Compulsory exercise 1: Group 12

Evgeni Vershinin, Kristina Ødegård

2024-02-08

Problem 1

a)

Quantative: Time, income earned, horsepower of a car.
Qvalitative: Marital status, origin, Gender.

b)

KNN, LDA, QDA can be used for multi-class classifications

c)

i-iii)

The three components is called bias, variance and irreducible error.
$(E[f(X) - \hat{f}(X)])^2$ represents the bias. It is the squared expectation between our model and the true data. A high bias indicates that our model is too simple and does not capture the complexities in our data(underfitting).
$Var(\hat{f}(X))$ is the variance of our model. A high variance means our model is following our training data quite tight. It could mean our model too complex, it might follow our training data quite good, but fail on test data(overfitting).
$var(\epsilon)$ is the irreducible error. This error we have no control over. It is the inherit noise of the data, the randomness and natural variability of the true data.

ii)

Often by increasing the bias, we will reduce the variance, and vice verca. High variance could lead to higher score on training data, but if increased too much it would also decrease the score on the test data(overfitting). A high bias could make our model more interpretative and give us somehow good score on test data, but if too low it could make our model miss relevant relations between predictors and response(Underfitting). This is why we have a bias-variance trade off.

d)

The nereast neighbour for k=1 is a blue dot, so our classification is blue.
For K=3, two of the nearest neighbors are red and 1 blue. This gives 2/3 red, so it is red.
For K=5 we have 3/5 red, so it is red.

e)

i)

```
data(Boston)
data = Boston
model = lm(medv ~ rm + age,data=data)
summary(model)
```

```
##
## Call:
## lm(formula = medv ~ rm + age, data = data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -20.555  -2.882  -0.274   2.293  40.799
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -25.27740    2.85676  -8.848  < 2e-16 ***
## rm            8.40158    0.41208  20.388  < 2e-16 ***
## age          -0.07278    0.01029  -7.075 5.02e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.316 on 503 degrees of freedom
## Multiple R-squared:  0.5303, Adjusted R-squared:  0.5284
## F-statistic: 283.9 on 2 and 503 DF,  p-value: < 2.2e-16
```

ii)

```
cor_matrix = cor(data.frame(data$medv, data$rm, data$age))
print(cor_matrix)
```

```
##            data.medv    data.rm   data.age
## data.medv  1.0000000  0.6953599 -0.3769546
## data.rm    0.6953599  1.0000000 -0.2402649
## data.age  -0.3769546 -0.2402649  1.0000000
```

iii)

