# Homework 3 for Data Science II

Roxy Zhang

3/22/2022

## Contents



### Data import and cleaning

```
auto = read_csv("auto.csv") %>%
  janitor::clean_names() %>%
  mutate(
    mpg_cat = as.factor(mpg_cat),
    mpg_cat = fct_relevel(mpg_cat, "low"),
    cylinders = as.factor(cylinders),
    year = as.factor(year),
    origin = case_when(
      origin == "1" ~ "American",
      origin == "2" ~ "European",
      origin == "3" ~ "Japanese"),
    origin = as.factor(origin)
    )

# reorder columns for future visualization
col_order = c("cylinders", "year", "origin",
               "displacement", "horsepower", "weight", "acceleration", "mpg_cat")
auto = auto[ , col_order]

# check for NA
colSums(is.na(auto))
```

```
##    cylinders         year       origin displacement   horsepower       weight
```

```
##            0            0            0            0            0            0
## acceleration      mpg_cat
##            0            0
```

### Data partition

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

```r
set.seed(2570)

index_train = createDataPartition(
  y = auto$mpg_cat,
  p = 0.7,
  list = FALSE
)

train = auto[index_train, ]
test = auto[-index_train, ]

head(train)
```

```
## # A tibble: 6 x 8
##   cylinders year  origin   displacement horsepower weight acceleration mpg_cat
##   <fct>     <fct> <fct>           <dbl>      <dbl>  <dbl>        <dbl> <fct>
## 1 8         70    American          304        150   3433         12   low
## 2 8         70    American          429        198   4341         10   low
## 3 8         70    American          454        220   4354          9   low
## 4 8         70    American          390        190   3850          8.5 low
## 5 8         70    American          383        170   3563         10   low
## 6 8         70    American          340        160   3609          8   low
```

```r
# matrix of predictors
# x = model.matrix(mpg_cat~., auto)[ , -1] # remove intercept
# y = test$mpg_cat
```

### Exploratory Data Analysis

Produce some graphical or numerical summaries of the data.

```r
dim(auto)
```

```
## [1] 392   8
```

```r
summary(auto)
```

```
##  cylinders      year          origin     displacement     horsepower
##  3:  4     73     : 40   American:245   Min.   : 68.0   Min.   : 46.0
##  4:199     78     : 36   European: 68   1st Qu.:105.0   1st Qu.: 75.0
##  5:  3     76     : 34   Japanese: 79   Median :151.0   Median : 93.5
##  6: 83     75     : 30                  Mean   :194.4   Mean   :104.5
```

```
## 8:103     82     : 30                  3rd Qu.:275.8   3rd Qu.:126.0
##           70     : 29                  Max.   :455.0   Max.   :230.0
##           (Other):193
##      weight      acceleration   mpg_cat
## Min.   :1613   Min.   : 8.00   low :196
## 1st Qu.:2225   1st Qu.:13.78   high:196
## Median :2804   Median :15.50
## Mean   :2978   Mean   :15.54
## 3rd Qu.:3615   3rd Qu.:17.02
## Max.   :5140   Max.   :24.80
##
```

```
skimr::skim(auto)
```

Table 1: Data summary

| | |
|---|---|
| Name | auto |
| 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 |

