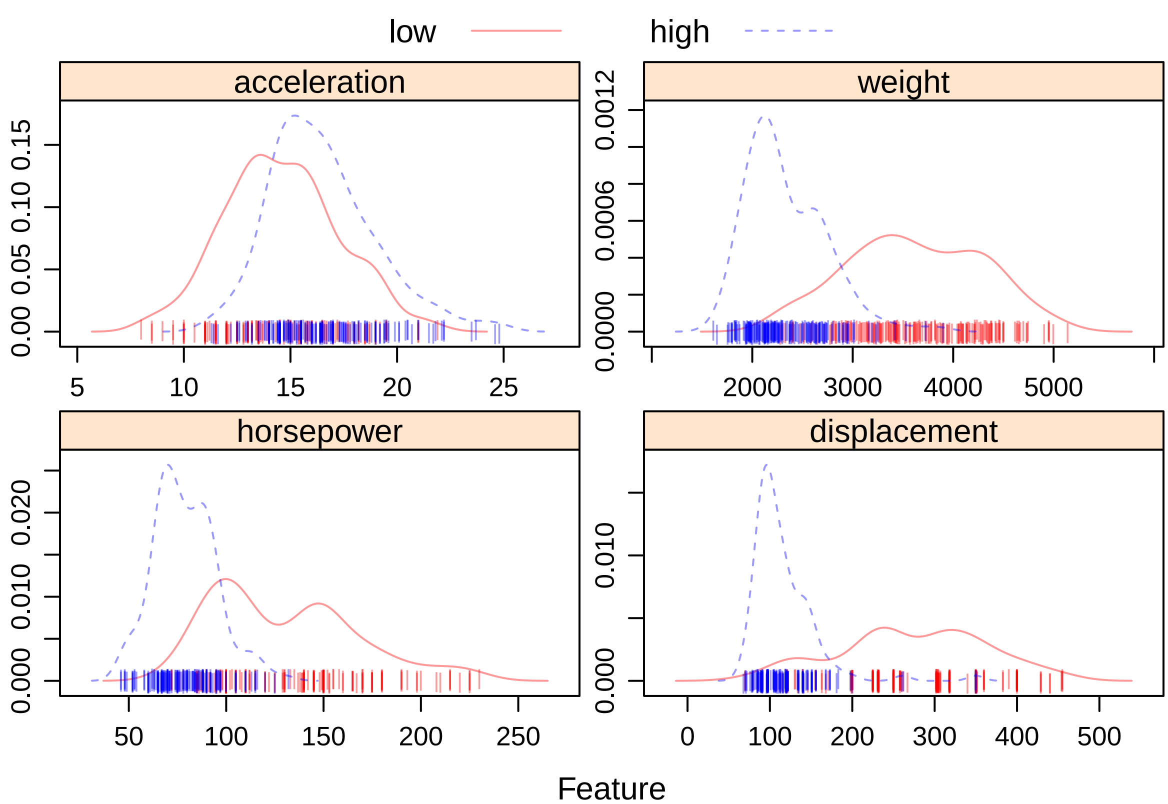
skim_variable	n_missing	complete_rate	ordered	n_unique	top_counts
cylinders	0	1	FALSE	5	4: 199, 8: 103, 6: 83, 3: 4
year	0	1	FALSE	13	73: 40, 78: 36, 76: 34, 75: 30
origin	0	1	FALSE	3	Ame: 245, Jap: 79, Eur: 68
mpg_cat	0	1	FALSE	2	low: 196, hig: 196

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100
displacement	0	1	194.41	104.64	68	105.00	151.0	275.75	455.0
horsepower	0	1	104.47	38.49	46	75.00	93.5	126.00	230.0
weight	0	1	2977.58	849.40	1613	2225.25	2803.5	3614.75	5140.0
acceleration	0	1	15.54	2.76	8	13.78	15.5	17.02	24.8