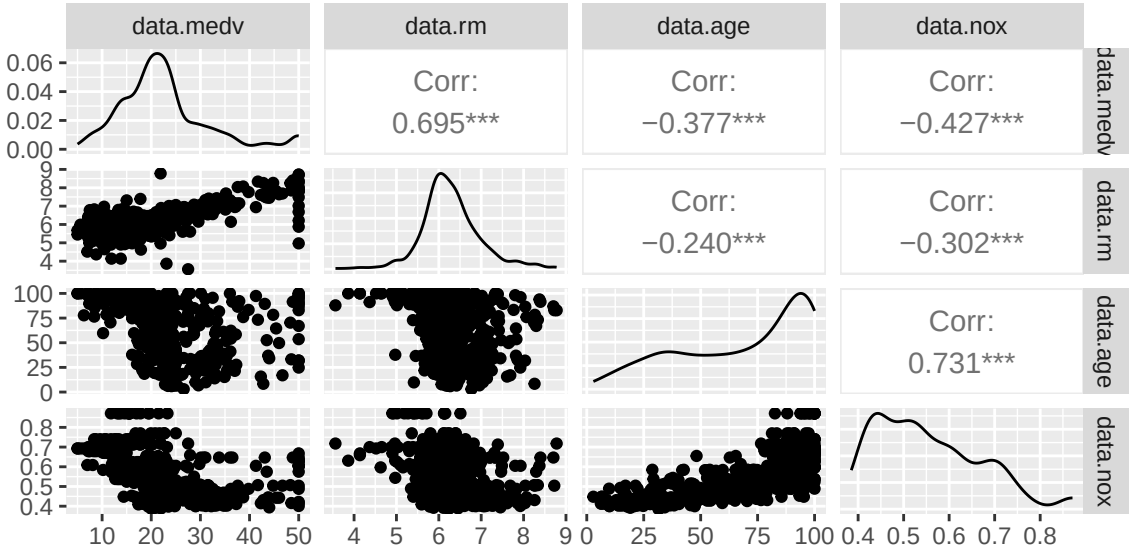
```
model2 = lm(medv ~ rm + age + nox, data=data)
summary(model2)
```

```
##
## Call:
## lm(formula = medv ~ rm + age + nox, data = data)
##
```

```
## Residuals:
##     Min      1Q  Median      3Q     Max
## -18.343  -3.168  -0.539   2.221  40.260
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -19.08308    3.33919  -5.715 1.88e-08 ***
## rm            8.12542    0.41525  19.568  < 2e-16 ***
## age          -0.03686    0.01449  -2.544 0.011269 *
## nox         -12.47877    3.58434  -3.481 0.000542 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.247 on 502 degrees of freedom
## Multiple R-squared:  0.5413, Adjusted R-squared:  0.5386
## F-statistic: 197.5 on 3 and 502 DF,  p-value: < 2.2e-16
```

iv)

```
ggpairs(data.frame(data$medv, data$rm, data$age, data$nox))
```



Looking at the correlation between Age and NOX, it is 0.731, which is quite high which suggest it has multicollinearity, which means they give some of the same information for the model.

Problem 2

a)

```
model3 = lm(medv ~ poly(rm, 2) + I(age*crim) + age + crim, data=data)
print(model3$coefficient)
```

```
##   (Intercept)  poly(rm, 2)1  poly(rm, 2)2 I(age * crim)           age
##  28.096140860 123.366564291  64.836353773   0.005791976  -0.067283041
##          crim
##  -0.796543557
```

```
valuechanged = (-10*(-0.796544)+60*(-0.067283)+0.005792 * (-10 + 60))*1000
```

If the crime is reduced by 10 and age is 60, and then considering all other factor are kept equal our median value of the property is changed by 4218.06.

b)

First thing we could do is to plot all our data with say ggpairs from GGAlly. We can then look how the data is spread and correlated. We want to carefully identify if we have any outliers. If most our points follow a specific pattern, but then one point stick out, we could carefully consider to remove it. One approach could be to look at the mean and std of a given data columns and if some point lie 2 standard deviations away from the mean we could remove them.

c)

```
model4 <- lm(medv ~ crim + age + rm, data = Boston)
sm = summary(model4)
stderr = sm$coefficients[4, "Std. Error"]
```

i)

A t-value is calculated with the follow formula: $t = \frac{Coefficient\ Estimate}{Standard\ Error} = \frac{10}{0.402}$

The t-value for rm if the estimated coefficient was 10, but with the same standard error is 24.88.

A higher coefficient with standard error kept the same gives us a higher t value and higher t values makes our estimate for statistically significant, as the p value drops. The t distribution is like a normal distribution but with a estimated variance from the sample. With a low amount of sample the t distributions curve has fatter tails and lower around the mean, as sample increases it converges to a normal distribution. So a higher absolute t value puts our score further towards the tails, strengthening our result.

