STAT 562- Homework 2

Prashanna Raj Pandit

2025-11-07

1.Data preprocessing

```
library(ISLR2)
data(Auto)
#str(Auto)
sum(is.na(Auto))
```

```
## [1] 0
```

```
Auto$mpg01<-ifelse(Auto$mpg > median(Auto$mpg),1,0)
Auto$mpg01<-as.factor(Auto$mpg01)
Auto$origin<-as.factor(Auto$origin)
Auto<-subset(Auto,select= -c(mpg,name))
prop.table(table(Auto$mpg01))
```

```
##
##   0   1
## 0.5 0.5
```

2.Train/test split

Answer: After performing the stratified 75-25 split, both the training and test sets ended up with the same distribution of the response variable 50% of the cars have low mileage (mpg01 = 0) and 50% have high mileage (mpg01 = 1).

This shows that the split worked perfectly: each subset represents the overall data well, with no imbalance between the two classes. Because of this, the models we build later won't be biased toward either high- or low-mileage cars, leading to fairer and more reliable results.

```
library(rsample)
set.seed(123)
split<-initial_split(Auto,prop = 0.75,strata = mpg01)
train_data<-training(split)
test_data<-testing(split)

prop.table(table(train_data$mpg01))
```

```
##
##   0   1
## 0.5 0.5
```

```
prop.table(table(test_data$mpg01))
```

```
## 
##   0   1 
## 0.5 0.5
```

3. Perform LDA, QDA, Naive Bayes with 10-fold cross validation.

```
library(caret)

## Loading required package: ggplot2

## Loading required package: lattice

## 
## Attaching package: 'caret'

## The following object is masked from 'package:rsample':
## 
##     calibration

library(MASS)

## 
## Attaching package: 'MASS'

## The following object is masked from 'package:ISLR2':
## 
##     Boston

library(e1071)

## 
## Attaching package: 'e1071'

## The following object is masked from 'package:rsample':
## 
##     permutations

set.seed(123)
train_control<-trainControl(method = "cv", number=10)
model_lda<-caret::train(mpg01~., data=train_data,
method='lda',trControl=train_control,metric="Accuracy")
model_qda<-caret::train(mpg01~., data=train_data,
method="qda",trControl=train_control,metric="Accuracy")
model_nb<-caret::train(mpg01~.,
data=train_data,method='naive_bayes',trControl=train_control,metric="Accuracy")
model_lda

## Linear Discriminant Analysis 
## 
## 294 samples
##   7 predictor
##   2 classes: '0', '1' 
## 
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 264, 265, 265, 265, 264, 266, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.9082594  0.8167377
```

```
model_qda
```

```
## Quadratic Discriminant Analysis
##
## 294 samples
##   7 predictor
##   2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 266, 266, 264, 264, 264, 264, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.9037028  0.8073974
```

```
model_nb
```

```
## Naive Bayes
##
## 294 samples
##   7 predictor
##   2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 264, 265, 265, 265, 265, 264, ...
## Resampling results across tuning parameters:
##
##   usekernel  Accuracy   Kappa
##   FALSE      0.9016092  0.8035999
##    TRUE      0.8981609  0.7967124
##
## Tuning parameter 'laplace' was held constant at a value of 0
## Tuning
##  parameter 'adjust' was held constant at a value of 1
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were laplace = 0, usekernel = FALSE
##  and adjust = 1.
```

4. Predictions & confusion matrices

Answer: All three models (LDA, QDA, and Naive Bayes) performed almost identically on the test set, each achieving about 91.8% accuracy. They all showed perfect **precision** (1.00), meaning every car predicted as high mileage truly was high mileage. The **recall** of roughly 0.84

shows that the models correctly identified most, but not all, of the high-mileage cars. Their **F1-scores** (around 0.91) indicate a very good balance between precision and recall.

From the **ROC** curves, all three lines are close to the top-left corner, showing that the models can clearly separate high and low-mileage cars. Their AUC values confirm this: QDA (0.958) was just slightly higher than LDA (0.955), while Naive Bayes (0.934) was only a bit behind.

