

P8106 HW3

Brian Jo Hsuan Lee

3/22/2022

Load packages

```
library(tidyverse)
library(knitr)
library(AppliedPredictiveModeling)
library(pROC)
library(caret)
library(klaR)
library(MASS)
```

Import and tidy data

```
data = read_csv("auto.csv") %>%
  mutate(
    year = factor(year),
    origin = factor(origin),
    mpg_cat = factor(mpg_cat),
    mpg_cat = fct_relevel(mpg_cat, c("low", "high"))
  )
```

Partition the data for model training

```
set.seed(2022)

# partition data into training and testing sets as randomized 7:3 splits
train_index = createDataPartition(y = data$mpg_cat, p = 0.7, list = FALSE)
train_data = data[train_index, ]
train_cont_data = data[train_index, -6:-7]
test_data = data[-train_index, ]

# matrices of predictors
train_pred = model.matrix(mpg_cat ~ ., train_data)[ , -1]
train_cont_pred = model.matrix(mpg_cat ~ ., train_data)[ , 2:6]
# test_pred = model.matrix(mpg_cat ~ ., test_data)[ , -1]
# test_cont_pred = model.matrix(mpg_cat ~ ., train_data)[ , 2:6]

# vectors of response
train_resp = train_data$mpg_cat
test_resp = test_data$mpg_cat
```

a)

Produce data summaries

Calculate descriptive statistics for the training data: quantile data for the continuous variables and count data for the categorical variables. Number of cylinders is arguably an ordinal categorical variable but is treated as a continuous variable here. Most cars have an American origin (category 1), and the number of high and low mileage car samples are the same.

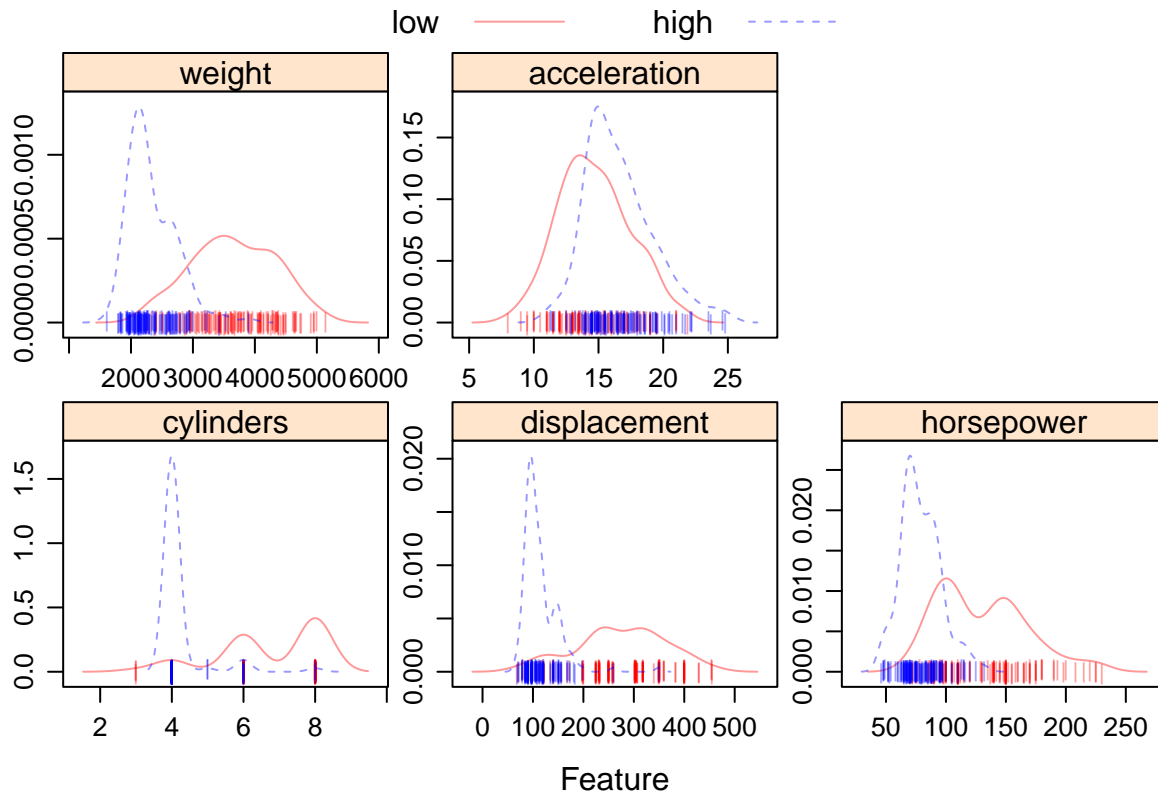
```
summary(train_data)
```

```
##      cylinders      displacement      horsepower      weight      acceleration
##  Min.   :3.00    Min.   : 68.0    Min.   : 46.0    Min.   :1613    Min.   : 8.00
##  1st Qu.:4.00    1st Qu.:100.2    1st Qu.: 75.0    1st Qu.:2222    1st Qu.:13.90
##  Median :4.00    Median :151.0    Median : 95.0    Median :2798    Median :15.50
##  Mean   :5.46    Mean   :194.3    Mean   :104.5    Mean   :2991    Mean   :15.66
##  3rd Qu.:8.00    3rd Qu.:302.0    3rd Qu.:129.2    3rd Qu.:3635    3rd Qu.:17.32
##  Max.   :8.00    Max.   :455.0    Max.   :230.0    Max.   :5140    Max.   :24.80
##
##      year      origin mpg_cat
##  78      : 28    1:175    low :138
##  73      : 27    2: 47    high:138
##  72      : 22    3: 54
##  75      : 22
##  76      : 22
##  79      : 22
##  (Other):133
```

