HW3

Yangyang Chen (UNI: yc4384)

3/25/2022

Contents

Set-Up and Data Preprocessing

Data Pre-processing

```
set.seed(66)

# Load data, clean column names, eliminate rows containing NA entries
auto_df = read_csv("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)
```

```r
indexTrain = createDataPartition(y = auto_df$mpg_cat,
                                 p = 0.7,
                                 list = FALSE)
```

Exploratory Data Analysis

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

```
##  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(auto_df)
```

Table 1: Data summary

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

Variable type: factor

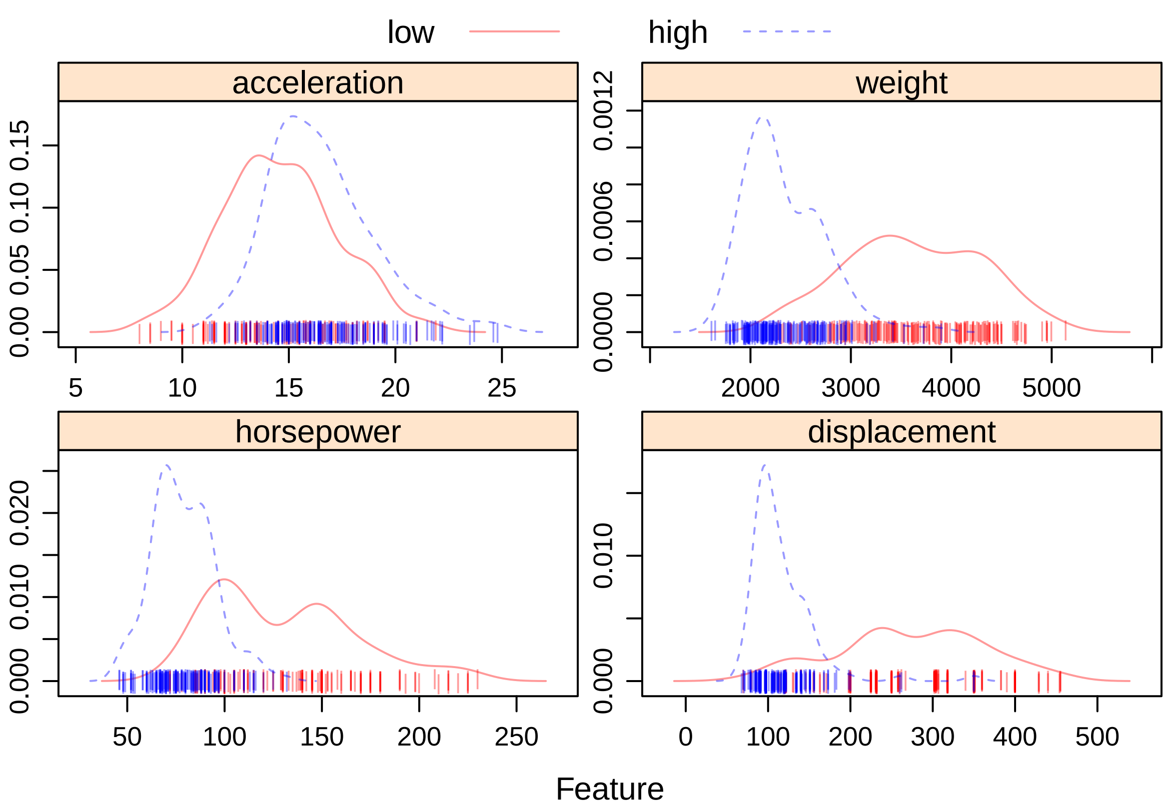
skim_variable	n_missing	complete_rate	ordered	n_unique	top_counts
cylinders	0	1	FALSE	5	4: 199, 8: 103, 6: 83, 3: 4
year	0	1	FALSE	13	73: 40, 78: 36, 76: 34, 75: 30
origin	0	1	FALSE	3	Ame: 245, Jap: 79, Eur: 68
mpg_cat	0	1	FALSE	2	low: 196, hig: 196

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.

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

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



We conduct a few basic exploratory analyses. Our feature plot of continuous covariates shows that cars with high MPG tend to have lower displacement, lower horsepower, lower weight, and higher acceleration.