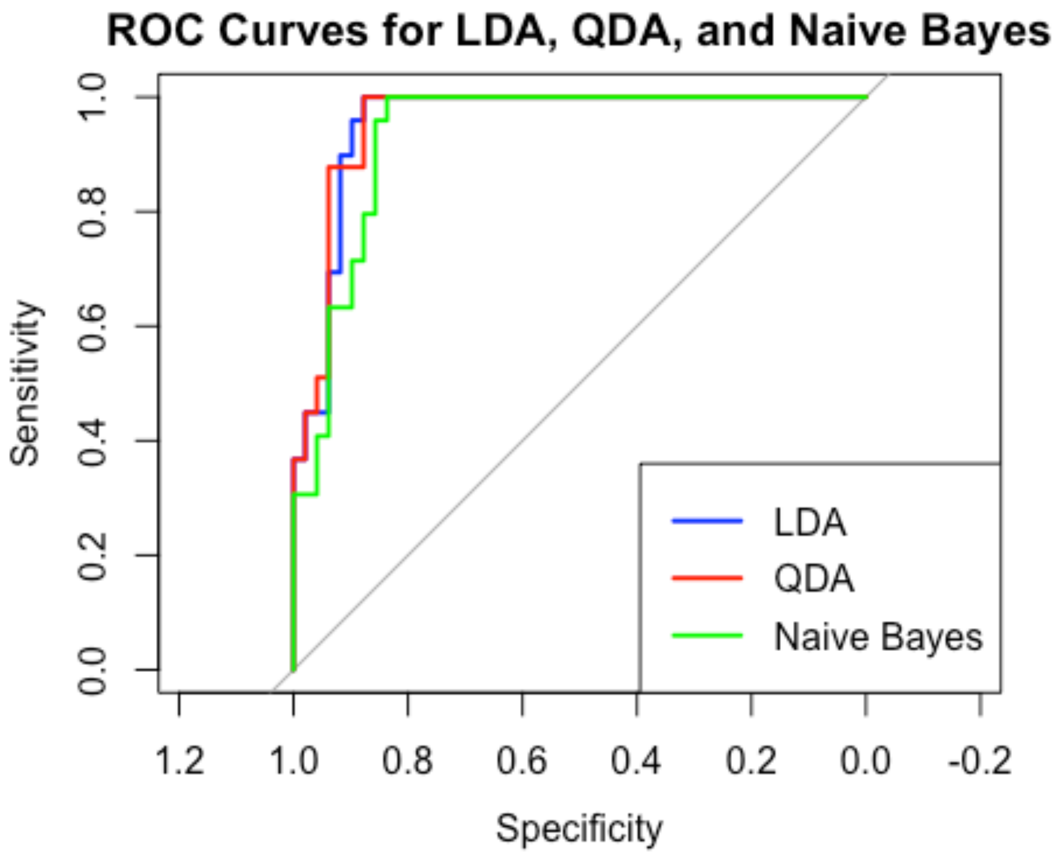
Even though QDA had the highest AUC, the difference between the three models is tiny. Because LDA is simpler and gives almost the same performance, it would likely be the best choice overall it’s easier to interpret and less likely to overfit.

```
# ---- Make predictions on the test data ----
pred_lda <- predict(model_lda, newdata = test_data)
pred_qda <- predict(model_qda, newdata = test_data)
pred_nb  <- predict(model_nb,  newdata = test_data)

# ---- Confusion matrices ----
lda_cm <- caret::confusionMatrix(pred_lda, test_data$mpg01)
qda_cm <- caret::confusionMatrix(pred_qda, test_data$mpg01)
nb_cm  <- caret::confusionMatrix(pred_nb,  test_data$mpg01)

lda_cm$table

##           Reference
## Prediction  0  1
##          0 41  0
##          1  8 49

qda_cm$table

##           Reference
## Prediction  0  1
##          0 41  0
##          1  8 49

nb_cm$table

##           Reference
## Prediction  0  1
##          0 41  0
##          1  8 49

# --- LDA ---
lda_acc  <- lda_cm$overall["Accuracy"]
lda_prec <- lda_cm$byClass["Precision"]
lda_rec  <- lda_cm$byClass["Recall"]
lda_f1   <- lda_cm$byClass["F1"]

# --- QDA ---
qda_acc  <- qda_cm$overall["Accuracy"]
qda_prec <- qda_cm$byClass["Precision"]
qda_rec  <- qda_cm$byClass["Recall"]
```

```
qda_f1   <- qda_cm$byClass["F1"]

# --- Naive Bayes ---
nb_acc  <- nb_cm$overall["Accuracy"]
nb_prec <- nb_cm$byClass["Precision"]
nb_rec  <- nb_cm$byClass["Recall"]
nb_f1   <- nb_cm$byClass["F1"]

library(pROC)

## Type 'citation("pROC")' for a citation.

##
## Attaching package: 'pROC'

## The following objects are masked from 'package:stats':
##
##     cov, smooth, var

lda_prob<-predict(model_lda,newdata = test_data,type="prob")[,2]
qda_prob<-predict(model_qda,newdata = test_data,type="prob")[,2]
nb_prob<-predict(model_nb,newdata = test_data,type="prob")[,2]

roc_lda <- roc(test_data$mpg01, lda_prob)

## Setting levels: control = 0, case = 1

## Setting direction: controls < cases

roc_qda <- roc(test_data$mpg01, qda_prob)

## Setting levels: control = 0, case = 1
## Setting direction: controls < cases

roc_nb  <- roc(test_data$mpg01, nb_prob)

## Setting levels: control = 0, case = 1
## Setting direction: controls < cases

auc_lda <- pROC::auc(roc_lda)
auc_qda <- pROC::auc(roc_qda)
auc_nb  <- pROC::auc(roc_nb)

metrics <- data.frame(
  Model     = c("LDA", "QDA", "Naive Bayes"),
  Accuracy  = c(lda_acc, qda_acc, nb_acc),
  Precision = c(lda_prec, qda_prec, nb_prec),
  Recall    = c(lda_rec, qda_rec, nb_rec),
  F1_Score  = c(lda_f1, qda_f1, nb_f1),
  AUC       = c(auc_lda, auc_qda, auc_nb)
)

metrics
```

```
##          Model  Accuracy Precision    Recall  F1_Score       AUC
## 1          LDA 0.9183673         1 0.8367347 0.9111111 0.9554352
## 2          QDA 0.9183673         1 0.8367347 0.9111111 0.9583507
## 3 Naive Bayes 0.9183673         1 0.8367347 0.9111111 0.9337776
```

```
library(ggplot2)
plot(roc_lda, col = "blue", main = "ROC Curves for LDA, QDA, and Naive Bayes")
plot(roc_qda, col = "red", add = TRUE)
plot(roc_nb,  col = "green", add = TRUE)
legend("bottomright", legend=c("LDA", "QDA", "Naive Bayes"),
       col=c("blue", "red", "green"), lwd=2)
```


ROC Curves for LDA, QDA, and Naive Bayes
Sensitivity
Specificity
LDA
QDA
Naive Bayes