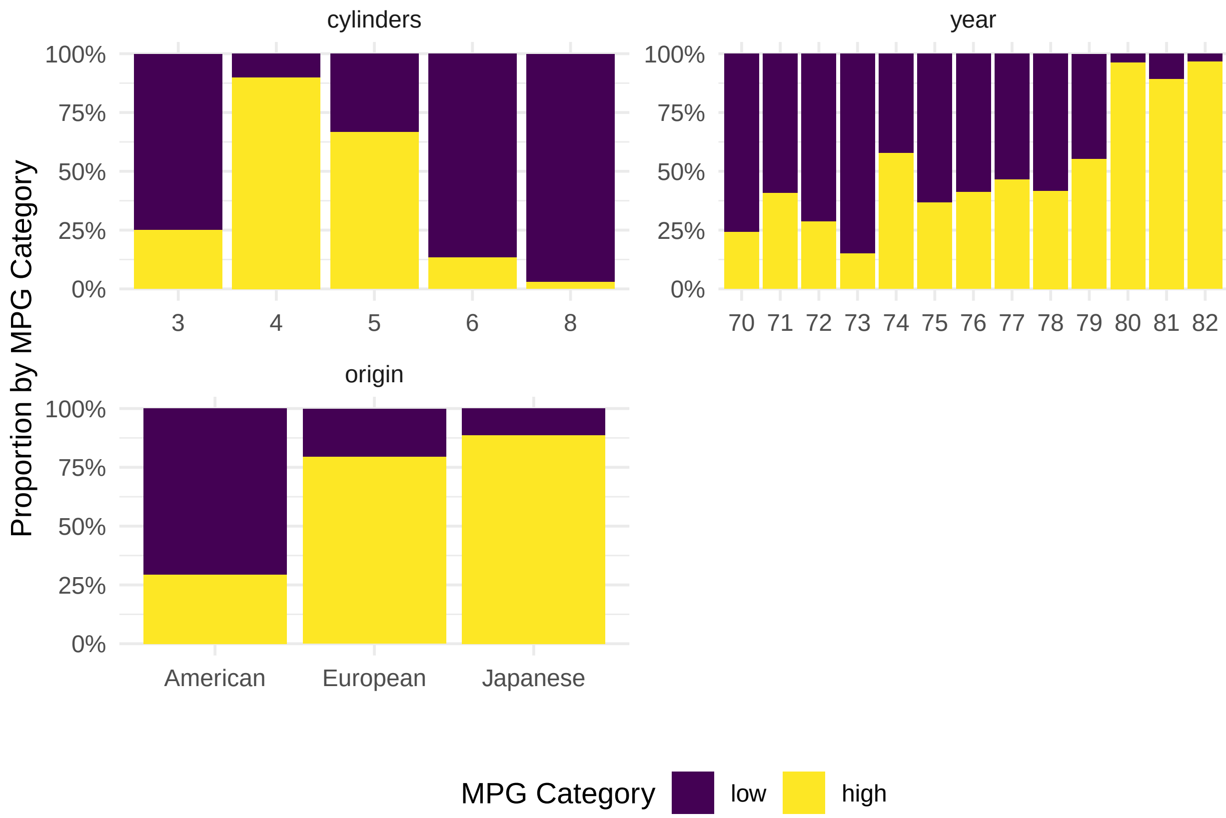
We have 392 observations with 8 parameters: 7 predictors, including 4 continuous variables (`displacement`, `horsepower`, `weight`, `acceleration`) and 3 categorical variables (`cylinders`, `year`, `origin`), along with one binary outcome variable, `mpg_cat`, which takes values "high" and "low." Half our observations have the "high" label while the other half have the "low" label.

```
# Simple visualizations of the data

# Feature plot for all data (training and test), continuous predictors only
theme1 = transparentTheme(trans = 0.4)
trellis.par.set(theme1)

featurePlot(x = data %>% dplyr::select(horsepower, displacement, acceleration, weight),
            y = data$mpg_cat,
            scales = list(x = list(relation = "free"),
                          y = list(relation = "free")),
            plot = "density", pch = "|",
            auto.key = list(columns = 2))
```

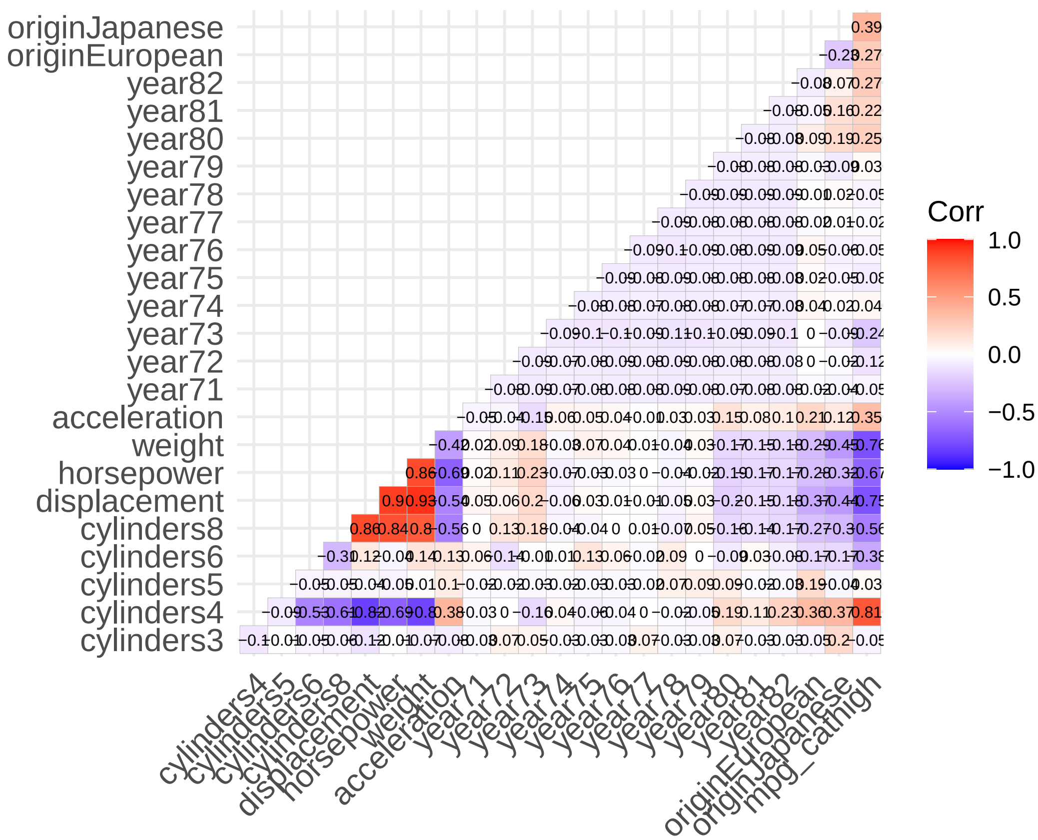


```r
# Alternative view: Distributions of continuous variables factored by outcome
data %>%
  dplyr::select(displacement, horsepower, weight, acceleration, mpg_cat) %>%
  melt(id.vars= "mpg_cat") %>%
  ggplot(aes(x = value, y = mpg_cat)) +
  stat_density_ridges(aes(color = mpg_cat, fill = mpg_cat), alpha = 0.2, quantile_lines = TRUE, quantil
  facet_wrap(~variable, scales = "free", nrow = 2) +
  labs(x = "",
        y = "Density",
       fill = "MPG Class",
       color = "MPG Class")
```