ii)

```
f_test <-anova(model4)
print(f_test)
```

```
## Analysis of Variance Table
##
## Response: medv
##            Df  Sum Sq Mean Sq F value    Pr(>F)
## crim        1  6440.8  6440.8 173.458 < 2.2e-16 ***
## age         1  2809.8  2809.8  75.671 < 2.2e-16 ***
## rm          1 14825.7 14825.7 399.275 < 2.2e-16 ***
```

```
## Residuals 502 18640.0    37.1
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

An F-test was utilized to decide if at least one of the predictors is useful in predicting the response.

In this case, all three predictors has a p-value associated with the F-statisic that is $< 2.2 \cdot 10^{-16}$. Since the p-value is essentially zero, we have strong evidence to reject the null hypothesis, which would be that none of the predictors were useful in predicting the response variable. Thus, at least one of the predictors is useful in predicting the response.

iii)

```
model5 <- lm(medv ~ crim + age, data = Boston)
f_test_2 <-anova(model5)
print(f_test_2)
```

```
## Analysis of Variance Table
##
## Response: medv
##            Df Sum Sq Mean Sq F value    Pr(>F)
## crim        1   6441  6440.8  96.807 < 2.2e-16 ***
## age         1   2810  2809.8  42.232 1.954e-10 ***
## Residuals 503  33466    66.5
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The p-value associated with F values, though increased for age in this model without the predictor rm, are still significantly lower than the typical significance level of 0.05. Thus the null hypothesis can be rejected for both predictors. We conclude that the model is still useful.

d)

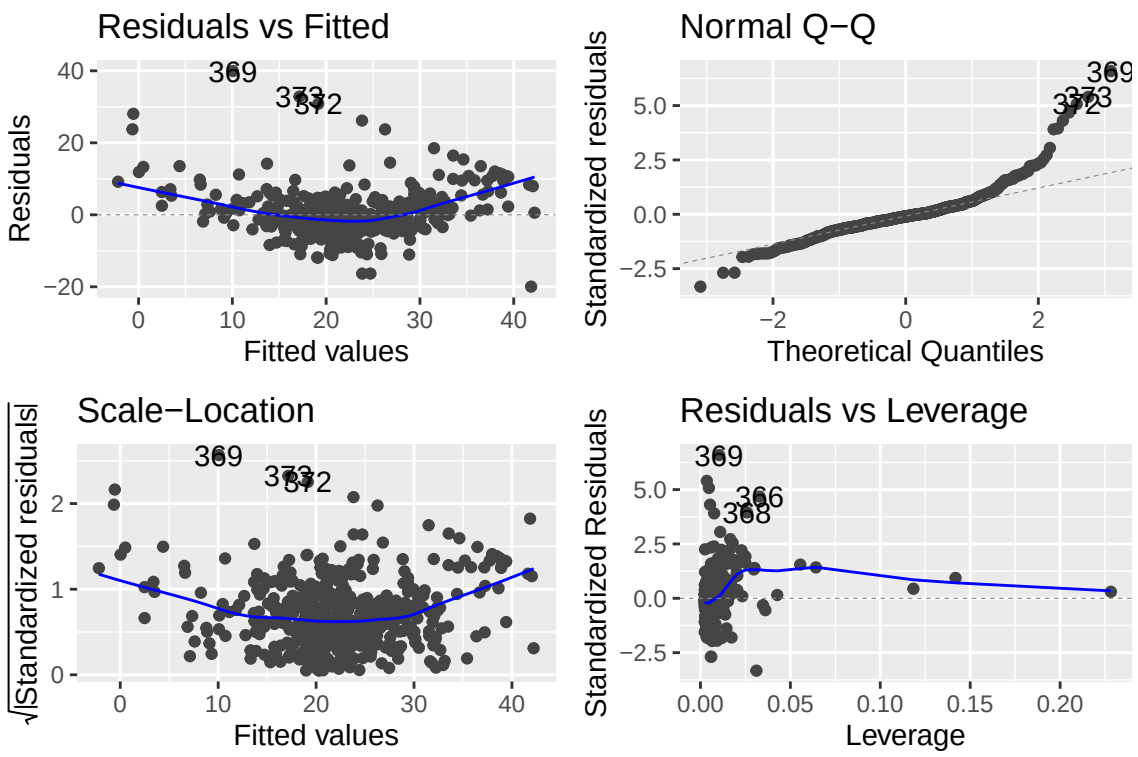
i, ii, iii)

```
new_data <- data.frame(crim = 10, age = 90, rm = 5)
# Predict the response variable using the model for the confidence interval
prediction_c <- predict(model4, newdata = new_data, interval = "confidence", level = 0.99)
# Extract lower and upper bounds from the confidence interval prediction
lower_bound_c <- prediction_c[2]
upper_bound_c <- prediction_c[3]
# Predict the response variable using the model for the prediction interval
prediction_p <- predict(model4, newdata = new_data, interval = "prediction", level = 0.99)
# Extract lower and upper bounds from the prediction interval prediction
lower_bound_p <- prediction_p[2]
upper_bound_p <- prediction_p[3]
```