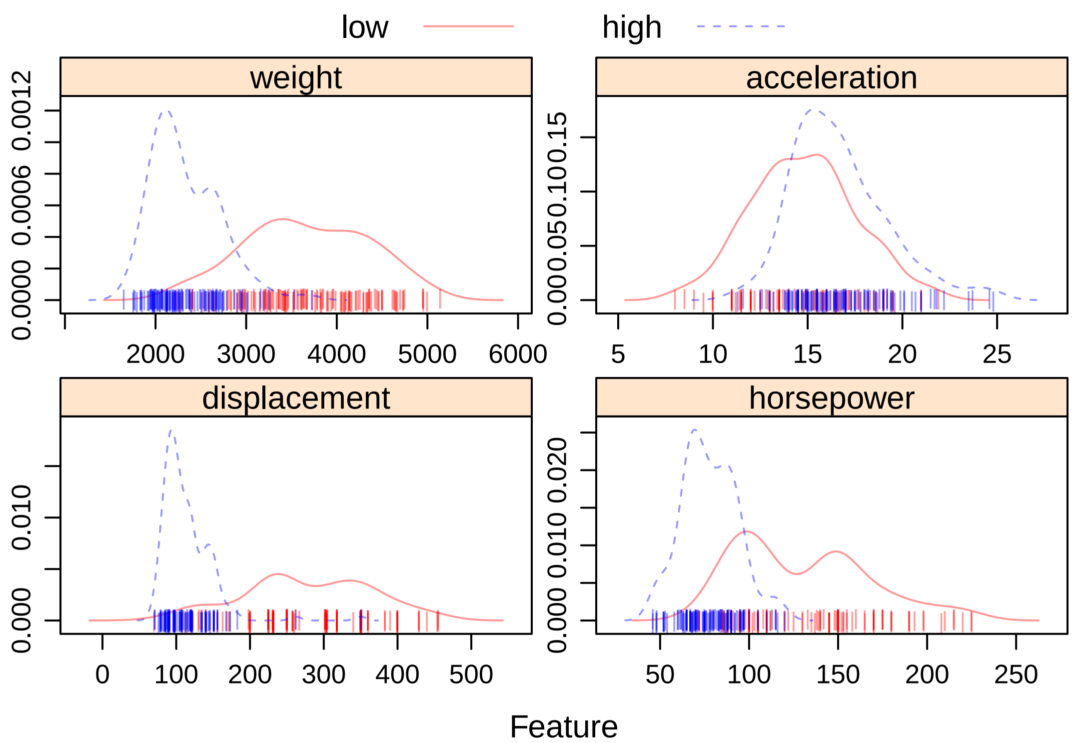
**Variable type: numeric**

| skim_variable | n_missing | complete_rate | mean | sd | p0 | p25 | p50 | p75 | p100 | hist |
|---|---|---|---|---|---|---|---|---|---|---|
| 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 | |

There are 392 rows and 8 columns in the full data, including 4 numeric predictors: `displacement`, `horsepower`, `weight`, `acceleration`, 3 categorical predictors: `cylinders`, `year`, `origin`, and 1 categorical response variable: `mpg_cat`.

For better illustration, all EDA plots are done using train data.

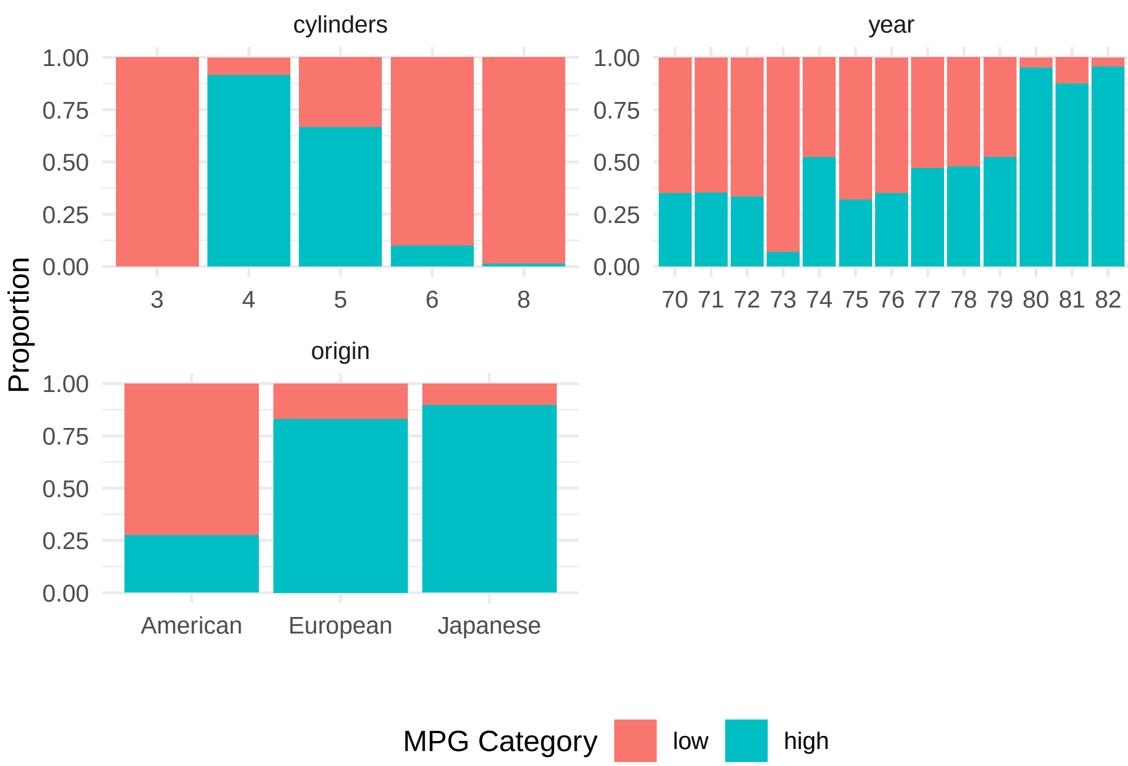
```r
# visualization for numeric variables using feature plot

# set plot theme
theme1 = transparentTheme(trans = .4)
trellis.par.set(theme1)

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




The feature plot shows that higher MPG category is associated with lower weight, higher acceleration, lower displacement and lower horsepower.

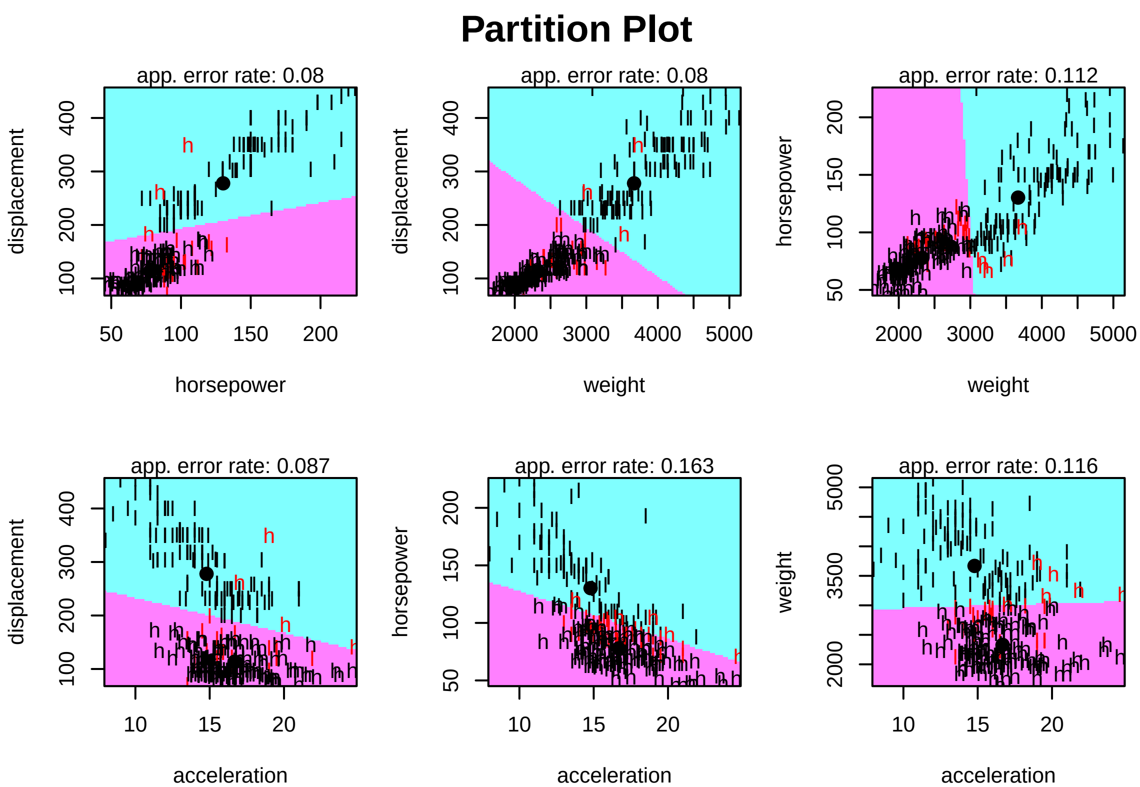
```r
# visualization for categorical variables using ggplot

