

TMA4268 - Project 1

Libraries used throughout the exercise

```
library(ggplot2)
library(palmerpenguins) # Contains the data set 'penguins'.
library(ggfortify)
library(tidyverse)

## -- Attaching packages ----- tidyverse 1.3.1 --

## v tibble  3.1.6      v dplyr   1.0.7
## v tidyr   1.2.0      v stringr 1.4.0
## v readr   2.1.2      v forcats 0.5.1
## v purrr   0.3.4

## -- Conflicts ----- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()     masks stats::lag()

library(GGally)

## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg      ggplot2

library(MASS)

##
## Attaching package: 'MASS'

## The following object is masked from 'package:dplyr':
##
##   select

library(class)
library(base)
library(ROSE)

## Loaded ROSE 0.0-4
```

Problem 1

a)

If we have that $y_0 = f(x_0) + \epsilon$, where ϵ is a random error independent of x with 0 mean. And x_0 is a unseen test observation we find,

$$\begin{aligned} E \left[\left(y_0 - \hat{f}(x_0) \right)^2 \right] &= E \left[\left(f(x_0) - \hat{f}(x_0) \right)^2 \right] + E [\epsilon^2] + 2E \left[\left(f(x_0) - \hat{f}(x_0) \right) \cdot \epsilon \right] \\ &= E \left[f(x_0)^2 - 2f(x_0)\hat{f}(x_0) + \hat{f}(x_0)^2 \right] + E [\epsilon^2] + 2E \left[\left(f(x_0) - \hat{f}(x_0) \right) \right] \cdot E[\epsilon] \\ &= f(x_0)^2 - 2f(x_0)E[\hat{f}(x_0)] + E[\hat{f}(x_0)^2] + Var [\epsilon] \\ &= \left(f(x_0) - E[\hat{f}(x_0)] \right)^2 + Var[\hat{f}(x_0)] + Var [\epsilon] \\ &= Bias^2 + Variance + Irreducible error \end{aligned}$$

b)

The irreducible error is a term that can't be reduced by fitting the data well. The variance is a term describing how much uncertainty our statistical model \hat{f} , how much \hat{f} changes if we change the training data. The bias is the expected difference between the true model f and our statistical model \hat{f} .

c)

- i) True
- ii) False
- iii) True
- iv) False

d)

- i) False
- ii) False
- iii) True
- iv) False

e)

- (iii) 0.76

$$\rho_{x_1, x_2} = \frac{\text{cov}(x_1, x_2)}{\sigma_{x_1} \sigma_{x_2}} = \frac{33}{\sqrt{50}\sqrt{38}} = 0.76$$

Problem 2

a)

One error made is excluding sex because of a low p-value, the p-value is the probability to observe a result equal or more extreme than the one we did, given that the null hypothesis $H_0 : \beta = 0$, so a lower p-value means the result is more statistically significant.

Another error made is deciding whether to reject the null hypothesis that the species coefficient overall is actually zero based on the p-values of the coefficients. Since the null hypothesis in this case is $H_0 : \beta_{Chinstrap} = \beta_{Gentoo} = 0$ at the same time. We need to do a F-test.