Our confidence interval is [8.26, 11.24]. This is the interval for our expected value, if we would repeat this experiment a 100 times, 99 times the mean would fall into this range. Our prediction interval is [-6.08, 25.57], this one is different. This is if we would take our data point and repeat it for 100 individual true responses, 99 of those responses would fall into this range.

iv)

```
# Create diagnostic plots
autoplot(model4, which = c(1, 2, 3, 5))
```



In our first Tukey-Anscombe diagram, also known as Residuals vs Fitted is designed to visualize the relationship between the residuals. Here we would like a even spread above and under 0 across the whole x axis, In our case we u shape suggesting it is not linear or that we should try to add some quadratic terms to even out residuals and fitted values. Scale-Location is somehow similar, but instead of looking at the mean it looks at the variance. We have the same u shape here, meaning we would need to stabilize our variance somehow. Residuals vs Leverage shows how much leverage our points have. That is the further the point is to the right the higher influence it has on the coefficients. On the y axis it is how far it some from the coefficients they are. Points far to the right and from the middle line is bad. The normal Q-Q plot show if our data follows a normal distribution. Ideally we want a straight line which tells our data is normal. In our case we can see that the plot diverge to the end. Suggesting that we have some heavy tails. Here we should add more predictors to straight up the line.

e)

i)

The gender is a binary qualitative variable. The student has used two binary variables to describe what only needs one. If two binary qualitative variables were used, it would in fact allow for four different cases:

Case 1: Male and Female	$x_{male} = 1$ and $x_{female} = 1$
Case 2: Male and Not Female	$x_{male} = 1$ and $x_{female} = 0$
Case 3: Not Male and Female	$x_{male} = 0$ and $x_{female} = 1$
Case 4: Not Male and Not Female	$x_{male} = 0$ and $x_{female} = 0$

In reality, there are only two cases:

Case 1: Male (and Not Female)	$x_{gender} = 1$
Case 2: Female (and Not Male)	$x_{gender} = 0$

ii)

The formula should only have one binary quantitative variable to describe the gender. Our model would in this case make a base case for female and then if we have a male it could add something in β_1, say 10kg.

$y = \beta_0 + \beta_1 x_{gender} + \varepsilon$

where

$$x_{gender} = \begin{cases} 1 & if\ male \\ 0 & if\ female \end{cases}$$

iii)

A formula for a linear model that predicts income based on the predictor education degree with three categories {Bachelor, Master, PhD} is:

$y = \beta_0 + \beta_1 x_{master} + \beta_2 x_{Phd} + \varepsilon$

Generally we need k-1 "dummy" variables for k classes. In this case bachelor is the base case. If one has master but not PhD x_{master} would be 1, but 0 if one has PhD. Then only the x_{phd} would 1. Both would be 0 for the base case.