train %>%
  dplyr::select(-displacement, -horsepower, -weight, -acceleration) %>%
  melt(id.vars = "mpg_cat") %>%
  ggplot(aes(x = value, fill = mpg_cat)) +
  geom_bar(position = "fill") +
```

```
  #scale_y_continuous(labels = scales::percent) + # % on y axis
  labs(x = "",
       y = "Proportion",
       fill = "MPG Category", # legend title
       color = "MPG Category") +
  facet_wrap(~variable, scales = "free", nrow = 2)
```



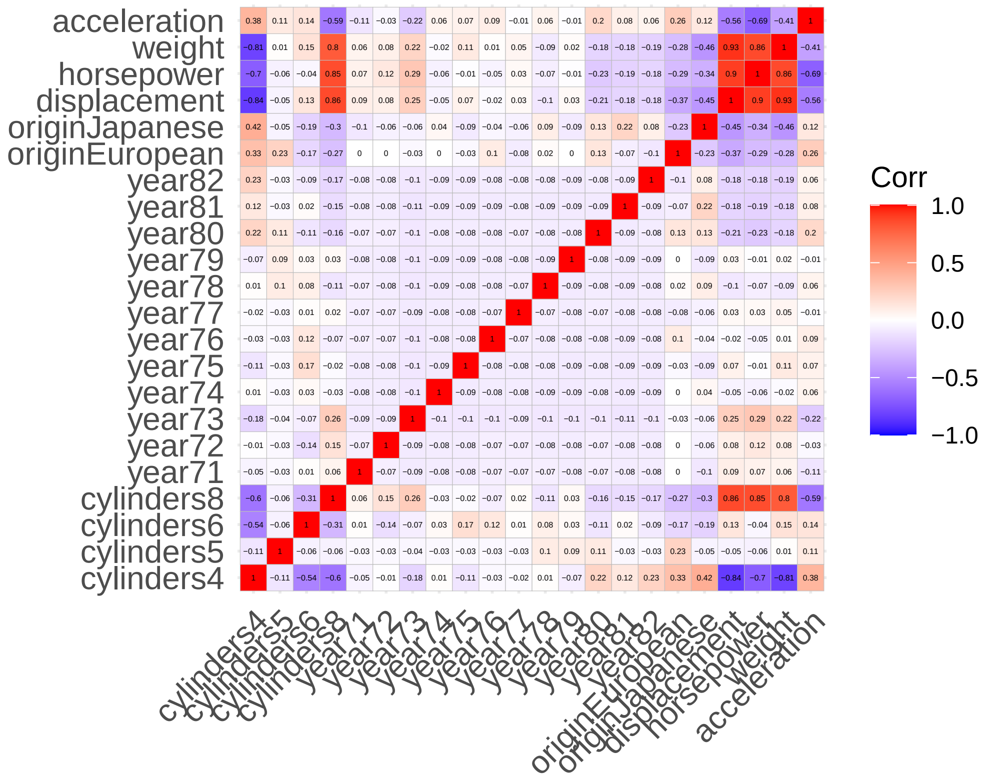

This plot shows that higher MPG category mainly lies in cars with 5 or 6 cylinders, model year 1908s, and origin of European and Japanese.