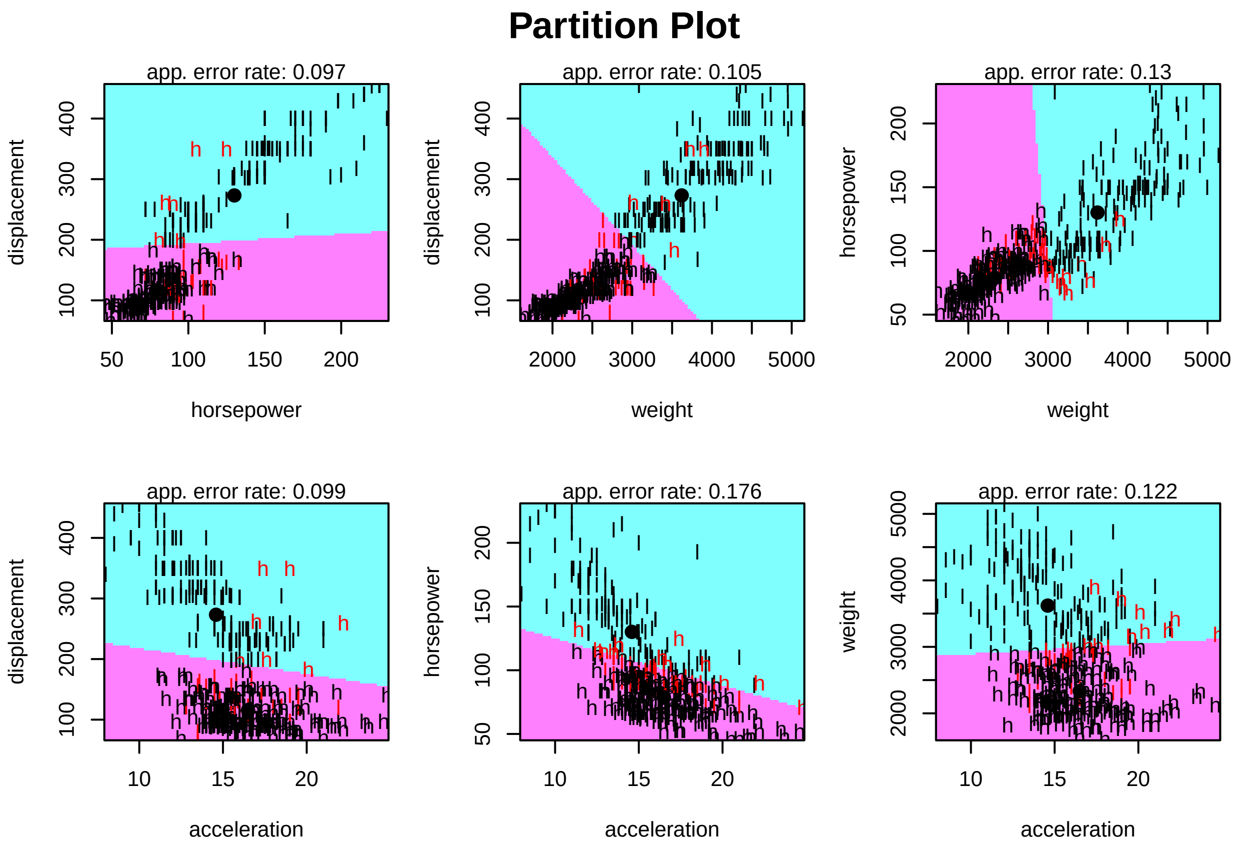

```r
# Distributions of categorical variables factored by outcome
data %>%
  dplyr::select(-displacement, -horsepower, -weight, -acceleration) %>%
  melt(id.vars = "mpg_cat") %>%
  ggplot(aes(x = value, fill = mpg_cat)) +
  geom_bar(position = "fill") +
  facet_wrap(~variable, scales = "free", nrow = 2) +
  scale_y_continuous(labels = scales::percent) +
  labs(x = "",
        y = "Proportion by MPG Category",
       fill = "MPG Category",
       color = "MPG Category")
```




```
# Correlation plot for all data
model.matrix(~0+., data = data) %>%
  cor(use = "pairwise.complete.obs") %>%
  ggcorrplot(show.diag = F, type = "lower", lab = TRUE, lab_size = 2)
```




```
# LDA partition plots (continuous vars only), all data
partimat(mpg_cat ~ displacement + horsepower + weight + acceleration, method = "lda", data = data)
```



We conduct a few basic exploratory analyses. First, our feature plot of continuous covariates shows that cars with high MPG tend to have lower displacement, lower horsepower, lower weight, and higher acceleration. Similarly, looking at categorical covariates, we find that cars with higher MPG tend to have 4 or 5 cylinders, come from the 1980s (rather than 1970s), and be European or Japanese rather than American. From the correlation plot, we see that high MPG has the most positive correlation with the indicator for having 4 cylinders, and the most negative correlation with weight, displacement, and horsepower. There may also be some collinearity between these three continuous variables, potentially leading to some redundancy in the model. Finally, from the partition plots using LDA, we see how we would partition the classes based on every combination of two variables (continuous only), giving us the decision boundary. Red points are considered misclassified. Our error rate is lowest for the following combinations of two predictors: `horsepower` and `displacement`, and `acceleration` and `displacement`. On the other hand, our error rate is highest for `acceleration` and `horsepower`. (Note that we exclude factor variables from this analysis because the decision boundaries would be somewhat misleading.)