Part (a): Logistic Regression

```r
set.seed(2716)

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

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

```
##
## Call:
## glm(formula = mpg_cat ~ ., family = binomial(link = "logit"),
##     data = auto_df, subset = indexTrain)
##
## Coefficients:
##                  Estimate Std. Error z value Pr(>|z|)
## (Intercept)    18.970691   7.732910   2.453  0.01416 *
## cylinders4      4.419363   3.944139   1.120  0.26251
## cylinders5      3.875363   4.536107   0.854  0.39292
## cylinders6      1.119378   4.394016   0.255  0.79892
## cylinders8      5.202670   5.303502   0.981  0.32660
## displacement   -0.007835   0.022476  -0.349  0.72741
## horsepower     -0.075947   0.045999  -1.651  0.09873 .
## weight         -0.004343   0.002489  -1.745  0.08103 .
## acceleration   -0.129298   0.286235  -0.452  0.65147
## year71         -0.677850   3.110364  -0.218  0.82748
## year72         -3.830602   1.591524  -2.407  0.01609 *
## year73         -4.882870   1.838138  -2.656  0.00790 **
## year74          1.023248   3.240995   0.316  0.75221
## year75          0.802235   1.804157   0.445  0.65657
## year76          1.959470   1.962134   0.999  0.31797
## year77         -0.901258   1.781109  -0.506  0.61285
## year78         -0.498177   1.664409  -0.299  0.76470
## year79          4.416230   1.733827   2.547  0.01086 *
## year80          3.694878   2.614387   1.413  0.15757
## year81          3.863140   1.913497   2.019  0.04350 *
## year82          4.993409   1.876139   2.662  0.00778 **
## originEuropean -0.569306   1.210575  -0.470  0.63816
## originJapanese  0.718977   1.154084   0.623  0.53329
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 382.62  on 275  degrees of freedom
## Residual deviance:  67.58  on 253  degrees of freedom
## AIC: 113.58
##
## Number of Fisher Scoring iterations: 8
```

Here, we build a logistic regression model (without penalization) from our training data. At the 0.05 significance level, `year72`, `year79`, and `year81` are significant predictors of our outcome `mpg_cat`. At the 0.01 significance level, i.e. even more significantly, our indicator variable `year73`, `year82` is a statistically significant predictor of our outcome as well. Other variables are considered as redundent variables.

Part (b): Model Performance

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

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction low high
##       low   54    7
##       high   4   51
##
##                Accuracy : 0.9052
##                  95% CI : (0.8367, 0.9517)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.8103
##
##  Mcnemar's Test P-Value : 0.5465
##
##             Sensitivity : 0.8793
##             Specificity : 0.9310
##          Pos Pred Value : 0.9273
##          Neg Pred Value : 0.8852
##              Prevalence : 0.5000
##          Detection Rate : 0.4397
##    Detection Prevalence : 0.4741
##       Balanced Accuracy : 0.9052
##
##        'Positive' Class : high
##
```

Our confusion matrix shows that our accuracy, or overall fraction of correct predictions, is roughly 90% (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 87.9% sensitive (true detected positives out of all actual positives) and 93.1% specific (true detected negatives out of all actual negatives), with a positive predictive value of 92.7% (true detected positives out of all predicted positives) and a negative predictive value of 88.5% (true detected negatives out of all predicted negatives). Our sensitivity and specificity average to 90.5%, which is our balanced accuracy. Our kappa, at 0.8103, means that our inter-rater agreement is quite high, even accounting for the possibility of agreement by chance.