```
# LDA partition plot for numeric variables
partimat(
  mpg_cat ~ displacement + horsepower + weight + acceleration,
  data = auto,
  subset = index_train,
  method = "lda")
```




The LDA partition plot is based on every combination of two numeric variables, which gives the decision boundrary of making classification.
Red labels are misclassified data.
Although in LDA we use all the predictors rather than just the combination of two predictors, this plot shows some potential patterns of the data (since we cannot visualize things easily in high-dimensional space).

```
# correlation plot for all data
model.matrix(mpg_cat~., data = train)[ , -1] %>%
  cor(use = "pairwise.complete.obs") %>%
  ggcorrplot(type = "full", lab = TRUE, lab_size = 1)
```




We can see from the correlation plot that the numeric predictors `displacement`, `horsepower`, `weight`, `acceleration` are highly correlated, which may potentially result in some redundancy for model building. Also, `cylinders8` is highly correlated with above numeric predictors.

## Logistic Regression

```
set.seed(1115)

# check for the response variable level
contrasts(auto$mpg_cat)
```

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

```
# fit glm model
glm_fit = glm(
  mpg_cat ~ .,
  data = auto,
  subset = index_train,
  family = binomial(link = "logit"))

summary(glm_fit)
```

```
##
## Call:
## glm(formula = mpg_cat ~ ., family = binomial(link = "logit"),
##     data = auto, subset = index_train)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
##  -2.30    0.00    0.00    0.00    1.67
##
## Coefficients:
##                  Estimate Std. Error z value Pr(>|z|)
## (Intercept)     2.423e+01  7.972e+04   0.000   0.9998
## cylinders4      2.317e+01  7.946e+04   0.000   0.9998
## cylinders5      2.519e+00  7.955e+04   0.000   1.0000
## cylinders6     -2.501e+01  7.966e+04   0.000   0.9997
## cylinders8     -2.000e+01  7.966e+04   0.000   0.9998
## year71         -1.921e+01  6.385e+03  -0.003   0.9976
## year72         -1.931e+01  6.385e+03  -0.003   0.9976
## year73         -2.263e+01  6.385e+03  -0.004   0.9972
## year74          9.045e+00  5.417e+04   0.000   0.9999
## year75          8.786e+00  1.614e+04   0.001   0.9996
## year76         -1.718e+01  6.385e+03  -0.003   0.9979
## year77          8.554e+00  5.831e+04   0.000   0.9999
## year78         -1.662e+01  6.385e+03  -0.003   0.9979
## year79          3.156e+01  5.827e+03   0.005   0.9957
## year80          2.158e+01  6.248e+03   0.003   0.9972
## year81          3.523e+01  5.827e+03   0.006   0.9952
## year82          3.181e+01  5.827e+03   0.005   0.9956
## originEuropean  5.892e+00  3.039e+00   1.939   0.0525 .
## originJapanese  1.174e+00  1.748e+00   0.672   0.5018
## displacement    3.201e-02  3.989e-02   0.802   0.4223
## horsepower      1.243e-02  6.743e-02   0.184   0.8537
## weight         -1.226e-02  4.788e-03  -2.561   0.0104 *
## acceleration   -2.713e-01  3.883e-01  -0.699   0.4848
## ---
## 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:  29.956  on 253  degrees of freedom
## AIC: 75.956
##
## Number of Fisher Scoring iterations: 22
```

From the summary above, we can see that for the logistic regression model, `weight` and `originEuropean` are **statistically significant predictors** under 0.05 significance level, and `weight` is significant under 0.01 significance level.

