Compulsory exercise 1: Group 49

TMA4268 Statistical Learning V2024

Jasper Steinberg, Maximilian Rønseth

23 February, 2024

Problem 1

a)

Examples of qualitative variables are eye colour (green, blue, brown), coffee roast (dark, medium, light) and sex (male, female).

Examples of quantitative variables are height (0-300cm), light (in lumen) and speed (in m/s).

b)

KNN can be used, since it is simply a method on categorizing data based on the category of its neighbors. Linear regression can not be used as it produces probabilities outside of $[0, 1]$. Logistic regression can be used as it is always bounded between $[0, 1]$. LDA can be used, since it is fundamentaly about prediction given the data conditioned on the response. Since QDA only differs from LDA in assumptions, it is also applicable.

c)

i)

The term $E[f(X) - \hat{f}(X)]^2$ corresponds to the bias of the model, i.e. the error between the real data and the model. The model variance is given by the term $\text{Var}(\hat{f}(X))$, i.e. how much the model varies when we change the data. The last term $\text{Var}(\epsilon)$ represents the irreducible error, omnipresent in all of science.

ii)

The bias-variance trade-off is that since the LHS is a fixed number and the variance of the irreducible error always exists, changing the bias will lead to change in the variance and vice versa (to maintain equality in the expected mean squared error expression).

iii)

Since the variance is given by $\text{Var}(x) = E(x^2) - E(x)^2$, we have that

$$E[(Y - \hat{Y})^2] = \text{Var}(Y - \hat{Y}) + E[(Y - \hat{Y})]^2.$$

Now we insert the expressions in terms of f to obtain

$$= \text{Var}(f(X) + \epsilon - \hat{f}(X)) + E(f(X) - \hat{f}(X) + \epsilon)^2.$$

First handling the variance term, we use that f deterministic (i.e. has 0 variance). Thus, by also the standard property of the variance,$\text{Var}(aX + Y) = a^2\text{Var}(X) + \text{Var}(Y)$, we have

$$\text{Var}(f(X) + \epsilon - \hat{f}(X)) = \text{Var}(\hat{f}(X)) + \text{Var}(\epsilon).$$

To handle the expectation term, we use that E is linear and that the irreducible error has expectation 0. Thus

$$E(f(X) - \hat{f}(X) + \epsilon)^2 = E(f(X) - \hat{f}(X))^2.$$

Combining what we found we can conclude that the original expression reduces to

$$= \text{Var}(\hat{f}(X)) + \text{Var}(\epsilon) + E(f(X) - \hat{f}(X))^2,$$

which proves the result.

d)

We see that for $K = 1$ the nearest neighbor is blue, thus the KNN classification of the black dot would be blue. For $K = 3$ the nearest neighbors are blue, red, red, thus the black dot is classified as red. For $K = 5$ we count 2 blue and 3 red dots, thus the classification yields red.

e)

i)

```r
library(MASS)
# data(Boston)

lm1 <- lm(medv ~ rm + age, data = Boston)

summary(lm1)
```

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

```
indexLM <- c(6, 7, 14)

reducedBoston <- Boston[, indexLM]

cor_matrix <- cor(reducedBoston)

cor_matrix
```

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

iii)

```
lm2 <- lm(medv ~ rm + age + nox, data = Boston)
summary(lm2)
```

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

The change could attributed to compounding effects, essentially saying that the age and amount of air pollution are correlated variables. Below we compute the correlation and see that the variables are highly correlated.

```
cor(Boston)[5, 7]
```

```
## [1] 0.7314701
```

Problem 2

a)

i)

```r
# load the boston housing price dataset
data(Boston)


# fit the linear regression model
lm_model <- lm(medv ~ crim * age + rm + I(rm^2), Boston)

summary(lm_model)
```

ii)

We first note that all terms not involving X_{crim} will cancel, since they remain the same. Thus the expression for the change in housing prices will read

$$\Delta Y_{\text{medv}} = \beta_1 \Delta X_{\text{crim}} + \beta_3 \Delta X_{\text{crim}} X_{\text{age}}.$$