Part (b): Logistic Regression

```
set.seed(2132)

# Logistic regression using the training data (note: not using penalized logistic regression in this ca
glm.fit = glm(mpg_cat ~ .,
              data = data,
              subset = indexTrain,
              family = binomial(link = "logit"))
```

```r
# Check for statistically significant predictors
summary(glm.fit)
```

```
##
## Call:
## glm(formula = mpg_cat ~ ., family = binomial(link = "logit"),
##     data = data, subset = indexTrain)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.20795  -0.07231  -0.00002   0.05756   3.10984
##
## Coefficients:
##                  Estimate Std. Error z value Pr(>|z|)
## (Intercept)     4.355e+00  5.798e+03   0.001  0.99940
## cylinders4      2.058e+01  5.798e+03   0.004  0.99717
## cylinders5      2.036e+01  5.798e+03   0.004  0.99720
## cylinders6      1.835e+01  5.798e+03   0.003  0.99748
## cylinders8      2.268e+01  5.798e+03   0.004  0.99688
## displacement    5.310e-03  1.938e-02   0.274  0.78407
## horsepower     -8.281e-02  4.183e-02  -1.980  0.04773 *
## weight         -4.706e-03  1.986e-03  -2.369  0.01781 *
## acceleration   -3.308e-01  2.561e-01  -1.292  0.19641
## year71         -9.044e-01  2.134e+00  -0.424  0.67175
## year72         -2.436e+00  1.456e+00  -1.673  0.09433 .
## year73         -1.755e+00  1.511e+00  -1.162  0.24537
## year74          1.735e+00  2.486e+00   0.698  0.48525
## year75          1.641e+00  1.528e+00   1.074  0.28295
## year76          1.393e+00  1.703e+00   0.818  0.41329
## year77          3.228e-01  1.575e+00   0.205  0.83764
## year78          1.984e-01  1.425e+00   0.139  0.88926
## year79          3.275e+00  1.571e+00   2.085  0.03707 *
## year80          2.079e+01  1.993e+03   0.010  0.99167
## year81          4.441e+00  1.632e+00   2.721  0.00651 **
## year82          2.107e+01  1.926e+03   0.011  0.99127
## originEuropean  1.086e+00  1.100e+00   0.987  0.32350
## originJapanese  4.251e-01  1.063e+00   0.400  0.68913
## ---
## 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:  78.041  on 253  degrees of freedom
## AIC: 124.04
##
## Number of Fisher Scoring iterations: 18
```

Here, we build a logistic regression model (without penalization) from our training data. At the 0.05 significance level, `weight`, `horsepower`, and `year79` are significant predictors of our outcome `mpg_cat`. At the 0.01 significance level, i.e. even more significantly, our indicator variable `year81` is a statistically significant predictor of our outcome as well.

```r
# Check performance on test data (use simple classifier with cut-off of 0.5)
test.pred.prob = predict(glm.fit, newdata = data[-indexTrain,],
                           type = "response")

test.pred = rep("low", length(test.pred.prob))

test.pred[test.pred.prob>0.5] = "high"

confusionMatrix(data = as.factor(test.pred),
                reference = data$mpg_cat[-indexTrain],
                positive = "high")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction low high
##       low   51    2
##       high   7   56
##
##                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 : 0.1824
##
##             Sensitivity : 0.9655
##             Specificity : 0.8793
##          Pos Pred Value : 0.8889
##          Neg Pred Value : 0.9623
##              Prevalence : 0.5000
##          Detection Rate : 0.4828
##    Detection Prevalence : 0.5431
##       Balanced Accuracy : 0.9224
##
##        'Positive' Class : high
##
```

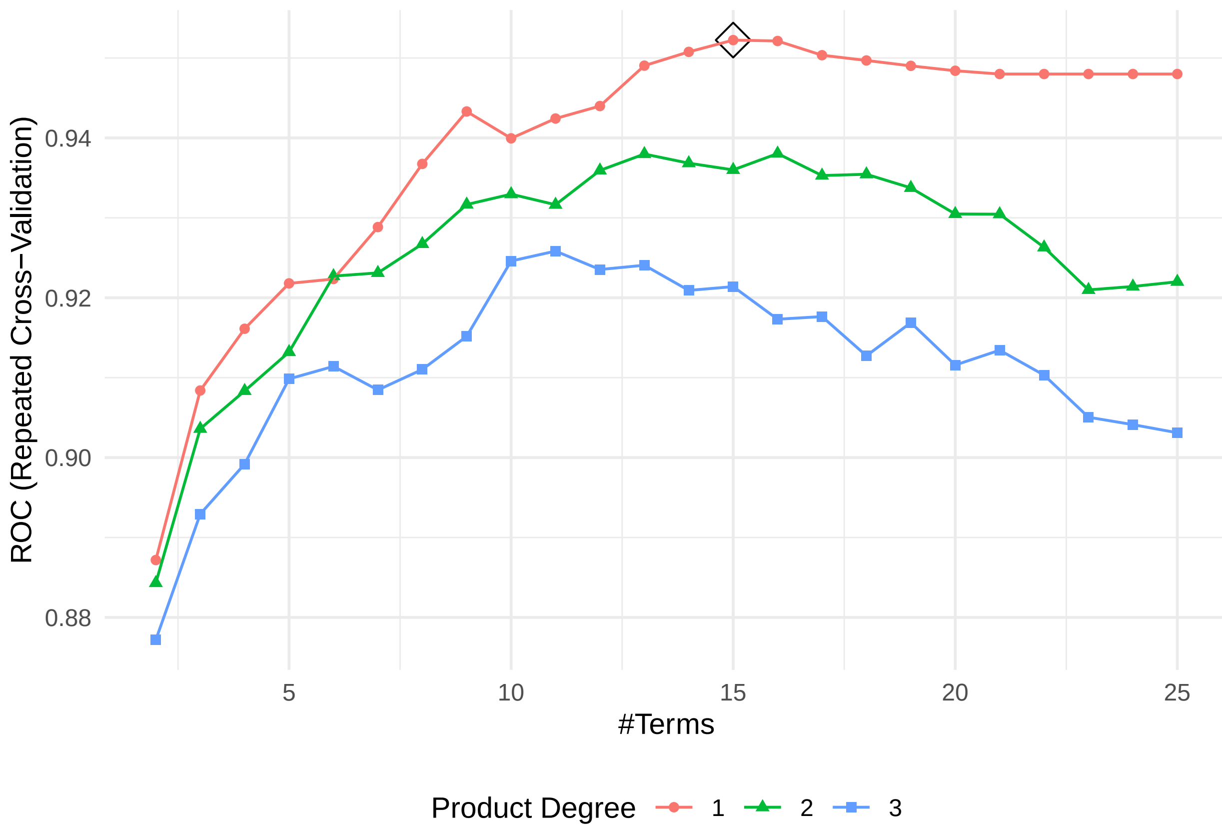
Our confusion matrix shows that our accuracy, or overall fraction of correct predictions, is roughly 92% (95% CI: 86% to 96%) once our model is applied to test data. The confusion matrix also tells us that our no information rate is 50%, which means that if we had no information and made the same class prediction for all observations, our model would be 50% accurate. Our p-value near 0 tells us that our accuracy is statistically significantly better than our no information rate. The model' is 96.7% sensitive (true detected positives out of all actual positives) and 87.9% specific (true detected negatives out of all actual negatives), with a positive predictive value of 88.9% (true detected positives out of all predicted positives) and a negative predictive value of 96.2% (true detected negatives out of all predicted negatives). Our sensitivity and specificity average to 92.2%, which is our balanced accuracy. Our kappa, at 0.8448, means that our inter-rater agreement is quite high, even accounting for the possibility of agreement by chance.