```r
# test model performance
test_pred_prob = predict(
  glm_fit,
  newdata = test,
  type = "response") # get predicted probabilities
```

```r
test_pred = rep("low", length(test_pred_prob))

# use a simple classifier with a cut-off of 0.5
test_pred[test_pred_prob>0.5] = "high"

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

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction low high
##       low   52    8
##       high   6   50
##
##                Accuracy : 0.8793
##                  95% CI : (0.8058, 0.9324)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.7586
##
##  Mcnemar's Test P-Value : 0.7893
##
##             Sensitivity : 0.8621
##             Specificity : 0.8966
##          Pos Pred Value : 0.8929
##          Neg Pred Value : 0.8667
##              Prevalence : 0.5000
##          Detection Rate : 0.4310
##    Detection Prevalence : 0.4828
##       Balanced Accuracy : 0.8793
##
##        'Positive' Class : high
##
```

- As the confusion matrix shows above, there are 52 true low MPG category and 50 true high MPG category, with a prediction accuracy of 0.8793.

- The No Information Rate is 0.5, which means if we have no information at all and predict all the MPG category to be low (or high), the prediction accuracy will be 0.5.

- The p-value is approximately 0, showing that the fitted model is significantly better than the one generates no information rate.

- Sensitivity is 0.8621, which is the rate of predicting MPG category as high given the true value is high. Specificity is 0.8966, which is the rate of predicting MPG category as low given the true value is low.

- Positive predictive value is 0.8929, which is the rate of a true high value given the predicted value is high. Negative predictive value is 0.8929, which is the rate of a true low value given the predicted value is low.

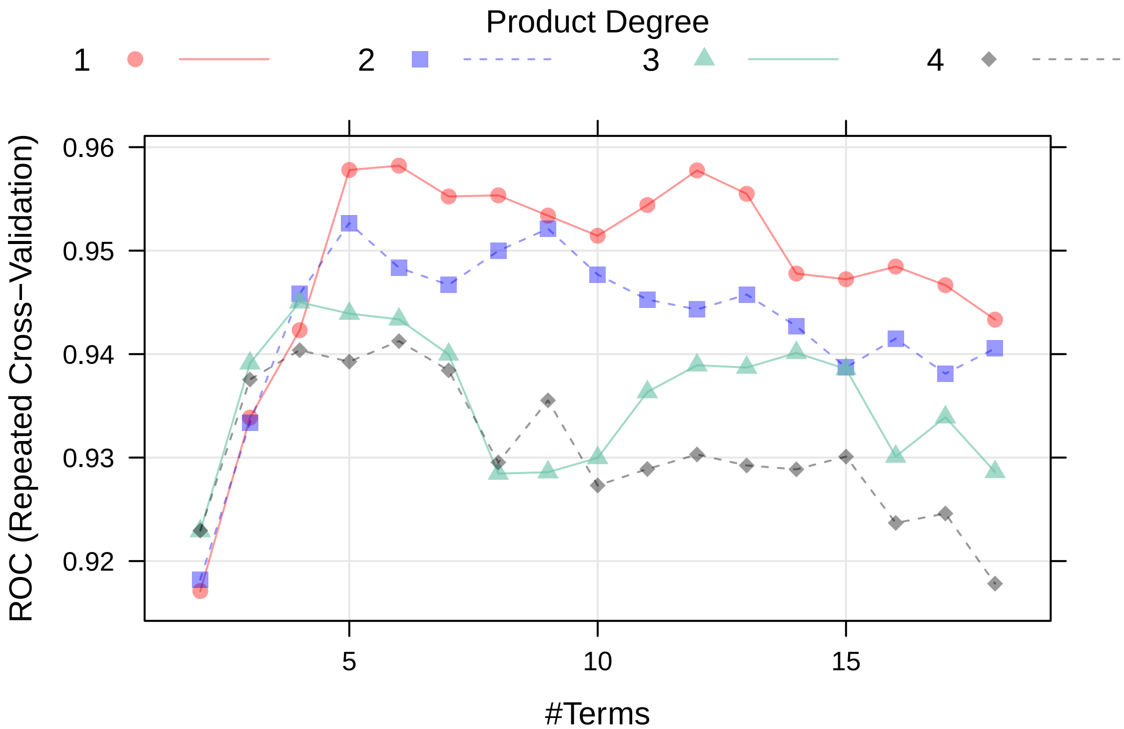
- Kappa is 0.7586, which means the agreement of observations and predictions is relatively high.

### MARS (multivariate adaptive regression spline) model

```
set.seed(1115)

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

mars_fit = train(
  x = train[ , 1:7],
  y = train$mpg_cat,
  method = "earth",
  tuneGrid = expand.grid(degree = 1:4,
                         nprune = 2:18),
  metric = "ROC",
  trControl = ctrl)

summary(mars_fit)
```

```
## Call: earth(x=tbl_df[276,7], y=factor.object, keepxy=TRUE,
##             glm=list(family=function.object, maxit=100), degree=1, nprune=6)
##
## GLM coefficients
##                            high
## (Intercept)          -0.2371402
## cylinders4            3.1419082
## year73               -3.8513130
## h(displacement-163)   0.1322963
## h(displacement-183) -0.5229625
## h(displacement-200)   0.3938448
##
## GLM (family binomial, link logit):
##  nulldev  df       dev  df   devratio     AIC iters converged
##  382.617 275    98.004 270      0.744     110     7         1
##
## Earth selected 6 of 27 terms, and 3 of 22 predictors (nprune=6)
## Termination condition: Reached nk 45
## Importance: cylinders4, year73, displacement, cylinders5-unused, ...
## Number of terms at each degree of interaction: 1 5 (additive model)
## Earth GCV 0.05760174    RSS 14.6561    GRSq 0.7712596    RSq 0.7875928
```