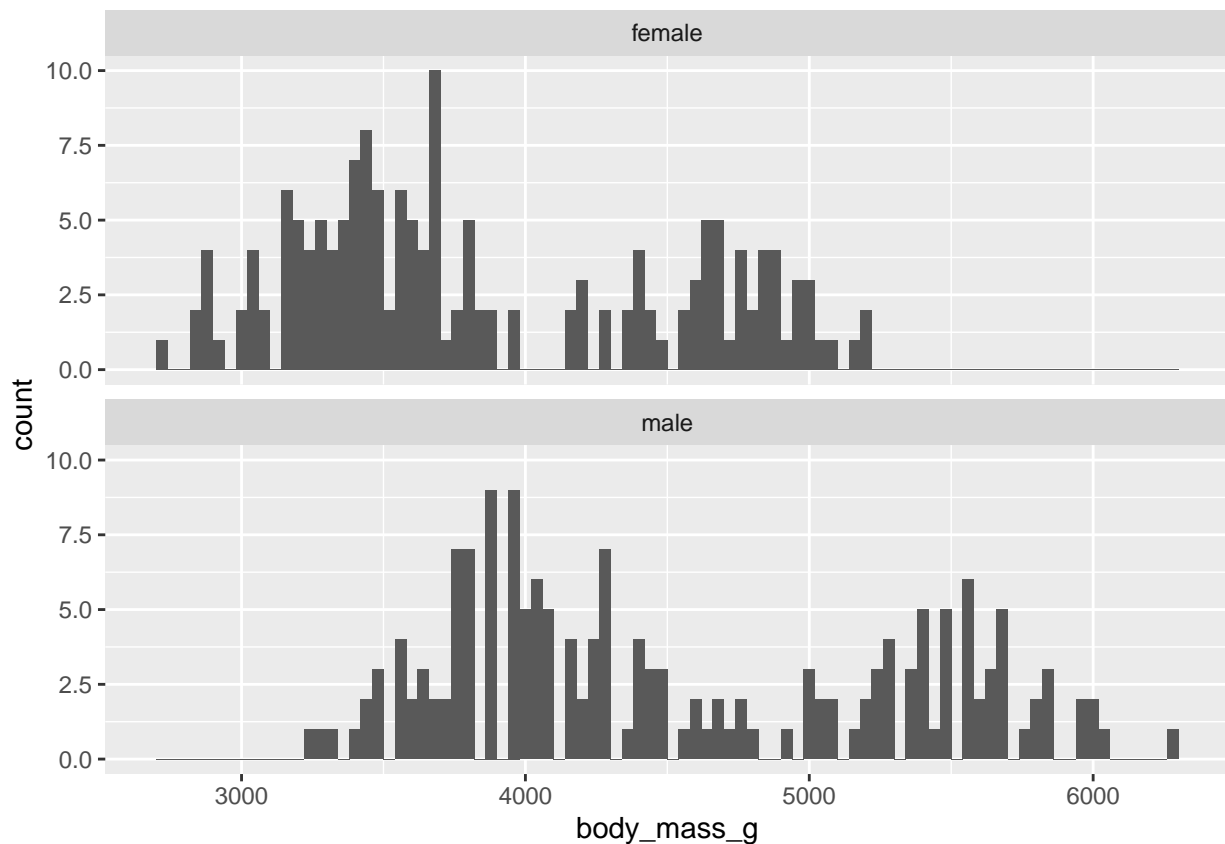
A third error made is claiming that chinstrap penguins are the largest since they have the largest coefficient. Having a large coefficient only means that for a given set of parameters a chinstrap penguin is going to be heavier than another type of penguin. And when we include interaction terms which changes the the slopes, a large coefficient only guarantees it is larger for a range of parameters near zero. And by looking at body mass of the penguins in the data set we find that in fact gentoo penguins have the highest mean body mass.

b)

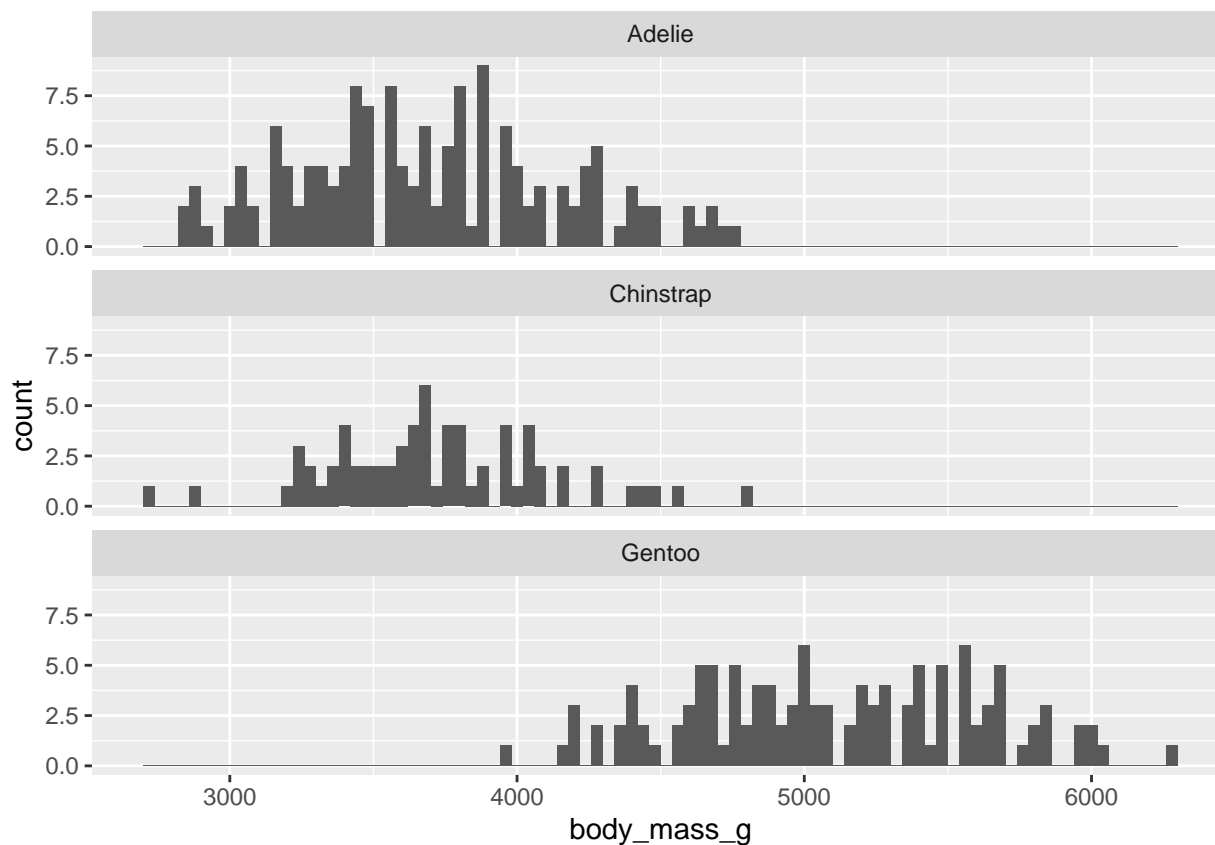
```
data(penguins)

Penguins <- subset(penguins, select = -c(island, year))
Penguins <- na.omit(Penguins)

ggplot(data = Penguins) +
  geom_histogram(binwidth = 40, mapping = aes(x = body_mass_g)) +
  facet_wrap(~ sex, nrow = 2)
```



```
ggplot(data = Penguins) +
  geom_histogram(binwidth = 40, mapping = aes(x = body_mass_g)) +
  facet_wrap(~ species, nrow = 3)
```