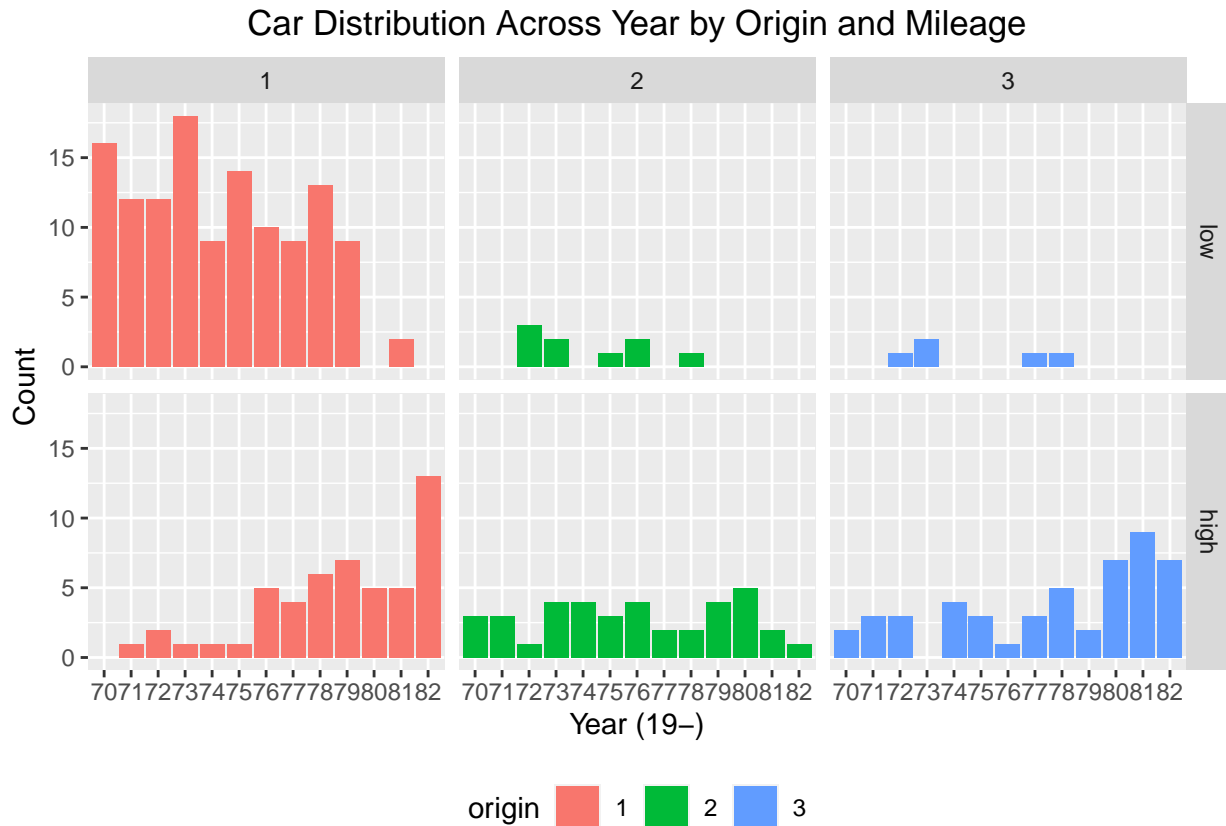
Visualize training data distribution. In general, cars with high mileage have lower weights, cylinder count, engine displacement in inches, and horsepower. Note the unequal distribution of car count when conditioned on their origin and mileage.

```
# set graphic theme
trellis.par.set(transparentTheme(trans = .4))

# plot distribution of the continuous predictors
featurePlot(train_cont_pred, train_resp,
            scales = list(x = list(relation = "free"),
                           y = list(relation = "free")),
            plot = "density", pch = "|",
            auto.key = list(columns = 2))
```



```
# plot distribution of the categorical predictors
train_data %>%
  count(year, origin, mpg_cat) %>%
  ggplot(aes(x = year, y = n, fill = origin)) +
  geom_col() +
  facet_grid(cols = vars(origin), rows = vars(mpg_cat)) +
  labs(
    title = "Car Distribution Across Year by Origin and Mileage",
    x = "Year (19-)",
    y = "Count"
  ) +
  theme(
    plot.title = element_text(hjust = 0.5),
    legend.position = "bottom"
  )
)
```