Given that crime is reduced by 10, we have that the change in crime will be $\Delta X_{\text{crim}} = -10$. Furthermore, we were given that $X_{\text{age}} = 60$. To conclude we only need the coefficients, which we extract from the previous task as $\beta_1 = -0.796544$ and $\beta_3 = 0.005792$. Inserting all the numbers we find $\Delta Y_{\text{medv}} \simeq 4.49$.

b)

If we want to reduce uncertainty in the model parameters we could increase the sample size, since this would decrease the standard error of the parameter estimates.

c)

i)

If we were to use $\hat{\beta}_3 = 10$ in the formula for the t-value, we get

$$t = \frac{\hat{\beta}_3}{\text{SE}(\hat{\beta}_3)} = \frac{10}{0.40201} = 24.875.$$

###ii)

Yes, we see that the F-statistic ($= 216.1$) in the summary has an extremely low p-value, $p < 2.2e - 16$, thus we expect at least one of the predictors to be helpful in predicting the response.

iii)

```r
lm_model2 <- lm(medv ~ crim + age, data = Boston)
summary(lm_model2)
```

```
##
## Call:
## lm(formula = medv ~ crim + age, data = Boston)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -13.940  -4.991  -2.420   2.110  32.033
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 29.80067    0.97078  30.698  < 2e-16 ***
## crim        -0.31182    0.04510  -6.914 1.43e-11 ***
## age         -0.08955    0.01378  -6.499 1.95e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.157 on 503 degrees of freedom
## Multiple R-squared:  0.2166, Adjusted R-squared:  0.2134
## F-statistic: 69.52 on 2 and 503 DF,  p-value: < 2.2e-16
```

Thus, even with this reduced model, we would still find it highly probable that at least one of the predictors would be beneficial to predicting the response; this is due to the extremely low p-value associated with the F-test.

d)

```r
# We override the previous lm_model to fit the task
lm_model <- lm(medv ~ crim + age + rm, data = Boston)
```

i)

```r
new_data <- data.frame(crim = 10, age = 90, rm = 5)

conf_int <- predict(lm_model, newdata = new_data, interval = "confidence", level = 0.99)

# Lower bound
conf_int[1, 2]
```

```
## [1] 8.25632
```

```r
# Upper bound
conf_int[1, 3]
```

```
## [1] 11.23677
```

ii)

```r
pred_int <- predict(lm_model, newdata = new_data, interval = "prediction", level = 0.99)

# Lower bound
pred_int[1, 2]
```

```
## [1] -6.079652
```

```r
# Upper bound
pred_int[1, 3]
```

```
## [1] 25.57274
```

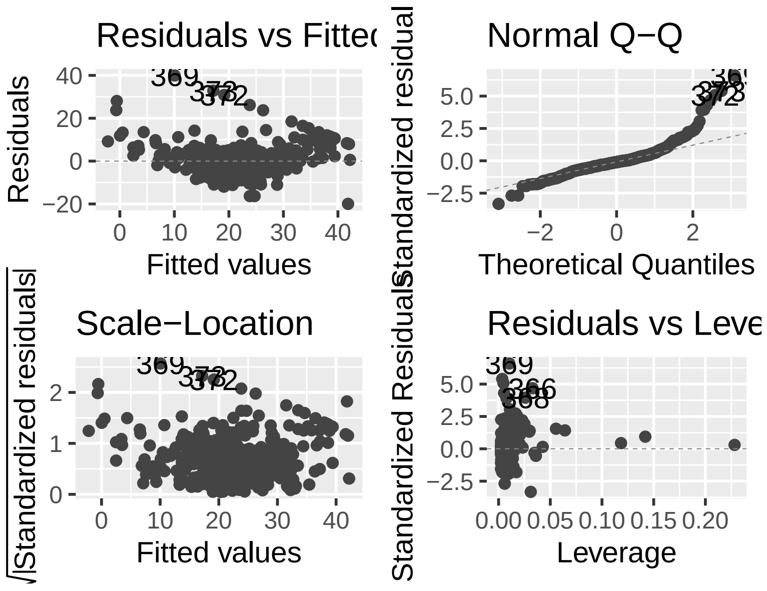
iii)

The confidence interval of level 99% is defined in the following way: given some parameter you are trying to estimate, we will in 99% of times find that the true value of the parameter lies in our interval.