```
plot(mars_fit)
```



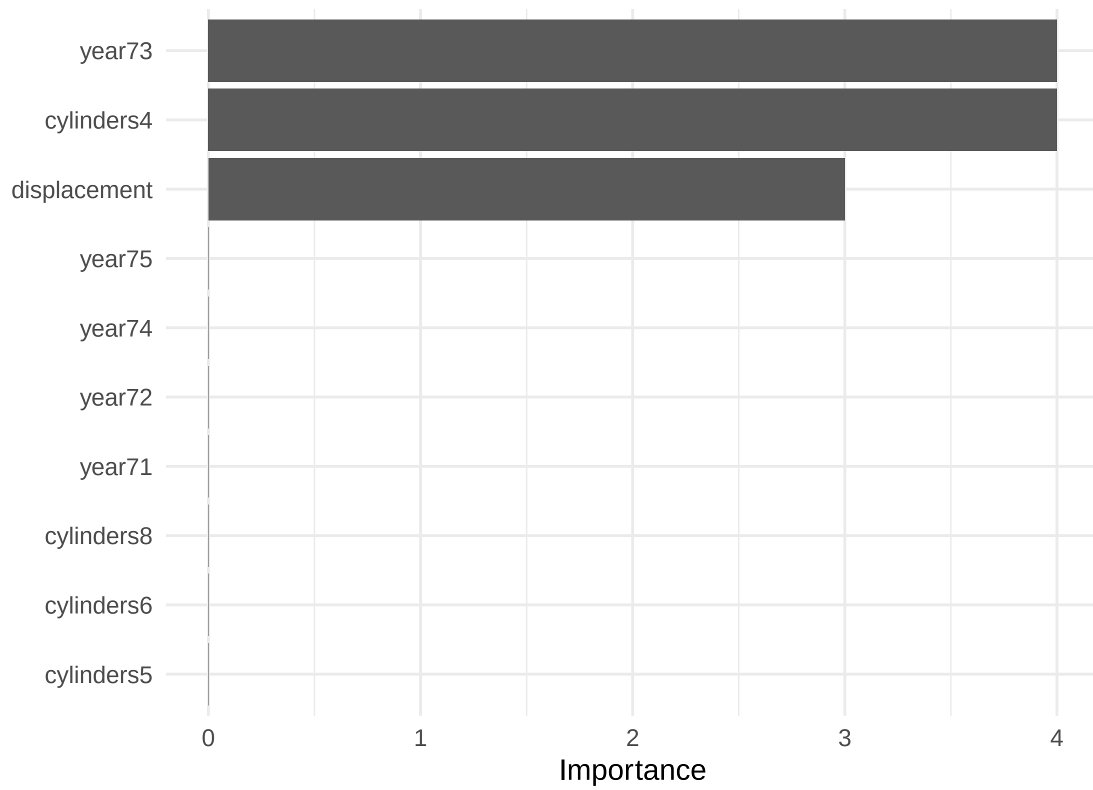

```r
mars_fit$bestTune
```

```
##   nprune degree
## 5      6      1
```

```r
coef(mars_fit$finalModel)
```

```
##          (Intercept)           cylinders4               year73 h(displacement-163)
##           -0.2371402            3.1419082           -3.8513130           0.1322963
## h(displacement-200) h(displacement-183)
##            0.3938448           -0.5229625
```