Part (c): MARS Model

```r
# Train MARS model using the training data
set.seed(2132)

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

model.mars = train(x = data[indexTrain, 1:7],
                   y = data$mpg_cat[indexTrain],
                   method = "earth",
                   tuneGrid = expand.grid(degree = 1:3,
                                          nprune = 2:25),
                   metric = "ROC",
                   trControl = ctrl)

summary(model.mars)
```

```
## Call: earth(x=data.frame[276,7], y=factor.object, keepxy=TRUE,
##             glm=list(family=function.object, maxit=100), degree=1, nprune=15)
##
## GLM coefficients
##                           high
## (Intercept)          1.4214678
## cylinders4           2.8561504
## year72              -2.7056772
## year80              18.8054702
## year81               3.6110294
## year82              20.1248516
## h(displacement-134)  0.1221785
## h(displacement-168) -0.2157792
## h(displacement-231)  0.1422861
## h(horsepower-80)    -0.4954494
## h(horsepower-86)     0.4475045
## h(weight-2634)       0.0435791
## h(weight-2700)      -0.0945420
## h(weight-2800)       0.0496376
##
## GLM (family binomial, link logit):
##  nulldev  df       dev  df   devratio     AIC iters converged
##  382.617 275   76.2185 262      0.801   104.2    18         1
##
## Earth selected 14 of 26 terms, and 8 of 22 predictors (nprune=15)
## Termination condition: Reached nk 45
## Importance: cylinders4, weight, year82, year72, year80, year81, ...
## Number of terms at each degree of interaction: 1 13 (additive model)
## Earth GCV 0.07075317    RSS 15.89408    GRSq 0.7190344    RSq 0.769651
```

```r
ggplot(model.mars, highlight = T)
```




```
model.mars$bestTune %>% knitr::kable()
```

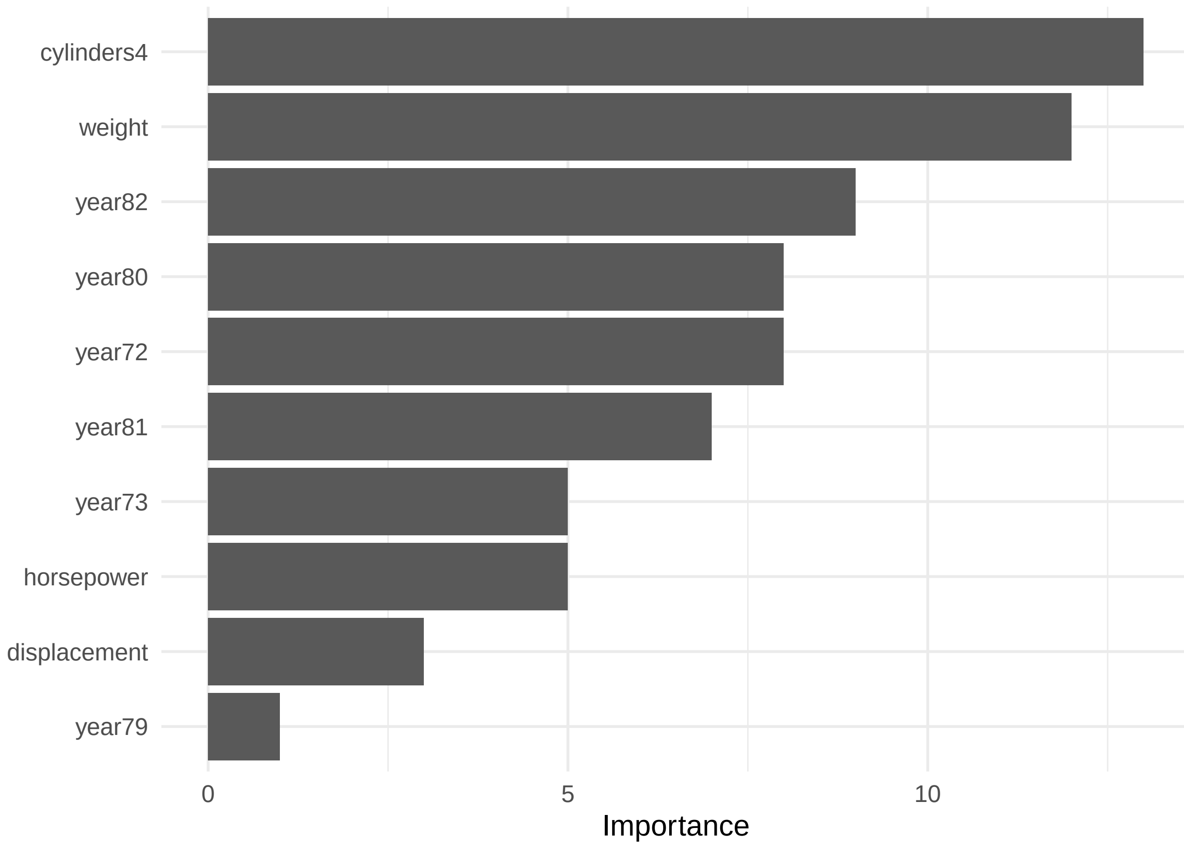
	nprune	degree
14	15	1

```
coef(model.mars$finalModel) %>% knitr::kable(col.names = "Coefficient")
```

