Quiz 2

```
###############
#loading libraries
###############
library(readr)
library(MASS)
library(naivebayes)
```

```
## naivebayes 1.0.0 loaded
```

```
## For more information please visit:
```

```
## https://majkamichal.github.io/naivebayes/
```

```
library(broom)
library(tidyr)
library(caret)
```

```
## Loading required package: ggplot2
```

```
## Loading required package: lattice
```

```
library(data.table)
```

```
##
## Attaching package: 'data.table'
```

```
## The following object is masked from 'package:naivebayes':
##
##     tables
```

```
library(arules)
```

```
## Loading required package: Matrix
```

```
##
## Attaching package: 'Matrix'
```

```
## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack
```

```
##
## Attaching package: 'arules'

## The following objects are masked from 'package:base':
##
##     abbreviate, write
```

```
library(arulesViz)
library(FNN)
library(ggplot2)

############
#Read data
############

quiz2 <- read_csv("~/Desktop/MTH443/Quiz/Quiz 2/quiz2.csv")
```

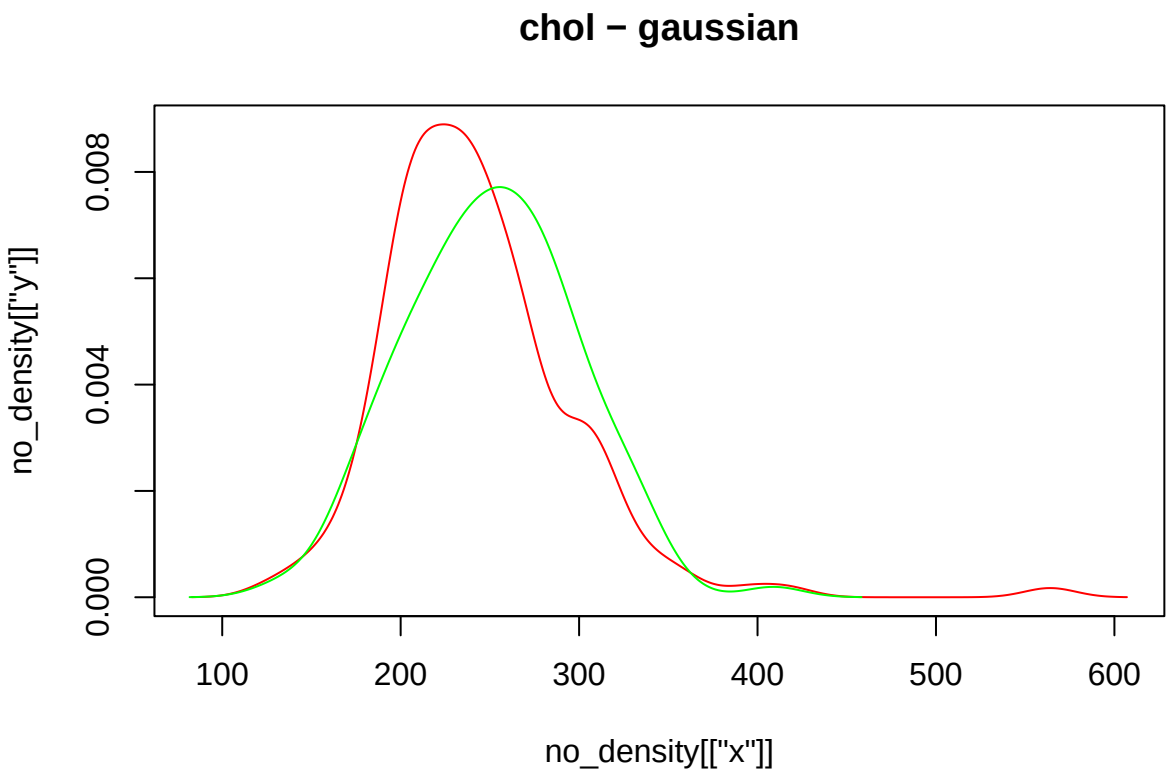
```
## Rows: 299 Columns: 11

## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl (11): age, cp, trestbps, chol, fbs, restecg, thalach, exang, oldpeak, ca...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
yesHDS <- na.omit(subset(quiz2, HDS == 1))
noHDS <- na.omit(subset(quiz2, HDS == 0))

# Variables for Density Estimation
variables <- c("chol")
kernels <- c("gaussian")
colors <- c("red", "green")

# Loop over variables and kernels
for (var in variables) {
  for (kernel in kernels) {
    no_density <- density(noHDS[[var]], na.rm = TRUE, kernel = kernel)
    yes_density <- density(yesHDS[[var]], na.rm = TRUE, kernel = kernel)
    plot(no_density[["x"]], no_density[["y"]], type = 'l', col = colors[1], main = paste(var, "-", kern
    lines(yes_density[["x"]], yes_density[["y"]], col = colors[2])
  }
}
```




```r
# Calculate P(chol > 250) for the HDS = 1 group
chol_threshold <- 250
prob_chol_above_250 <- sum(yes_density$y[yes_density$x > chol_threshold]) * diff(yes_density$x[1:2])

# Display the probability
print(paste("P(chol > 250) for HDS = 1 group:", round(prob_chol_above_250, 4)))
```

```
## [1] "P(chol > 250) for HDS = 1 group: 0.5096"
```

```r
#########
#Test/Train Split
#########

set.seed(123)
n <- nrow(quiz2)
test_indices <- seq((n - floor(0.1 * n) + 1), n)
train <- quiz2[-test_indices, ]
test <- quiz2[test_indices, ]

evaluate_quiz2_model <- function(train_data, test_data) {
  # LDA model
  lda_model <- lda(HDS ~ ., data = train_data)
  lda_train_pred <- predict(lda_model, train_data)$class
  lda_test_pred <- predict(lda_model, test_data)$class
```

```r
  # Misclassification rates for LDA
  lda_train_misclassification <- mean(lda_train_pred != train_data$HDS)
  lda_test_misclassification <- mean(lda_test_pred != test_data$HDS)

  # QDA model
  qda_model <- qda(HDS ~ ., data = train_data)
  qda_train_pred <- predict(qda_model, train_data)$class
  qda_test_pred <- predict(qda_model, test_data)$class

  # Misclassification rates for QDA
  qda_train_misclassification <- mean(qda_train_pred != train_data$HDS)
  qda_test_misclassification <- mean(qda_test_pred != test_data$HDS)

  # Return a summary of results
  return(data.frame(
    Model = c("LDA", "QDA"),
    Train_Misclassification = c(lda_train_misclassification, qda_train_misclassification),
    Test_Misclassification = c(lda_test_misclassification, qda_test_misclassification)
  ))
}

results <- evaluate_quiz2_model(train, test)
print(results)
```

```
##   Model Train_Misclassification Test_Misclassification
## 1   LDA               0.2185185              0.3103448
## 2   QDA               0.1777778              0.2758621
```