```r
# importance plot
vip(mars_fit$finalModel)
```




- From 'earth', the best tune metrics are nprune = 6, degree = 1, which is consistent with the ROC curve in the plot.
- There are 6 terms in the final model: intercept, `cylinders4`, `year73`, `h(displacement-163)`, `h(displacement-200)`, `h(displacement-183)`.
- From the importance plot above, there are 4 important predictors: `year73`, `cylinders4`, `displacement`.

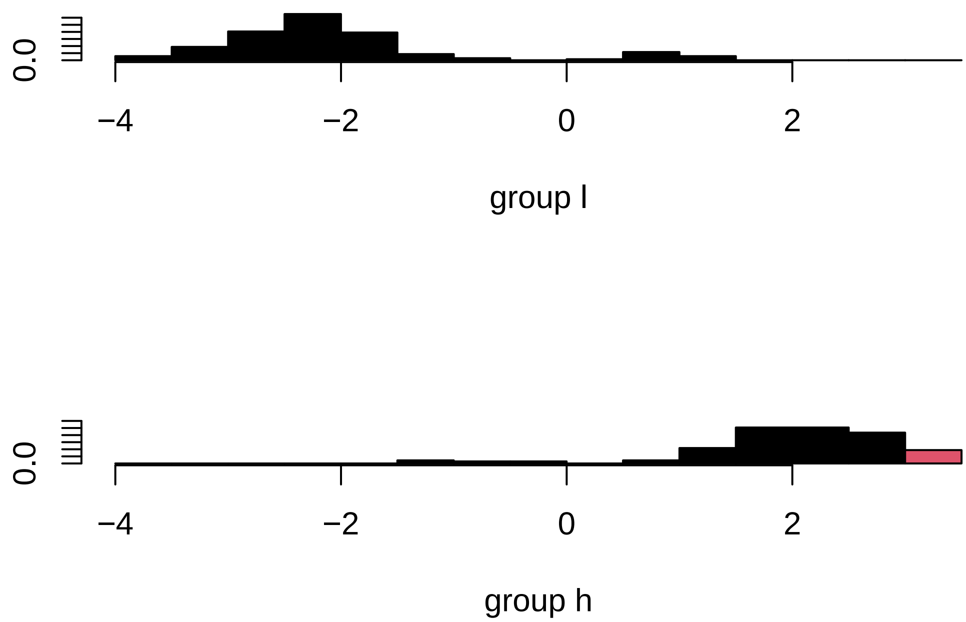
## LDA

```
lda_fit = lda(
  mpg_cat~.,
  data = auto,
  subset = index_train)

lda_pred = predict(lda_fit, newdata = auto[-index_train, ])

# probabilities of reponse level
head(lda_pred$posterior)
```

```
##         low         high
## 1 0.9995071 4.929342e-04
## 2 0.9997244 2.755540e-04
## 3 0.9993182 6.818449e-04
```

```
## 4 0.9992016 7.983627e-04
## 5 0.9999345 6.547503e-05
## 6 0.9999655 3.447012e-05
```

```
# plot linear discriminants
plot(lda_fit, col = as.numeric(auto$mpg_cat), abbrev = TRUE)
```




```
# scaling matrix
lda_fit$scaling
```

```
##                         LD1
## cylinders4      3.6969763304
## cylinders5      2.7920775239
## cylinders6      0.6697421939
## cylinders8      1.2283252963
## year71         -0.0340684304
## year72         -0.4236537995
## year73         -0.8140696186
## year74          0.3655853633
## year75          0.3008403033
## year76         -0.1917031563
## year77          0.4415032169
## year78         -0.0868919263
## year79          0.7396673704
## year80          0.9700822730
## year81          1.3395345249
```

```
## year82          1.1687945187
## originEuropean  0.4169519838
## originJapanese  0.2417803278
## displacement   -0.0018384177
## horsepower      0.0006593578
## weight         -0.0005835299
## acceleration   -0.0470062239
```

- LDA has no tuning parameters, it classifies the data by nearest centroid. Since there are 2 levels of the response variable, we have k = 2 - 1 = 1 linear discriminants.
- The linear discriminant plot shows the histogram of transformed X (predictors) for both levels. From the plot, when X is lower, data are tend to be classified in the high `mpg_cat` group, and vice versa.

## Model comparison

### ROC and AUC

```
glm_pred = predict(
  glm_fit,
  newdata = auto[-index_train, ],
  type = "response")

mars_pred = predict(
  mars_fit,
  newdata = auto[-index_train, ],
  type = "prob")[,2]

lda_pred = predict(
  lda_fit,
  newdata = auto[-index_train,])$posterior[,2]

# ROC
glm_roc = roc(auto$mpg_cat[-index_train], glm_pred)
mars_roc = roc(auto$mpg_cat[-index_train], mars_pred)
lda_roc = roc(auto$mpg_cat[-index_train], lda_pred)

# AUC
auc = c(glm_roc$auc[1], mars_roc$auc[1], lda_roc$auc[1])

model_names = c("glm","mars","lda")