	Coefficient
(Intercept)	1.4214678
cylinders4	2.8561504
year72	-2.7056772
year82	20.1248516
year80	18.8054702
year81	3.6110294
h(weight-2700)	-0.0945420
h(horsepower-80)	-0.4954494
h(horsepower-86)	0.4475045
h(weight-2634)	0.0435791
h(weight-2800)	0.0496376
h(displacement-168)	-0.2157792
h(displacement-231)	0.1422861

	Coefficient
h(displacement-134)	0.1221785

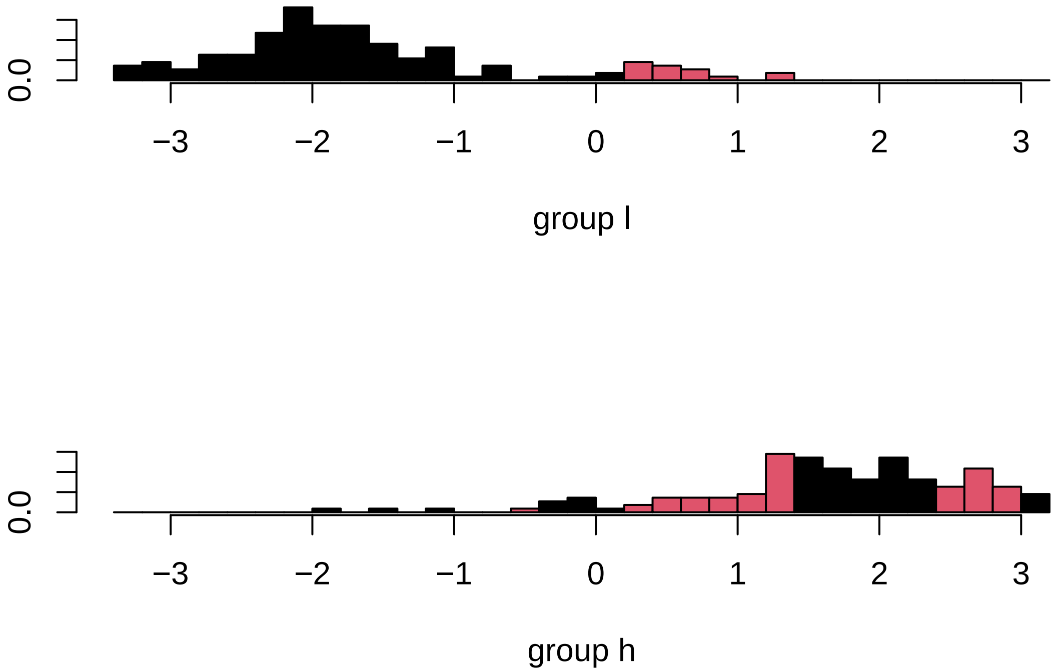
```
vip(model.mars$finalModel)
```



Overall, our MARS model tells us that `cylinders4` (indicator for having 4 cylinders) is the most important variable, with continuous variable `weight` and indicators `year82`, `year80`, and `year72` following closely behind, based on the overall impact of each variable on our regression function following a backward elimination procedure. Using `earth`, our model selects 14 out of 26 terms, representing 8 of 22 predictors (nprune terms = 15, product degree = 1). The model is optimized with and has an R-squared of 0.769.

Part (d): LDA

```
# LDA using the training data
lda.fit = lda(mpg_cat ~ ., data = data, subset = indexTrain)

# Plot the linear discriminants from LDA
plot(lda.fit, col = as.numeric(data$mpg_cat), abbrev = TRUE)
```




```r
# Obtain scaling matrix
lda.fit$scaling
```

```
##                           LD1
## cylinders4      3.1228601065
## cylinders5      2.7666662538
## cylinders6      1.0829788768
## cylinders8      1.9014478655
## displacement   -0.0001390422
## horsepower     -0.0026933834
## weight         -0.0009903955
## acceleration   -0.0415411072
## year71          0.1434316712
## year72         -0.5785741721
## year73         -0.3411186669
## year74          0.6103698170
## year75          0.5967338638
## year76          0.1863569507
## year77          0.3243683374
## year78          0.0120492352
## year79          0.8925042958
## year80          1.4036960312
## year81          1.4426797533
## year82          1.6123370953
## originEuropean  0.2038794806
```

```
## originJapanese  0.0560848244
```