c)

```
Penguins <- subset(penguins, select = -c(island, year))

penguin.model <- lm(body_mass_g ~ flipper_length_mm + sex + bill_depth_mm * species,
                     data = Penguins)

summary(penguin.model)
```

```
##
## Call:
## lm(formula = body_mass_g ~ flipper_length_mm + sex + bill_depth_mm *
##     species, data = Penguins)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -751.2  -183.8    -9.8   191.1   906.9
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -1336.58    646.92  -2.066  0.039615 *
## flipper_length_mm    17.38      2.91   5.971  6.17e-09 ***
## sexmale         432.90     44.63   9.699  < 2e-16 ***
## bill_depth_mm     82.98     22.32   3.717  0.000237 ***
## speciesChinstrap 1460.15     680.39   2.146  0.032610 *
```

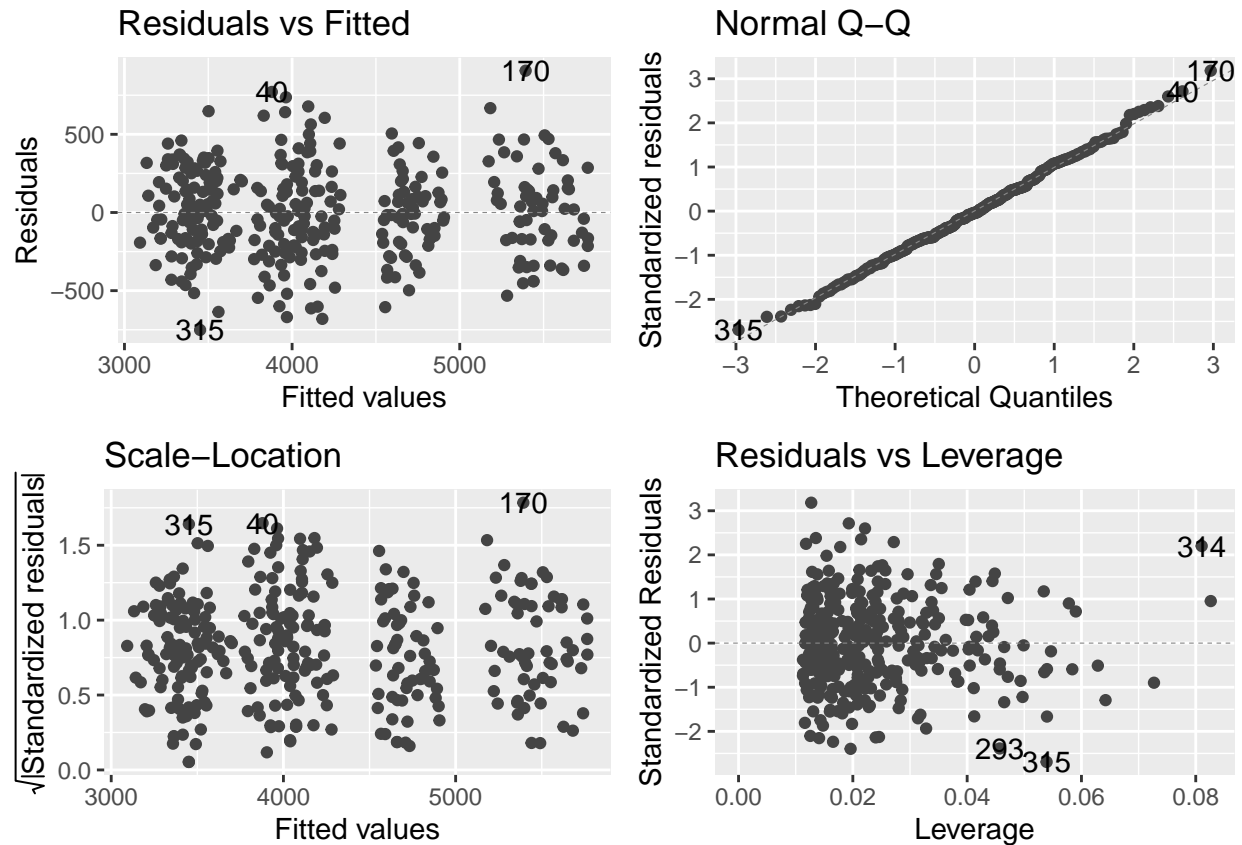
```
## speciesGentoo          644.88      542.57    1.189 0.235481
## bill_depth_mm:speciesChinstrap  -83.53      37.01   -2.257 0.024666 *
## bill_depth_mm:speciesGentoo     36.17      34.48    1.049 0.294955
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 286.8 on 325 degrees of freedom
## (11 observations deleted due to missingness)
## Multiple R-squared:  0.8758, Adjusted R-squared:  0.8732
## F-statistic: 327.5 on 7 and 325 DF,  p-value: < 2.2e-16
```

```
anova(penguin.model)
```

```
## Analysis of Variance Table
##
## Response: body_mass_g
##
##           Df      Sum Sq   Mean Sq    F value    Pr(>F)
## flipper_length_mm      1 164047703 164047703 1994.7424 < 2.2e-16 ***
## sex                    1   9416589   9416589  114.5013 < 2.2e-16 ***
## bill_depth_mm          1   3667377   3667377   44.5936 1.051e-10 ***
## species                2  10670525   5335262   64.8743 < 2.2e-16 ***
## bill_depth_mm:species   2    729458    364729    4.4349  0.01258 *
## Residuals             325  26728014    82240
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
autoplot(penguin.model, smooth.colour = NA)
```

```
## Warning: Removed 333 row(s) containing missing values (geom_path).
## Removed 333 row(s) containing missing values (geom_path).
## Removed 333 row(s) containing missing values (geom_path).
```