b)

Logit Regression

Fit a logit model and list its significant coefficients, which are car weight, model year 79, model year 81, and European origin.

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

```
##
## Call:
## glm(formula = mpg_cat ~ ., family = binomial(link = "logit"),
##      data = train_data)
##
## Deviance Residuals:
##      Min       1Q   Median       3Q      Max
## -2.11900  -0.07509  -0.00004   0.07578   2.85467
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  1.657e+01  5.458e+00   3.036  0.0024 **
## cylinders    2.827e-01  6.146e-01   0.460  0.6455
## displacement 1.306e-02  1.746e-02   0.748  0.4544
## horsepower  -2.206e-02  3.178e-02  -0.694  0.4877
```

```
## weight      -7.571e-03  1.919e-03  -3.946  7.96e-05 ***
## acceleration 5.695e-02  1.793e-01   0.318   0.7507
## year71      -1.327e+00  1.670e+00  -0.795   0.4267
## year72      -1.125e+00  1.490e+00  -0.755   0.4502
## year73      -1.740e+00  1.513e+00  -1.150   0.2502
## year74       1.077e+00  1.798e+00   0.599   0.5490
## year75       1.324e+00  1.629e+00   0.813   0.4164
## year76       2.209e+00  1.814e+00   1.218   0.2233
## year77       2.333e+00  2.057e+00   1.134   0.2568
## year78       1.763e+00  1.579e+00   1.116   0.2642
## year79       4.374e+00  1.738e+00   2.517   0.0118 *
## year80       2.102e+01  2.099e+03   0.010   0.9920
## year81       4.028e+00  1.774e+00   2.270   0.0232 *
## year82       2.089e+01  2.004e+03   0.010   0.9917
## origin2      2.509e+00  1.101e+00   2.279   0.0226 *
## origin3      1.213e+00  1.000e+00   1.213   0.2253
## ---
## 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: 81.557 on 256 degrees of freedom
## AIC: 121.56
##
## Number of Fisher Scoring iterations: 18
```