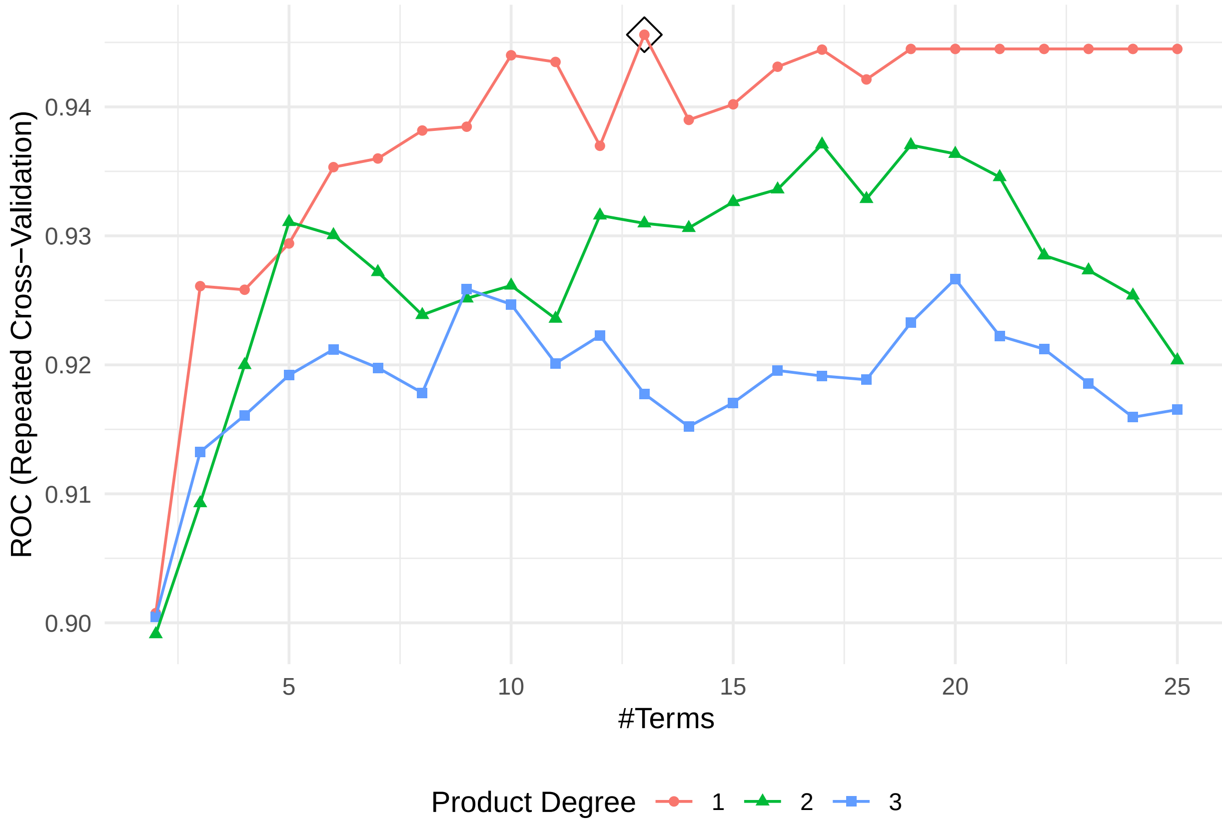
Part (c): MARS Model

```r
# Train MARS model using the training data
set.seed(2716)

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

model.mars = train(x = auto_df[indexTrain, 1:7],
                   y = auto_df$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=13)
##
## GLM coefficients
##                           high
## (Intercept)          1.4216923
## cylinders4           3.8810717
## cylinders5           0.9155663
## year72              -3.1999531
## year73              -4.1764975
## year80               3.4556153
## year82               4.6606573
## h(displacement-120) -1.8171899
## h(displacement-122)  1.9709686
## h(displacement-168) -0.2922313
## h(displacement-225)  0.1965370
## h(horsepower-81)    -0.3548841
## h(horsepower-85)     0.2559427
##
## GLM (family binomial, link logit):
##  nulldev  df       dev  df   devratio     AIC iters converged
##  382.617 275   74.3522 263      0.806   100.4     8         1
##
## Earth selected 13 of 27 terms, and 8 of 22 predictors (nprune=13)
## Termination condition: Reached nk 45
```

```
## Importance: cylinders4, displacement, year73, year72, horsepower, year82, ...
## Number of terms at each degree of interaction: 1 12 (additive model)
## Earth GCV 0.06575845    RSS 15.01032    GRSq 0.7388688    RSq 0.7824591
```