f) True, False, True, False

Problem 3 a) i)

```
data("titanic_train")
vars_to_be_removed <- c("PassengerId", "Name", "Ticket", "Cabin", "Embarked")
titanic_train <- titanic_train[, -which(names(titanic_train) %in% vars_to_be_removed)]
titanic_train$Pclass <- as.factor(titanic_train$Pclass)
train_idx <- sample(1:nrow(titanic_train), 0.8 * nrow(titanic_train))
titanic_test <- titanic_train[-train_idx, ]
titanic_train <- titanic_train[train_idx, ]
titanic_train <- na.omit(titanic_train)
titanic_test <- na.omit(titanic_test)
logReg = glm(Survived ~ Pclass + Sex + Age + SibSp + Parch + Fare, data = titanic_train, family = "bino
```

```r
predicted_probabilities <- predict(logReg, newdata = titanic_test, type = "response")
predicted_labels <- ifelse(predicted_probabilities > 0.5, 1, 0)
actual_labels <- titanic_test$Survived
misclassification_error <- mean(predicted_labels != actual_labels)
accuracy <- 1 - misclassification_error
print(round(accuracy,2))
```

```
## [1] 0.79
```

ii)

```r
chitest = anova(logReg,test="Chisq")
chitest
```

```
## Analysis of Deviance Table
##
## Model: binomial, link: logit
##
## Response: Survived
##
## Terms added sequentially (first to last)
##
##
##        Df Deviance Resid. Df Resid. Dev  Pr(>Chi)
## NULL                     564     763.63
## Pclass  2   69.483       562     694.15 8.163e-16 ***
## Sex     1  156.007       561     538.14 < 2.2e-16 ***
## Age     1   17.682       560     520.46 2.611e-05 ***
## SibSp   1   10.866       559     509.59 0.0009795 ***
## Parch   1    0.289       558     509.30 0.5911460
## Fare    1    1.074       557     508.23 0.3001461
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Given that th p value is less that $2.2 \cdot 10^{-16}$ suggest passenger class is really relevant predictor. In fact it is our strongest predictor. H_0 is that our predictor could just as well be 0, a low p value would break this hypothesis.

iii)

```r
newdata = data.frame(Pclass=as.factor(c(1,3)),Sex=c("female","female"),Age=c(40,40),SibSp=c(1,1),Parch=
casepredict = predict(logReg,newdata = newdata,type="response")
casepredict
```

```
##         1         2
## 0.9217204 0.4136693
```

Comparing the a woman with the all the same factors, except for class and fare price, the higher class woman has a 0.92 chance of survival, while the lower class woman has a 0.41 chance of survival.

iv)

```
ldamodel = lda(Survived ~ ., data = titanic_train)
print(ldamodel)
```

```
## Call:
## lda(Survived ~ ., data = titanic_train)
##
## Prior probabilities of groups:
##         0         1
## 0.5929204 0.4070796
##
## Group means:
##     Pclass2   Pclass3   Sexmale      Age     SibSp     Parch     Fare
## 0 0.2119403 0.6238806 0.8417910 30.74328 0.5432836 0.3820896 23.76219
## 1 0.2695652 0.2956522 0.3173913 28.71413 0.5000000 0.5217391 53.21353
##
## Coefficients of linear discriminants:
##                  LD1
## Pclass2 -0.789073856
## Pclass3 -1.504574355
## Sexmale -2.068582677
## Age     -0.026553115
## SibSp   -0.251774569
## Parch   -0.086165420
## Fare     0.001778469
```

```
predicted <- predict(ldamodel, newdata = titanic_test)
predicted_labels <- predicted$class
actual_labels <- titanic_test$Survived
misclassification_error = mean(predicted_labels != actual_labels)
accuracy <- 1 - misclassification_error
print(accuracy)
```

```
## [1] 0.7919463
```

v)

```
qdamodel = qda(Survived ~ ., data = titanic_train)
print(qdamodel)
```

```
## Call:
## qda(Survived ~ ., data = titanic_train)
##
## Prior probabilities of groups:
##         0         1
## 0.5929204 0.4070796
##
## Group means:
```