A prediction interval, is defined in the following: if you sample new data, then in 99% of the cases , we would expect our new value to lie within this interval.

iv)

```r
autoplot(lm_model, smooth.colour = NA)
```



The Tukey-Anscombe plot has a slight curvature, in addition to outlier points on a line, indicating that our linear modeling assumption is problematic. The QQ-plot shows that the normal assumption is violated, as it deviates sharply from the linear line towards the end.

e)

i)

The problem with the model is that is is not identifiable. In other words we could add an arbitrary constant to β_0, and subtract it from β_1 and β_2 without changing the model. Thus we loose the ability to identify the optimal parameters.

ii)

To solve the aforementioned problem we assign one of the groups as the reference group. The correct model would then read

$$y = \begin{cases} \beta_0 + \beta_1 + \epsilon \text{ , if } x_{male} = 1 \\ \beta_0 + \epsilon \text{ , if } x_{male} = 0 \end{cases}$$

iii)

If we first define Bachelor = 1, Master = 2 and PhD = 3. We also set the reference variable as *Bachelor*, i.e. $\beta_1 = 0$. Thus the model would read

$$y = \begin{cases} \beta_0 + \epsilon, \text{ if } x_1 = 1 \\ \beta_0 + \beta_2 + \epsilon, \text{ if } x_2 = 1 \\ \beta_0 + \beta_3 + \epsilon, \text{ if } x_3 = 1 \end{cases}$$

f)

i) TRUE ii) FALSE iii) TRUE iv) FALSE

Problem 3

a)

i)

```r
set.seed(123)

# prepare the dataset into training and test datasets
# install.packages('titanic')
library(titanic)

data("titanic_train")

# remove some variables that are difficult to handle.  NB! after the removal,
# the datasets have the variable names of [Survived, Pclass, Sex, Age, SibSp,
# Parch, Fare].
vars_to_be_removed <- c("PassengerId", "Name", "Ticket", "Cabin", "Embarked")
titanic_train <- titanic_train[, -which(names(titanic_train) %in% vars_to_be_removed)]
```

```r
# make Pclass a categorical variable
titanic_train$Pclass <- as.factor(titanic_train$Pclass)

# divide the dataset into training and test datasets
train_idx <- sample(1:nrow(titanic_train), 0.8 * nrow(titanic_train))
titanic_test <- titanic_train[-train_idx, ]
titanic_train <- titanic_train[train_idx, ]

# remove the rows with missing values
titanic_train <- na.omit(titanic_train)
titanic_test <- na.omit(titanic_test)

# [ TODO] fit the logistic regression model
logReg <- glm(Survived ~ ., data = titanic_train, family = binomial)
summary(logReg)
```

```
##
## Call:
## glm(formula = Survived ~ ., family = binomial, data = titanic_train)
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  4.048694   0.553988   7.308 2.71e-13 ***
## Pclass2     -1.230152   0.353680  -3.478 0.000505 ***
## Pclass3     -2.323032   0.368839  -6.298 3.01e-10 ***
## Sexmale     -2.675098   0.248332 -10.772  < 2e-16 ***
## Age         -0.042726   0.009268  -4.610 4.02e-06 ***
## SibSp       -0.420064   0.142759  -2.942 0.003256 **
## Parch       -0.099062   0.131840  -0.751 0.452422
## Fare         0.002732   0.002796   0.977 0.328516
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 763.63  on 564  degrees of freedom
## Residual deviance: 508.23  on 557  degrees of freedom
## AIC: 524.23
##
## Number of Fisher Scoring iterations: 5
```

```r
# [ TODO] compute the accuracy on the test set

# Make predictions on test set
predicted_probs <- predict(logReg, newdata = titanic_test, type = "response")


predicted_class <- ifelse(predicted_probs > 0.5, 1, 0)

# Calculate accuracy
accuracy <- mean(predicted_class == titanic_test$Survived)

accuracy
```