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




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

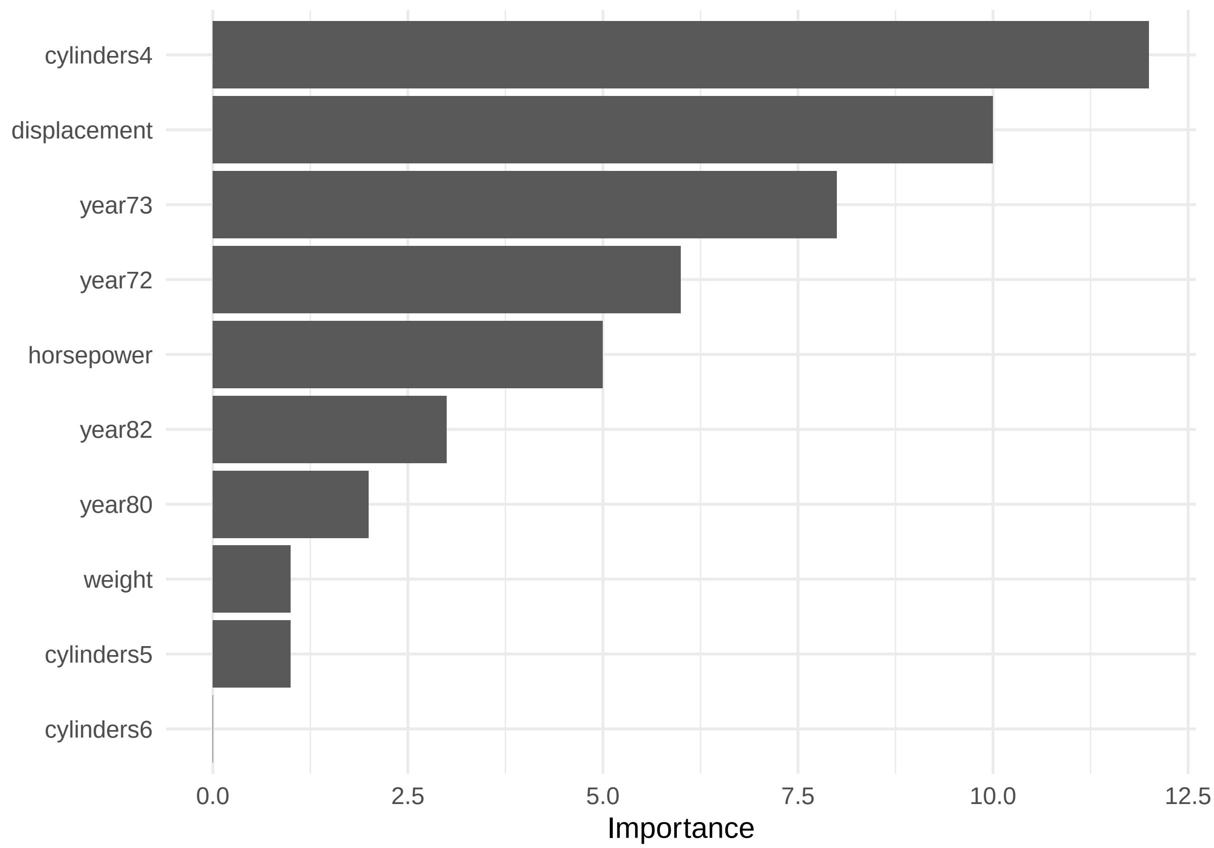
	nprune	degree
12	13	1

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

	Coefficient
(Intercept)	1.4216923
cylinders4	3.8810717
year73	-4.1764975
year72	-3.1999531
cylinders5	0.9155663
year82	4.6606573
year80	3.4556153
h(displacement-168)	-0.2922313

	Coefficient
h(displacement-122)	1.9709686
h(displacement-225)	0.1965370
h(displacement-120)	-1.8171899
h(horsepower-81)	-0.3548841
h(horsepower-85)	0.2559427

```
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 `displacement` and indicators `year73`, `year72`, and `horsepower` 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 13 out of 27 terms, representing 8 of 22 predictors (nprune terms = 13, product degree = 1). The model is optimized with and has an R-squared of 0.7824.

Importantly, MARS improves the prediction performance compared to logistic regression due to comparatively smaller AIC values (100.4) and deviance (74.3).

Part (d): LDA

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

```r
# Increase the bottom margin
par(mar = c(5, 4, 4, 2) + 0.1)

# Create a new plotting device with custom size
dev.new(width = 10, height = 8)

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

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

```
##                         LD1
## cylinders4      2.3803331367
## cylinders5      1.7330384654
## cylinders6     -0.2909014488
## cylinders8      0.2915740662
## displacement   -0.0018916459
## horsepower      0.0014499639
## weight         -0.0006709425
## acceleration   -0.0032086342
## year71          0.2091527292
## year72         -0.7138844151
## year73         -0.5565850785
## year74          0.5001581144
## year75          0.2970629253
## year76          0.2650423645
## year77          0.0491385661
## year78         -0.0957117542
## year79          1.0005785809
## year80          1.1006854632
## year81          1.0685780735
## year82          1.1768025500
## originEuropean -0.0609990246
## originJapanese  0.0534099586
```

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 $(k - 1) \times p$ 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.

```r
# Alternatively, use caret for LDA
set.seed(2716)

training_df = auto_df[indexTrain, ]

model.lda = train(mpg_cat ~ .,
                  data = training_df,
                  method = "lda",
```

```r
                   metric = "ROC",
                   trControl = ctrl)

model.lda$results
```

```
##   parameter       ROC      Sens      Spec      ROCSD     SensSD     SpecSD
## 1      none 0.9708085 0.8987912 0.9159341 0.02512759 0.08538134 0.06620683
```

For completeness, we also run an LDA model using `caret`, which has a 0.97 ROC, with 89.87% sensitivity and 91.59% specificity.