```
##     Pclass2   Pclass3   Sexmale      Age     SibSp     Parch     Fare
## 0 0.2119403 0.6238806 0.8417910 30.74328 0.5432836 0.3820896 23.76219
## 1 0.2695652 0.2956522 0.3173913 28.71413 0.5000000 0.5217391 53.21353
```

```
predicted <- predict(qdamodel, newdata = titanic_test)
predicted_labels <- predicted$class
actual_labels <- titanic_test$Survived
misclassification_error = mean(predicted_labels != actual_labels)
accuracy <- 1 - misclassification_error
print(accuracy)
```

```
## [1] 0.7986577
```

vi)

```
logreg_probs <- predict(logReg, newdata = titanic_test, type = "response")
lda_probs <- predict(ldamodel, newdata = titanic_test)$posterior[,2]
qda_probs <- predict(qdamodel, newdata = titanic_test)$posterior[,2]
actual_outcomes <- titanic_test$Survived
# ROC for Logistic Regression
roc_logreg <- roc(actual_outcomes, logreg_probs)
```

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

```
## Setting direction: controls < cases
```

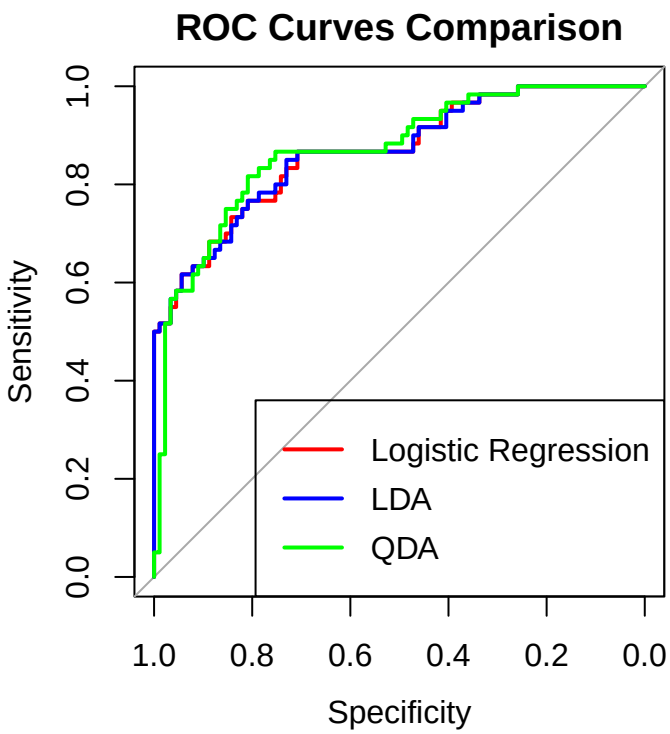
```
# ROC for LDA
roc_lda <- roc(actual_outcomes, lda_probs)
```

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

```
# ROC for QDA
roc_qda <- roc(actual_outcomes, qda_probs)
```

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

```
# Plot ROC curve for Logistic Regression
plot(roc_logreg, main="ROC Curves Comparison", col="red", xlim=c(1, 0), ylim=c(0, 1))
# Add ROC curve for LDA
lines(roc_lda, col="blue")
# Add ROC curve for QDA
lines(roc_qda, col="green")
# Add legend
legend("bottomright", legend=c("Logistic Regression", "LDA", "QDA"),
       col=c("red", "blue", "green"), lwd=2)
```




```
# AUC for Logistic Regression
auc_logreg <- roc_logreg$auc
# AUC for LDA
auc_lda <- roc_lda$auc
# AUC for QDA
auc_qda <- roc_qda$auc
# Print AUC values
cat("AUC for Logistic Regression:", auc_logreg, "\n")
```