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

ii)

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

```
## 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
```

We see that the p-value for Pclass is extremely low, less than machine epsilon. Thus Pclass seems like a very relevant predictor for survival.

iii)

```r
c_data <- data.frame(Pclass = factor(c(1, 3)), Sex = "female", Age = 40, SibSp = c(1,
    1), Parch = c(0, 0), Fare = c(200, 20))

predict(logReg, newdata = c_data, type = "response")
```

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

iv)

```r
LDA_Reg <- lda(Survived ~ ., data = titanic_train)
LDA_Reg
```

```
## 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
```

```r
predicted_probs2 <- predict(LDA_Reg, newdata = titanic_test, type = "response")


# Calculate accuracy
mean(predicted_probs2$class == titanic_test$Survived)
```

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

iv)

```r
QDA_Reg <- qda(Survived ~ ., data = titanic_train)
QDA_Reg
```

```
## 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
```

```r
predicted_probs3 <- predict(QDA_Reg, newdata = titanic_test, type = "response")


# Calculate accuracy
mean(predicted_probs3$class == titanic_test$Survived)
```

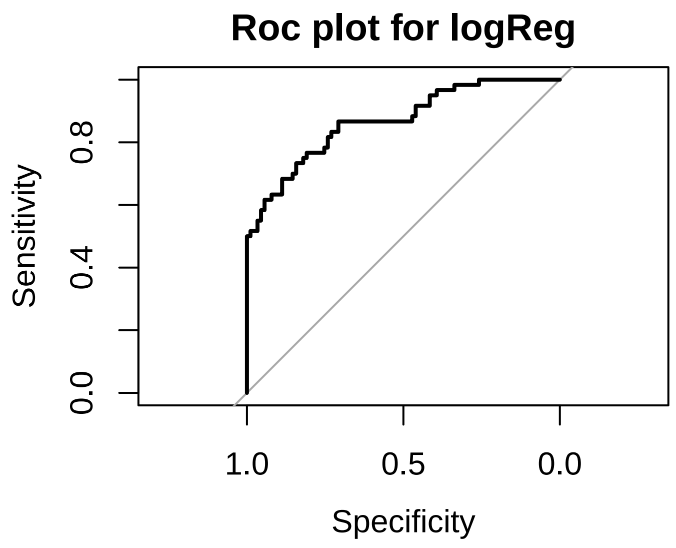
```
## [1] 0.7986577
```

vi)

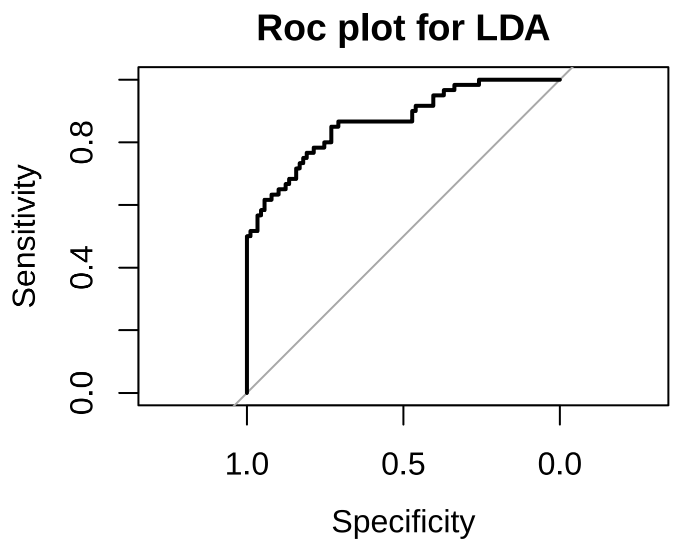
```r
library(pROC)

roc_logReg <- roc(titanic_test$Survived, predicted_probs)
roc_lda <- roc(titanic_test$Survived, predicted_probs2$posterior[, "1"])
roc_qda <- roc(titanic_test$Survived, predicted_probs3$posterior[, "1"])
```