Build a confusion matrix, and extract the overall fraction of correct predictions. This confusion matrix demonstrates the number of correct predictions generated by our logit model. The rows correspond to what the model predicted, and the columns correspond to the known truths; the number of true lows, true highs, false lows, and false highs are 49, 53, 5, and 9, respectively. The overall fraction of correct predictions could be calculated by $\frac{49+53}{49+9+5+53} = 0.880$.

```
# use the model to forecast new observations provided by the testing data
logit_pred_prob = predict(logit_fit, newdata = test_data,
                           type = "response")

# create a vector that is equal in length to the testing data and holds the predicted binary result
logit_pred = rep("low", length(logit_pred_prob))
logit_pred[logit_pred_prob > 0.5] = "high"

# create a confusion matrix
logit_cm = confusionMatrix(data = factor(logit_pred, levels = c("low", "high")),
                           reference = test_resp,
                           positive = "high")

# display the matrix
kable(logit_cm$table, "simple")
```

	low	high
low	49	5
high	9	53

```
# extract overall correctness of the model
logit_cm$byClass["Balanced Accuracy"]
```

```
## Balanced Accuracy
##          0.8793103
```

c)

Multivariate Adaptive Regression Spline

Set the resampling method for model fitting functions in the caret package.

```
ctrl = trainControl(method = "repeatedcv", repeats = 5, number = 10,
                    summaryFunction = twoClassSummary,
                    classProbs = TRUE)
```

Fit a MARS model.

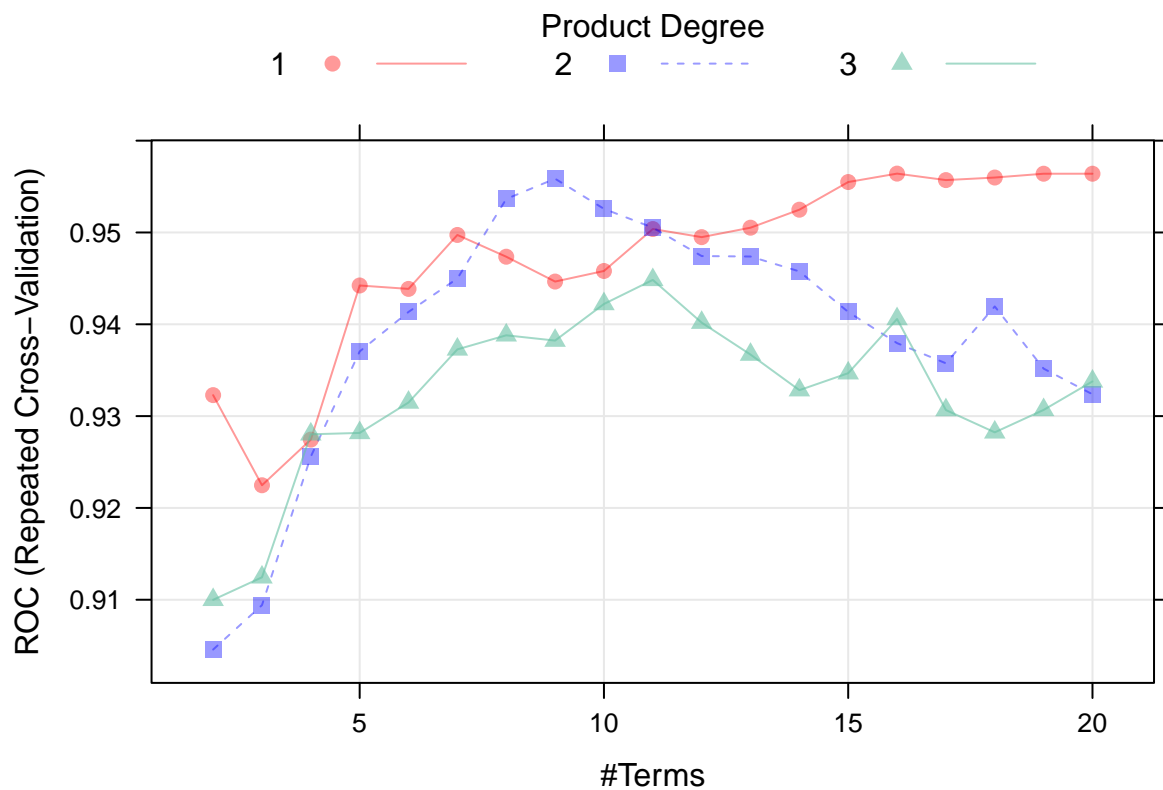
```
set.seed(2022)
```

```
# enable a grid of the 2 potential tuning parameters: degree of interactions and the number of retained
mars_grid = expand.grid(degree = 1:3, nprune = 2:20)
```

```
# fit and show
```

```
mars_fit = train(x = train_pred,
                y = train_resp,
                method = "earth",
                tuneGrid = mars_grid,
                metric = "ROC",
                trControl = ctrl)

plot(mars_fit)
```



```
# extract the optimal tuning parameters
kable(mars_fit$bestTune, "simple")
```