# plot ROC curve
dev.off() # to fix "invalid graphics state" error in ggroc
```

```
## null device
##           1
```

```
ggroc(
  list(glm_roc, mars_roc, lda_roc),
  legacy.axes = TRUE) +
```

```
  scale_color_discrete(
    labels = paste0(
      model_names,
      " (", round(auc, 3),")"),
    name = "Models (AUC)") +
  geom_abline(intercept = 0, slope = 1, color = "grey")
```

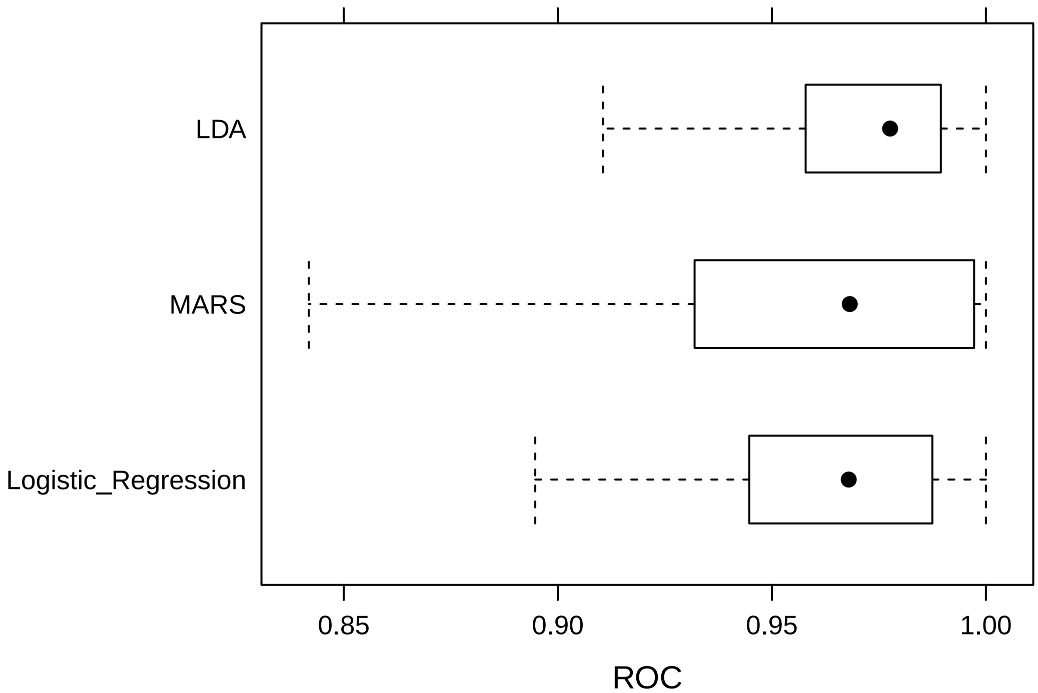
The LDA model or logistic regression model is preferred, since they have higher AUC than MARS model.

```
set.seed(1115)

# refit glm and lda models using caret to incorporate cross-validation
glm_caret =
  train(mpg_cat~.,
        data = auto,
        method = "glm",
        metric = "ROC",
        trControl = ctrl)

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

res = resamples(
  list(Logistic_Regression = glm_caret,
       MARS = mars_fit,
       LDA = lda_caret),
  times = 100)

summary(res)
```

```
##
## Call:
## summary.resamples(object = res)
##
## Models: Logistic_Regression, MARS, LDA
## Number of resamples: 50
##
## ROC
##                          Min.   1st Qu.    Median      Mean   3rd Qu. Max. NA's
## Logistic_Regression 0.8947368 0.9447368 0.9679605 0.9642922 0.9868750    1    0
## MARS                0.8418367 0.9323829 0.9682104 0.9582113 0.9966641    1    0
## LDA                 0.9105263 0.9578947 0.9776316 0.9721114 0.9888158    1    0
##
## Sens
##                          Min.   1st Qu.    Median      Mean   3rd Qu. Max. NA's
## Logistic_Regression 0.7894737 0.8947368 0.9000000 0.9131053 0.9500000    1    0
## MARS                0.7142857 0.8571429 0.9285714 0.8971429 0.9285714    1    0
## LDA                 0.6842105 0.8421053 0.8947368 0.8922105 0.9500000    1    0
##
```

```
## Spec
##                              Min.   1st Qu.    Median      Mean 3rd Qu. Max. NA's
## Logistic_Regression 0.7500000 0.8947368 0.9000000 0.9202632    0.95    1    0
## MARS                0.7142857 0.8571429 0.9285714 0.9304396    1.00    1    0
## LDA                 0.8000000 0.8947368 0.9000000 0.9195789    0.95    1    0
```

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




From the plot above, with cross-validation, the LDA model is preferred since it has the highest ROC.

### Misclassfication error rate

```
# function of calculating misclassfication error rate
error_rate = function(model_name, pred_prob, cutoff){
  pred.label = rep("low", length(pred_prob))
  pred.label[pred_prob > cutoff] = "high"
  confusionMatrix =
       table(
       tibble(pred = pred.label,
              reference = auto$mpg_cat[-index_train]))
  error = (confusionMatrix['high','low'] + confusionMatrix['low','high'])/length(pred_prob)
  print(error)
}
```

```r
# when cut-off is 0.5
error_rate('Logistic Regression', glm_pred, 0.5)
```

```
## [1] 0.1206897
```

```r
error_rate('MARS', mars_pred, 0.5)
```

```
## [1] 0.1293103
```

```r
error_rate('LDA', lda_pred, 0.5)
```

```
## [1] 0.137931
```

- Using a simple classifier with a cut-off of 0.5, Logistic Regression model is the best, since it has the lowest misclassification rate.
- One can also use the function to explore other situation with different cut-off or models.
- The higher the cut-off is, the more data will be classified to "lower" class.