LDA has no tuning parameters, and allows us to classify by nearest centroid. Because we have two classes, we have k = 2-1 = 1 linear discriminants, and so our linear discriminant plot gives us the histogram of our transformed X (predictors) for both classes. In this case, when our "X" is lower, we tend to classify in the high `mpg_cat` group, whereas when our "X" is higher, we tend to classify in the low `mpg_cat` group. Finally, the scaling object gives us our matrix A, which is (k-1) x p matrix, or in this case, a simple column vector with one entry per predictor, given we only have two outcome classes. This matrix allows us to build our x-tilde (which is AX, a product of our transformation matrix and original predictors) for each observation / data point.

```
# Alternatively, use caret for LDA
set.seed(2132)

training_df = data[indexTrain, ]

model.lda = train(mpg_cat ~ .,
                  data = training_df,
                  method = "lda",
                  metric = "ROC",
                  trControl = ctrl)

model.lda$results
```

```
##   parameter       ROC      Sens      Spec      ROCSD     SensSD     SpecSD
## 1      none 0.9668736 0.8738462 0.9043956 0.03384446 0.09061743 0.06764069
```

For completeness, we also run an LDA model using `caret`, which has a 0.958 ROC, with 84% sensitivity and 97% specificity.

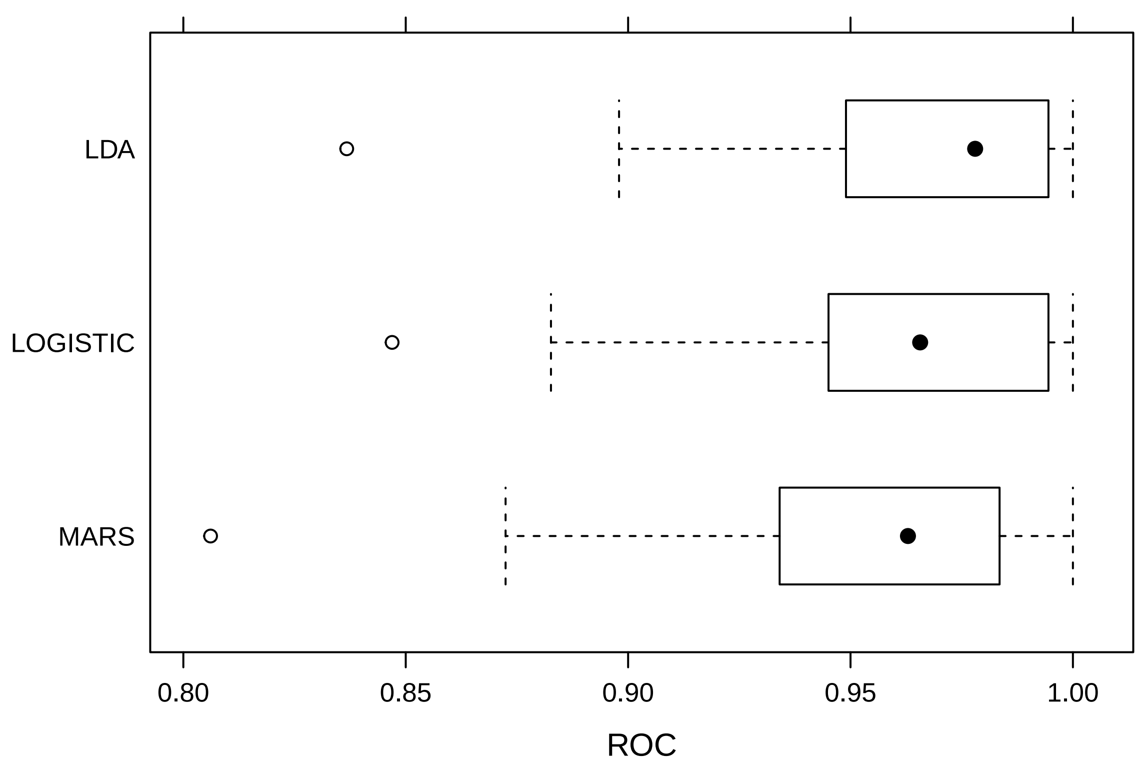
Part (e): Model Comparison and AUC/ROC

```
# Model comparison based on ROC (training data)

# Run caret logistic model
set.seed(2132)

glm.logit.caret = train(x = data[indexTrain, 1:7],
                        y = data$mpg_cat[indexTrain],
                        method = "glm",
                        metric = "ROC",
                        trControl = ctrl)

res = resamples(list(LOGISTIC = glm.logit.caret,
                     MARS = model.mars,
                     LDA = model.lda))

summary(res)
```

```
##
## Call:
```

```
## summary.resamples(object = res)
##
## Models: LOGISTIC, MARS, LDA
## Number of resamples: 50
##
## ROC
##               Min.   1st Qu.    Median      Mean   3rd Qu. Max. NA's
## LOGISTIC 0.8469388 0.9460361 0.9656593 0.9628167 0.9933281    1    0
## MARS     0.8061224 0.9340659 0.9629121 0.9522383 0.9825353    1    0
## LDA      0.8367347 0.9489796 0.9780220 0.9668736 0.9933281    1    0
##
## Sens
##               Min.   1st Qu.    Median      Mean   3rd Qu. Max. NA's
## LOGISTIC 0.7142857 0.8571429 0.9285714 0.9010989 0.9285714    1    0
## MARS     0.7142857 0.8571429 0.9285714 0.9028571 0.9285714    1    0
## LDA      0.6923077 0.7857143 0.8571429 0.8738462 0.9285714    1    0
##
## Spec
##               Min.   1st Qu.    Median      Mean   3rd Qu. Max. NA's
## LOGISTIC 0.7142857 0.8571429 0.9285714 0.9119780 0.9821429    1    0
## MARS     0.7692308 0.8571429 0.9285714 0.9097802 0.9285714    1    0
## LDA      0.7692308 0.8571429 0.9285714 0.9043956 0.9285714    1    0
```