nprune	degree
15	1

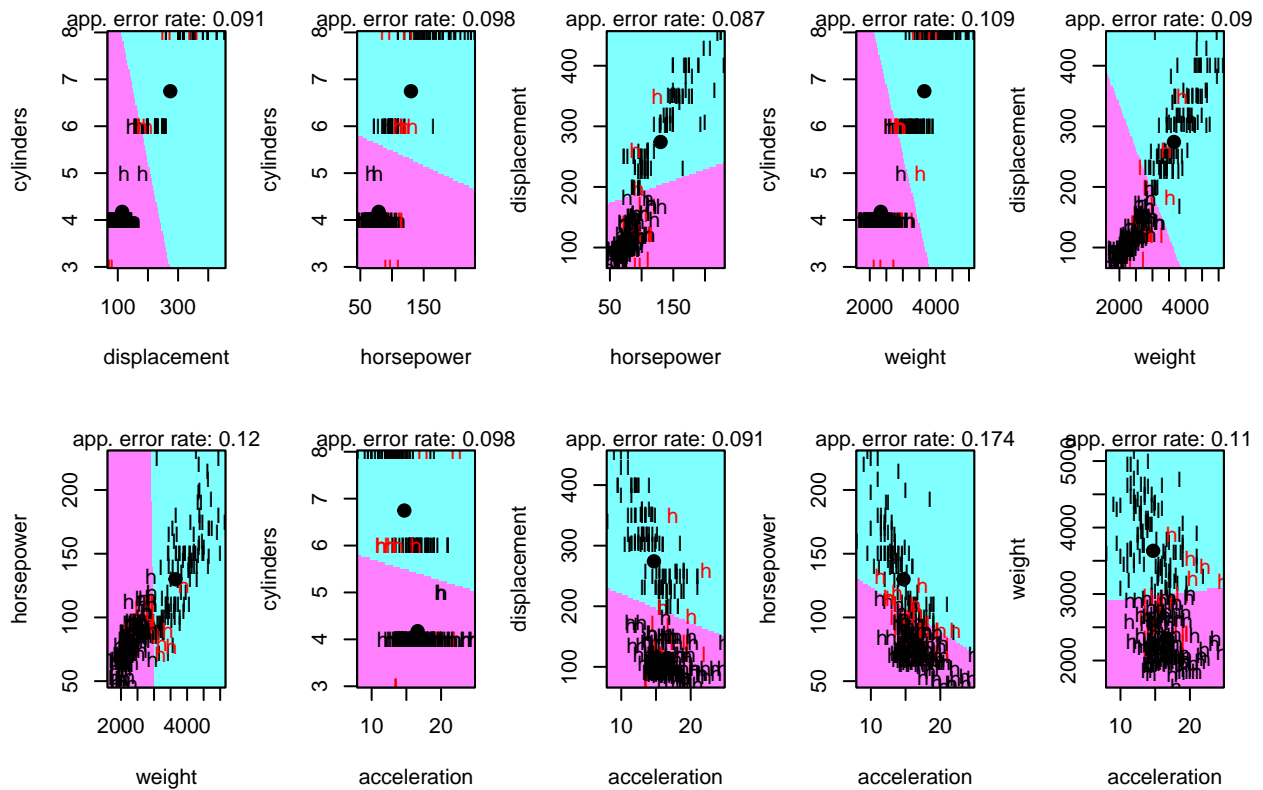
d)