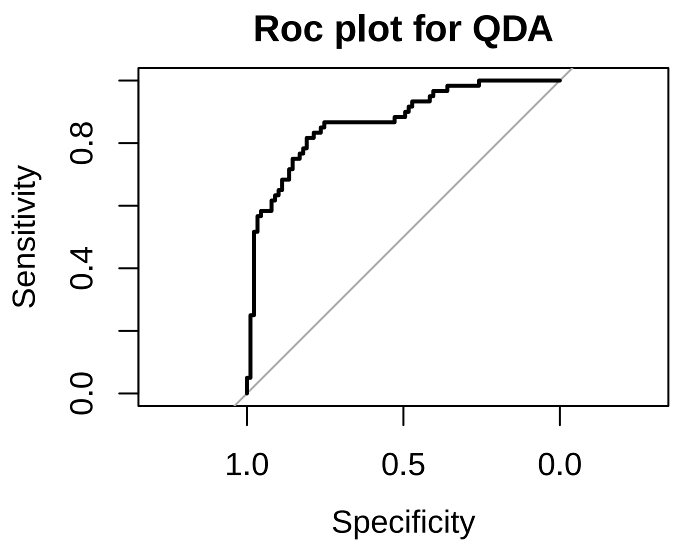
```r
plot(roc_logReg, main = "Roc plot for logReg")
```



```r
plot(roc_lda, main = "Roc plot for LDA")
```



```r
plot(roc_qda, main = "Roc plot for QDA")
```



vii)

```r
auc(roc_logReg)
```

```
## Area under the curve: 0.8678
```

```r
auc(roc_lda)
```

```
## Area under the curve: 0.8676
```

```r
auc(roc_qda)
```

```
## Area under the curve: 0.8702
```

viii)

Based on the AUC-scores, we see that the closes to 1, and thus the best was logistic regression. The worst was QDA. We note that they are very close numerically, thus it is hard to draw solid conclusions based on this test.

b)

i)

In essence the methodologies differ, in the sense that the diagnostic paradigm we directly infer on $P(Y = i \mid X = x)$, where i is some group classification, and x is the data. The sampling paradigm instead does inference on the priors and reverse condition through Bayes' rule.

ii)

Logistic regression, Naive Bayes classifier, LDA, and QDA belong to the diagnostic paradigm, while KNN belongs to the sampling paradigm.

c)

i)

Using the notation in the textbook, we can find the decision boundary by solving $\delta_1(x) = \delta_2(x)$. In our case $\mu_1 = -2$, $\mu_2 = 2$ and $\sigma_1 = \sigma_2 = 1.5^2$, thus we solve

$$x\frac{\mu_1}{\sigma_1^2} - \frac{\mu_1^2}{2\sigma_1^2} + \log(\pi_1) = x\frac{\mu_2}{\sigma_1^2} - \frac{\mu_2^2}{2\sigma_1^2} + \log(\pi_2)$$

Inserting numbers and doing the calculating we find $-\frac{9}{16}\log\left(\frac{7}{3}\right) \simeq -0.47665$.

ii)

```r
set.seed(123)  # Replace 123 with any number of your choice

# 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(X = c(x1, x2), class = c(rep(1, n_samples_class1), rep(2, n_samples_class2)))
# summary(df)

lda_model <- lda(class ~ X, data = df)
```

iii)