```
## AUC for Logistic Regression: 0.8677903
```

```
cat("AUC for LDA:", auc_lda, "\n")
```

```
## AUC for LDA: 0.867603
```

```
cat("AUC for QDA:", auc_qda, "\n")
```

```
## AUC for QDA: 0.8702247
```

AUC over 0.8 is generally considered good. QDA performs slightly better than LDA and Logistic Regression. However Logistic regression is almost as accurate, but more interpretable, for one it gives clearer relationships between the predictors and they are linear.

b)

i)

The diagnostic paradigm in classification directly models the relationship between predictors and class labels, aiming to diagnose an observation's class based on its features. This approach, often deterministic, uses estimated probabilities and predefined thresholds for decision-making.Conversely, the sampling paradigm models the feature distribution within each class to understand how data for each class is generated, using this generative model for classification. It's more about capturing the data generation process for each class. Key differences include the diagnostic paradigm's focus on the direct feature-to-class relationship and use of probabilities for decision-making, whereas the sampling paradigm emphasizes understanding the data generation process and uses modeled distributions for classification.

ii)

Logistic Regression: Diagnostic paradigm, models class probability directly.
K-Nearest Neighbors (KNN): Sampling paradigm, classifies based on the local data distribution.
Naive Bayes: Sampling paradigm, uses feature distributions within classes to compute posterior probabilities.
Linear Discriminant Analysis (LDA): Sampling paradigm, assumes Gaussian feature distribution with a common covariance matrix for classification.
Quadratic Discriminant Analysis (QDA): Sampling paradigm, similar to LDA but allows class-specific covariance matrices, leading to quadratic decision boundaries

c)

i)

The decision boundary is given by setting by taking the log of each pdf and adding the log of the prior probability equal each other.

$$
\begin{aligned}
X|\{Y=1\} &\sim N(-2, 1.5^2) \\
X|\{Y=2\} &\sim N(2, 1.5^2) \\
\pi_1 &= 0.3 \\
\pi_2 &= 0.7
\end{aligned}
$$

$$
\begin{aligned}
g_1(x|\pi_1) &= g_2(X|\pi_2) \\
log(\frac{1}{\sqrt{2\pi}\sigma_1} + e^{-\frac{(x-\mu_1)^2}{2\sigma_1^2}}) + log(\pi_1) &= log(\frac{1}{\sqrt{2\pi}\sigma_2} + e^{-\frac{(x-\mu_2)^2}{2\sigma_2^2}}) + log(\pi_2) \\
log(\frac{1}{\sqrt{2\pi}1.5} + e^{-\frac{(x+2)^2}{2\cdot 1.5^2}}) + log(0.3) &= log(\frac{1}{\sqrt{2\pi}1.5} + e^{-\frac{(x-2)^2}{2\cdot 1.5^2}}) + log(0.7) \\
\frac{(x-2)^2 - (x+2)^2}{4.5} + log(0.3) - log(0.7) &= 0 \\
\frac{x^2 - 4x + 4 - x^2 - 4x - 4}{4.5} + log(\frac{0.3}{0.7}) &= 0 \\
\frac{-8x}{4.5} - 0.847 &= 0 \\
x &= -0.477
\end{aligned}
$$

For x=-0.47 is the point where the graphs cross, at this point a model would change it decision of which class our point belongs to.

```
# generate data for the two normal distributions
n_samples_class1 <- 3000
n_samples_class2 <- 7000
x1 <- rnorm(n_samples_class1, mean = -2, sd = 1.5)
x2 <- rnorm(n_samples_class2, mean = 2, sd = 1.5)
# create a data frame with the generated data
df <- data.frame(X1 = c(x1, x2), class = c(rep(1, n_samples_class1), rep(2, n_samples_class2)))
```

Problem 4

a)

iv) True

b)

i)

Corrected three mistakes:

- Corrected the number of folds from 4 to 5.
-
- Corrected the RMSE formula.