LDA/QDA

Bonus: Create a partition plot for LDA exploratory data analysis. LDA and QDA do not handle categorical predictor well and they best be excluded.

```
partimat(mpg_cat ~ .,
  method = "lda",
  data = train_cont_data,
  nplots.vert = 2)
```

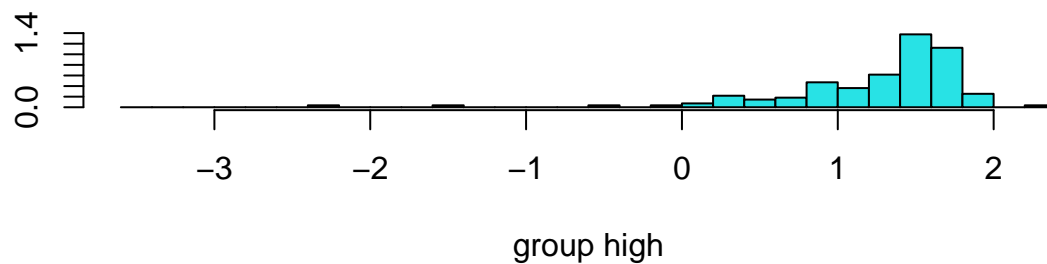
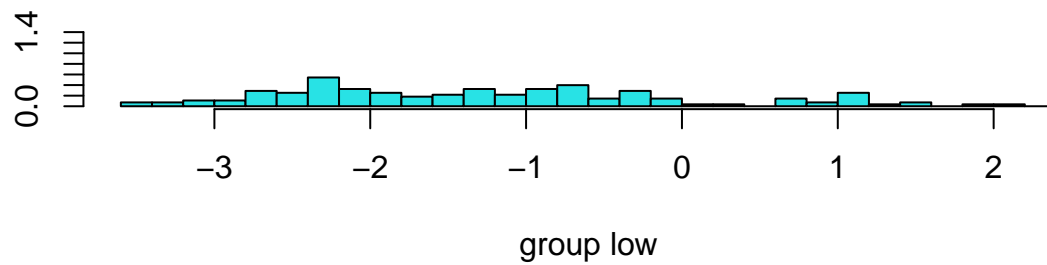
Partition Plot



Fit a LDA model.

```
lda_fit = lda(mpg_cat ~ ., data = train_cont_data)

# display linear discriminants
plot(lda_fit)
```

Fit a QDA model.

```
set.seed(2022)

qda_fit = train(x = train_cont_pred,
                 y = train_resp,
                 method = "qda",
                 metric = "ROC",
                 trControl = ctrl)
```

e)

Model Selection and justification

Compare ROC. Resamples() compares caret models, so we need to recreate a logit and an LDA model using the caret package

```
set.seed(2022)

logit_fit_caret = train(x = train_pred,
                        y = train_resp,
                        method = "glm",
                        metric = "ROC",
                        trControl = ctrl)
```