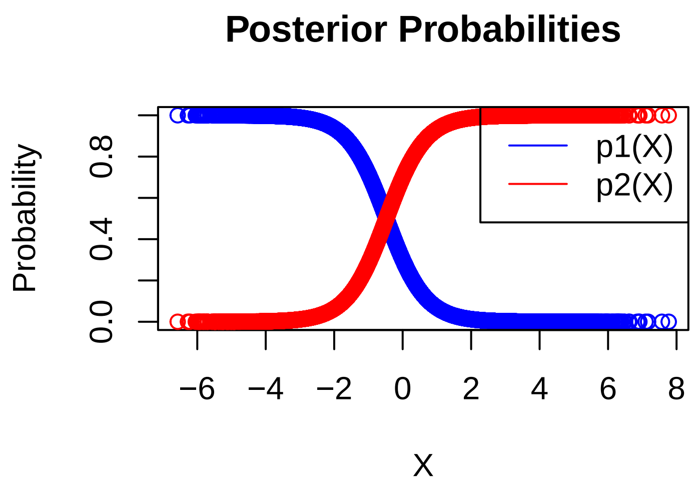
```r
pred_post <- predict(lda_model, newdata = df, type = "response")

p_1_x <- pred_post$posterior[, "1"]

p_2_x <- pred_post$posterior[, "2"]
```

iv)

```r
plot(df$X, p_1_x, col = "blue", xlab = "X", ylab = "Probability", main = "Posterior Probabilities")
points(df$X, p_2_x, col = "red")
legend("topright", legend = c("p1(X)", "p2(X)"), col = c("blue", "red"), lty = 1)
```



d)

i) TRUE ii) TRUE iii) TRUE iv) TRUE

Problem 4

a)

iv)

b)

i)

```
set.seed(123)

# Import the Boston housing price dataset

library(MASS)
library(caret)
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 = 4) Corrected: k=5
```

```r
folds <- createFolds(boston_selected$medv, k = 5)

# Size of the data, to do LOOCV set k = 506
length(boston_selected$medv)
```

```
## [1] 506
```

```r
rmse_list <- list()
for (i in 1:length(folds)) {
    # get the training and validation sets

    # Change the minus
    train <- boston_selected[-folds[[i]], ]
    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$medv))) # root mean squared error (RSME)
    # Corrected: Should be squared error
    rmse <- sqrt(mean((pred - val$medv)^2))

    rmse <- rmse[1]  # take out the value

    # store rmse in rmse_list rmse_list[[i]] <- rmse Corrected: Object is a
    # 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: 6.123494
```

ii)

To use LOOCV instead, we change $k = 5$ into $k = 506$, since its the size of he data. We commented in the code above.

c)

i)

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

# bootstrap

# In the context of the task, we accept B=10, since the population size is
# quite low.
B <- 10  # bootstrap sample size

# There is no need to make a matrix in this case, we change to a vector.
boot <- rep(NA, B)

for (i in 1:B) {

    # We should sample the size of the dataset, not 1. Also bootstraping uses
    # replacement.
    boot[i] <- median(sample(dataset, n, replace = TRUE))
}

# compute the standard error of the median from the bootstrap samples We
# correct grammar and give a more readable name.
standard_error <- sd(boot)
cat("Estimated standard error:", standard_error, "\n")
```

```
## Estimated standard error: 0.04792421
```

ii)

```r
# simulate data (no need to change this part)
set.seed(123)
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) {
    boot[i, ] <- median(sample(dataset, 1, replace = FALSE))
}
# compute the standard error of the median from the bootstrap samples
standard_erorr_of_the_median_bootstrap <- sd(boot)
cat("standard_erorr_of_the_median_bootstrap:", standard_erorr_of_the_median_bootstrap,
    "\n")
```

```
## standard_erorr_of_the_median_bootstrap: 0.9979578
```

Without the corrections, we just find the estimated variance of the sampling distribution i.e. a number close to 1. In the corrected version, we find the deviation of the expectation, i.e. a number close to 0.

d)

i) FALSE ii) TRUE iii) FALSE iv) TRUE