```
# Import the Boston housing price dataset
data(Boston)
# select specific variables
selected_vars <- c("crim", "rm", "age", "medv")
boston_selected <- Boston[, selected_vars]

# manually perform the 5-fold cross-validation
folds <- createFolds(boston_selected$medv, k = 5) # CORRECTED to 5
rmse_list <- list()
for (i in 1:length(folds)) {
# get the training and validation sets
train <- boston_selected[-folds[[i]], ] # CORRECTED
val <- boston_selected[-folds[[i]], ]
# fit a linear regression model
model <- lm(medv ~ ., data = train)
# compute RMSE on the validation set
pred <- predict(model, val)
rmse <- sqrt(mean((pred - val$medv)^2))  # CORRECTED squared error (RSME)
rmse <- rmse[1] # take out the value
# store rmse in rmse_list
rmse_list[[i]] <- rmse
}
# compute mean of rmse_list
rmse_mean <- mean(as.numeric(rmse_list))
cat("rmse_mean:", rmse_mean, "\n")
```

```
## rmse_mean: 6.058381
```

ii)

Changed: folds <- createFolds(boston_selected$medv, k = 5)

to:

folds <- createFolds(boston_selected$medv, k = nrow(boston_selected))

c)

i)

Corrected the following:

- Fixed the typo in the variable name "standard_erorr_of_the_median_bootstrap" to "standard_error_of_median_bootstrap".
- Changed the sample size (the second parameter of the sample function) from 1 to n. When bootstrapping, the sample size should be the same size as the original dataset, which facilitates the estimation of for example the median.
- Changed the replace variable of the sample function to TRUE to allow for an element to be selected multiple times in the same sample.
- Set seed before bootstrapping for reproducibility.

```r
# simulate data (no need to change this part)
set.seed(123) # CORRECTED
n <- 1000 # population size
dataset <- rnorm(n) # population
# bootstrap
B <- 10 # bootstrap sample size
boot <- matrix(NA, nrow = B, ncol = 1)
for (i in 1:B) {
  set.seed(123 + i) # CORRECTED
  boot[i, ] <- median(sample(dataset, n, replace = TRUE)) # CORRECTED
}
# compute the standard error of the median from the bootstrap samples
standard_error_of_the_median_bootstrap <- sd(boot) # CORRECTED
cat("standard_error_of_the_median_bootstrap:", standard_error_of_the_median_bootstrap, "\n")
```

```
## standard_error_of_the_median_bootstrap: 0.0415843
```

ii)

```r
# simulate data (no need to change this part)
set.seed(123)
n <- 1000 # population size
dataset <- rnorm(n) # population

# bootstrap with replacement
```

```r
B <- 10 # bootstrap sample size
boot_with_replacement <- matrix(NA, nrow = B, ncol = 1)
for (i in 1:B) {
  set.seed(123 + i) # setting seed for reproducibility
  boot_with_replacement[i, ] <- median(sample(dataset, n, replace = TRUE))
}

# bootstrap without replacement
boot_without_replacement <- matrix(NA, nrow = B, ncol = 1)
for (i in 1:B) {
  set.seed(123 + i) # setting seed for reproducibility
  boot_without_replacement[i, ] <- median(sample(dataset, n, replace = FALSE))
}

# compute the standard errors of the medians
standard_error_with_replacement <- sd(boot_with_replacement)
standard_error_without_replacement <- sd(boot_without_replacement)

cat("Standard Error with replacement:", standard_error_with_replacement, "\n")
```

```
## Standard Error with replacement: 0.0415843
```

```r
cat("Standard Error without replacement:", standard_error_without_replacement, "\n")
```

```
## Standard Error without replacement: 0
```

For the standard error without replacement, the same model is produced for each sample. That is because when sampling with the same size of the original data set without replacement, each sample naturally are the same. Thus, the standard error is 0 when bootstrapping without replacement. This indicates no variability in the standard error accros the different bootstrap samples, as all the samples are identical.

d)