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

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

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

```
lda_fit_caret = train(x = train_cont_pred,
                      y = train_resp,
```

```

        method = "lda",
        metric = "ROC",
        trControl = ctrl)

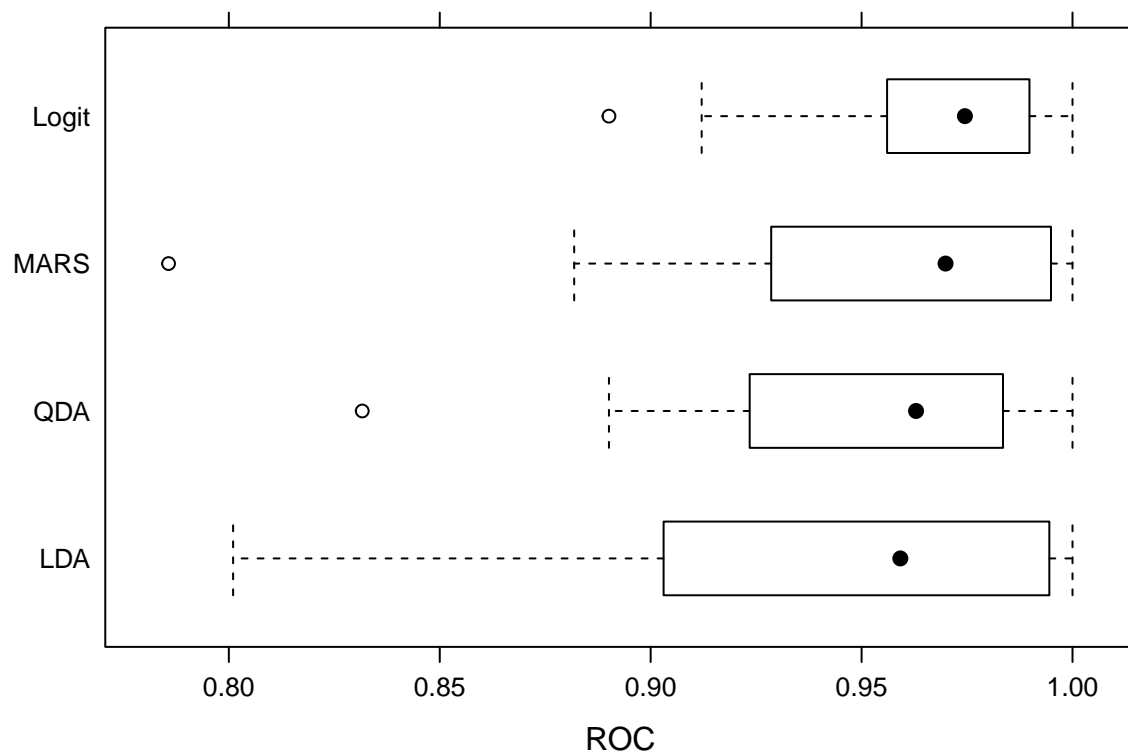
train_res = resamples(list(Logit = logit_fit_caret,
                          MARS = mars_fit,
                          LDA = lda_fit_caret,
                          QDA = qda_fit))

summary(train_res)

##
## Call:
## summary.resamples(object = train_res)
##
## Models: Logit, MARS, LDA, QDA
## Number of resamples: 50
##
## ROC
##           Min.    1st Qu.    Median      Mean    3rd Qu. Max. NA's
## Logit 0.8901099 0.9566779 0.9744898 0.9710657 0.9897959    1    0
## MARS  0.7857143 0.9285714 0.9699010 0.9564108 0.9936224    1    0
## LDA   0.8010204 0.9053179 0.9591837 0.9402433 0.9933281    1    0
## QDA   0.8316327 0.9234694 0.9629121 0.9516562 0.9835165    1    0
##
## Sens
##           Min.    1st Qu.    Median      Mean    3rd Qu. Max. NA's
## Logit 0.7142857 0.8571429 0.9285714 0.9092308 0.9285714    1    0
## MARS  0.7142857 0.8571429 0.9285714 0.8990110 0.9285714    1    0
## LDA   0.5714286 0.7733516 0.8571429 0.8475824 0.9285714    1    0
## QDA   0.7142857 0.8461538 0.8571429 0.8726374 0.9285714    1    0
##
## Spec
##           Min.    1st Qu.    Median      Mean    3rd Qu. Max. NA's
## Logit 0.7142857 0.8571429 0.9285714 0.9187912 1.0000000    1    0
## MARS  0.7857143 0.8571429 0.9285714 0.9245055 1.0000000    1    0
## LDA   0.8461538 0.9285714 1.0000000 0.9650549 1.0000000    1    0
## QDA   0.7692308 0.8571429 0.9285714 0.9141758 0.9285714    1    0

bwplot(train_res, metric = "ROC")

```



```
logit_roc = roc(test_resp, logit_pred_prob)
plot(logit_roc, legacy.axes = TRUE, print.auc = TRUE)
plot(smooth(logit_roc), col = 4, add = TRUE)
```