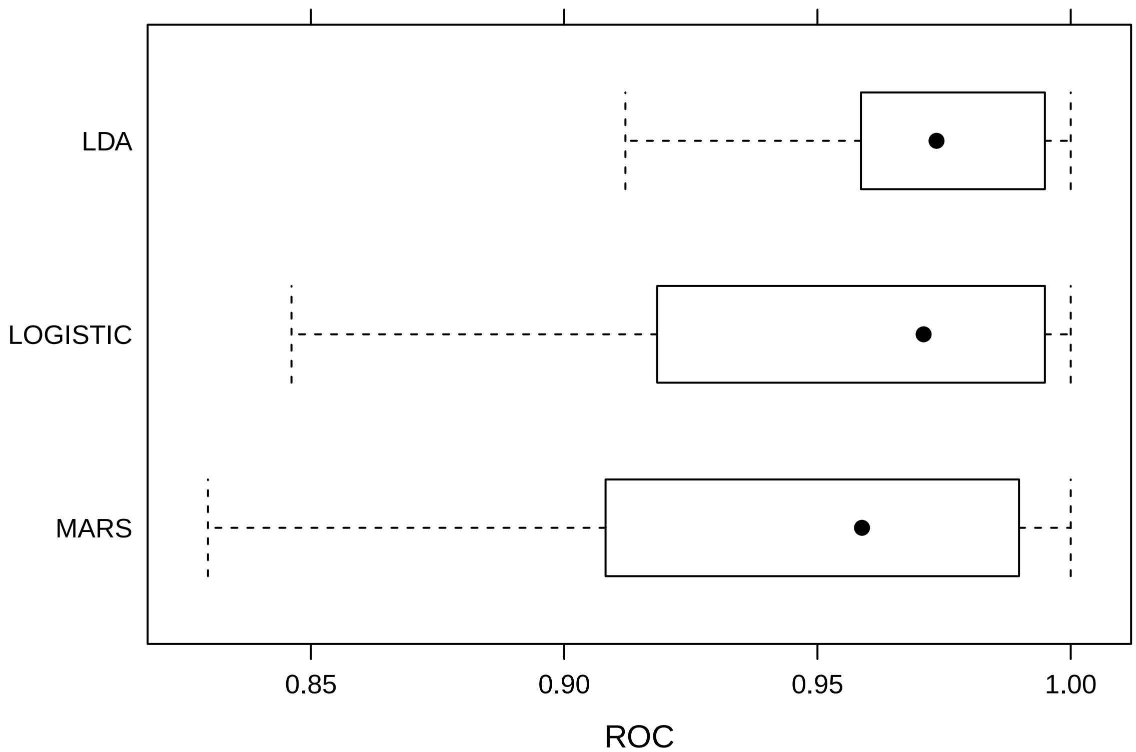
Part (e): Model Comparison and AUC/ROC

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

# Run caret logistic model
set.seed(2132)

glm.logit.caret = train(x = auto_df[indexTrain, 1:7],
                        y = auto_df$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 0.8461538 0.9196429 0.9709576 0.9493926 0.9936224    1    0
## MARS     0.8296703 0.9094388 0.9587912 0.9455950 0.9897959    1    0
## LDA      0.9120879 0.9587308 0.9735086 0.9708085 0.9947998    1    0
##
## Sens
##               Min.   1st Qu.    Median      Mean   3rd Qu. Max. NA's
## LOGISTIC 0.7857143 0.8571429 0.9258242 0.9101099 0.9285714    1    0
## MARS     0.6428571 0.8571429 0.9258242 0.9002198 0.9285714    1    0
## LDA      0.6428571 0.8571429 0.9230769 0.8987912 0.9285714    1    0
##
## Spec
```

```
##                Min.   1st Qu.    Median      Mean   3rd Qu. Max. NA's
## LOGISTIC 0.6923077 0.8571429 0.9285714 0.9085714 1.0000000    1    0
## MARS     0.6428571 0.8571429 0.9285714 0.9062637 0.9821429    1    0
## LDA      0.7692308 0.8571429 0.9285714 0.9159341 0.9821429    1    0
```