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



Based on resampling / general cross-validation from how our models perform on the training data, having not seen the test data, I would choose the LDA model for classification of our response variable `mpg_cat`, as it has the highest ROC.

```r
# Predictions and ROC
lda.predict = predict(model.lda, newdata = data[-indexTrain, 1:7], type = "prob")[,2]

roc.lda = roc(data$mpg_cat[-indexTrain], lda.predict)

# Report AUC and misclassification rate
auc_lda = roc.lda$auc[1]

auc_lda
```

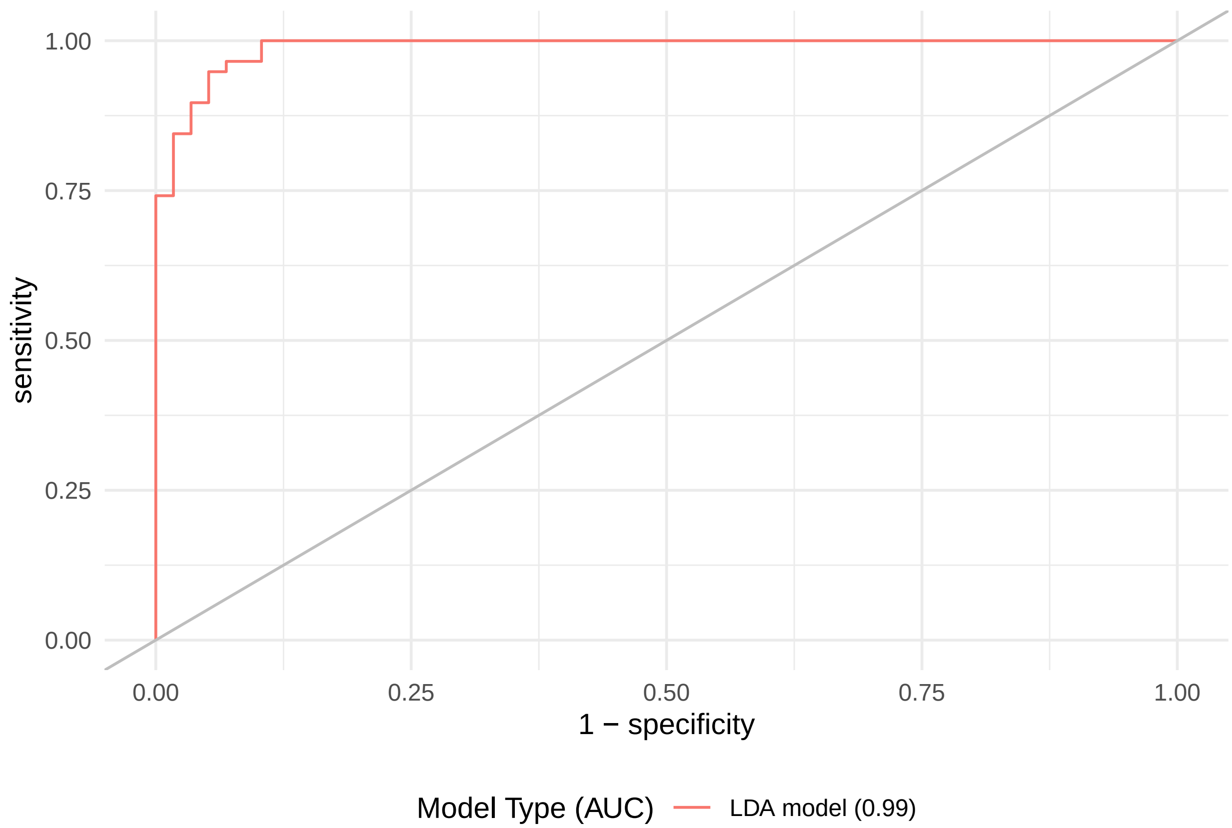
```
## [1] 0.9890012
```

```r
# Obtain classes
lda_class = lda.predict %>%
  as.data.frame() %>%
  mutate(
    class = case_when(. < 0.50 ~ "low",
                      . > 0.50 ~ "high")
  ) %>%
  dplyr::select(class) %>%
  as.matrix()

# Confusion matrix and misclassification error rate
confusionMatrix(data = as.factor(lda_class),
                reference = data$mpg_cat[-indexTrain],
                positive = "high")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction low high
##       low   54    3
##       high   4   55
##
##                Accuracy : 0.9397
##                  95% CI : (0.8796, 0.9754)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.8793
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.9483
##             Specificity : 0.9310
##          Pos Pred Value : 0.9322
##          Neg Pred Value : 0.9474
##              Prevalence : 0.5000
##          Detection Rate : 0.4741
##    Detection Prevalence : 0.5086
```

```
##        Balanced Accuracy : 0.9397
##
##         'Positive' Class : high
##
```

```r
# Plot ROC curve for best model (LDA)
modelName = "LDA model"

pROC::ggroc(list(roc.lda), legacy.axes = TRUE) +
  scale_color_discrete(labels = paste0(modelName, " (", round(auc_lda, 2),")"),
                       name = "Model Type (AUC)") +
  geom_abline(intercept = 0, slope = 1, color = "grey")
```



When applied to the previously unseen test data, the LDA model has a misclassification rate of 1 - 0.9397, or ~6%, when we use a threshold of 0.5 probability, as well as an AUC of 0.989, as observed on our ROC plot above.