```r
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 = auto_df[-indexTrain, 1:7], type = "prob")[,2]

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

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

auc_lda
```

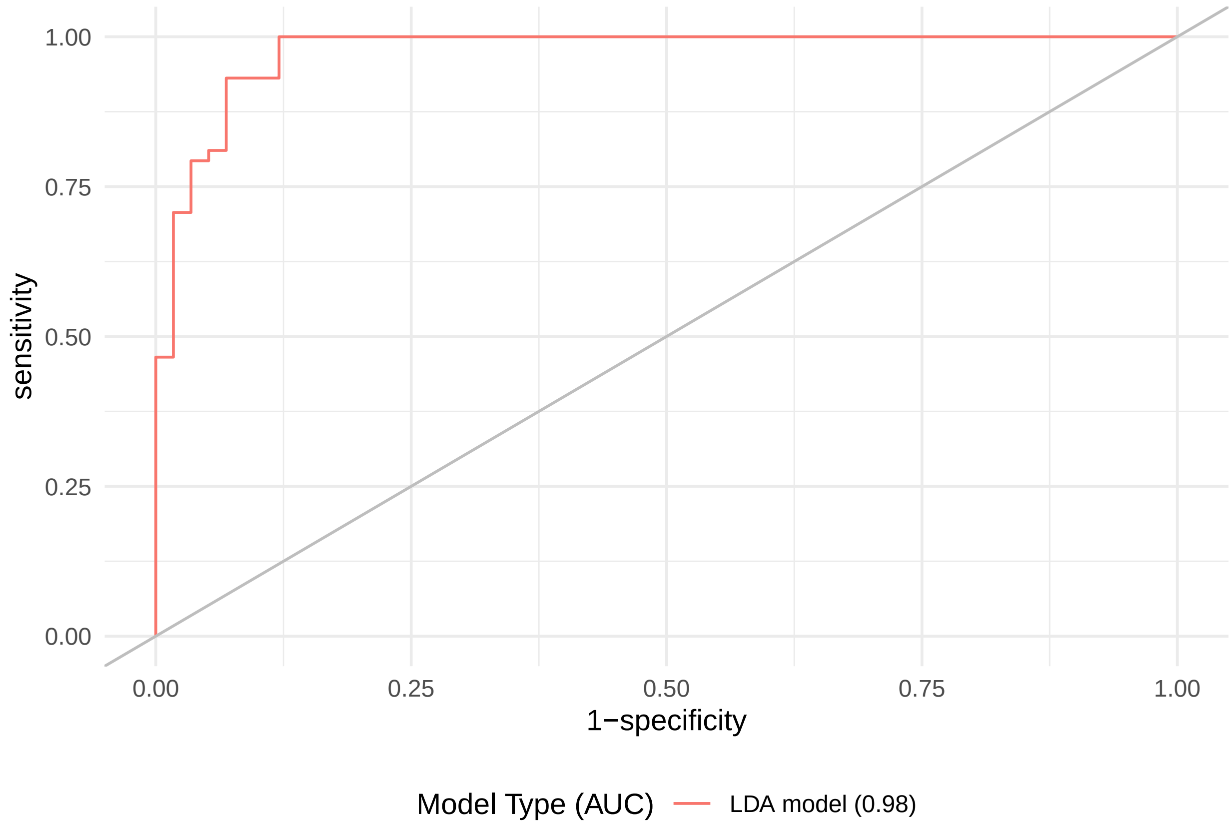
```
## [1] 0.975327
```

```r
# Obtain classes
lda_class = lda.predict |>
  as.data.frame() |>
  mutate(
    class = case_when(
```

```r
      lda.predict < 0.50 ~ "low",
      lda.predict > 0.50 ~ "high")
  ) |>
  dplyr::select(class) |>
  as.matrix()

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

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction low high
##       low   51    4
##       high   7   54
##
##                Accuracy : 0.9052
##                  95% CI : (0.8367, 0.9517)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.8103
##
##  Mcnemar's Test P-Value : 0.5465
##
##             Sensitivity : 0.9310
##             Specificity : 0.8793
##          Pos Pred Value : 0.8852
##          Neg Pred Value : 0.9273
##              Prevalence : 0.5000
##          Detection Rate : 0.4655
##    Detection Prevalence : 0.5259
##       Balanced Accuracy : 0.9052
##
##        '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 - 0.9052, or ~10%, when we use a threshold of 0.5 probability, as well as an AUC of 0.9753, as observed on our ROC plot above.