Problem 3

Now, the main idea is to classify the species of the penguins for a given body mass and flipper length. The dataset contains 3 type of penguin species, yet, classifying 3 penguin species is not straightforward. So, for simplicity, we define our goal to classify a penguin belonging to the species - Adelie, or not Adelie. This will give us two-class classification problem instead of three.

Since, we will use only three parameters from the dataset, it is wise to create a small dataset, and work with it. First, the variables should be converted to numeric (because knn function cannot handle the int class, and generates errors) and removes any missing observations.

```
penguins$adelie <- ifelse(penguins$species == "Adelie", 1, 0)

# Select only relevant variables and remove all rows with missing values in body
# mass, flipper length, sex or species.
Penguins_reduced <- penguins %>% dplyr::select(body_mass_g, flipper_length_mm,
adelie) %>% mutate(body_mass_g = as.numeric(body_mass_g),
flipper_length_mm = as.numeric(flipper_length_mm)) %>% drop_na()
```

Now, we need two datasets: the one to train the model, then to test the model. It is common to divide 70% of dataset as a training dataset, and the rest will be a test dataset.

```
set.seed(4268)

# 70% of the sample size for training set
```

```

training_set_size <- floor(0.7 * nrow(Penguins_reduced))
train_ind <- sample(seq_len(nrow(Penguins_reduced)), size = training_set_size)
train <- Penguins_reduced[train_ind, ]
test <- Penguins_reduced[-train_ind, ]

```

a)

```

log_reg = glm(adelle ~ body_mass_g + flipper_length_mm, data = train, family =
              "binomial")

summary(log_reg)$coef

```

i) Logistic regression

```

##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  37.7618776 5.1761640773  7.295340 2.979055e-13
## body_mass_g    0.0007120 0.0004619996  1.541127 1.232859e-01
## flipper_length_mm -0.2055804 0.0324291723 -6.339367 2.307116e-10

```

Here, we train the model using train dataset, and the result is available above.

```

prob_test_lr <- predict(log_reg, newdata = test, type = "response")

pred_test_lr <- ifelse(prob_test_lr > 0.5, 1, 0)

```

predict() gives the probabilities of the species = Adelie (class = 1). Then, we apply a cutoff value of 0.5 to those probabilities, to get predicted classes.

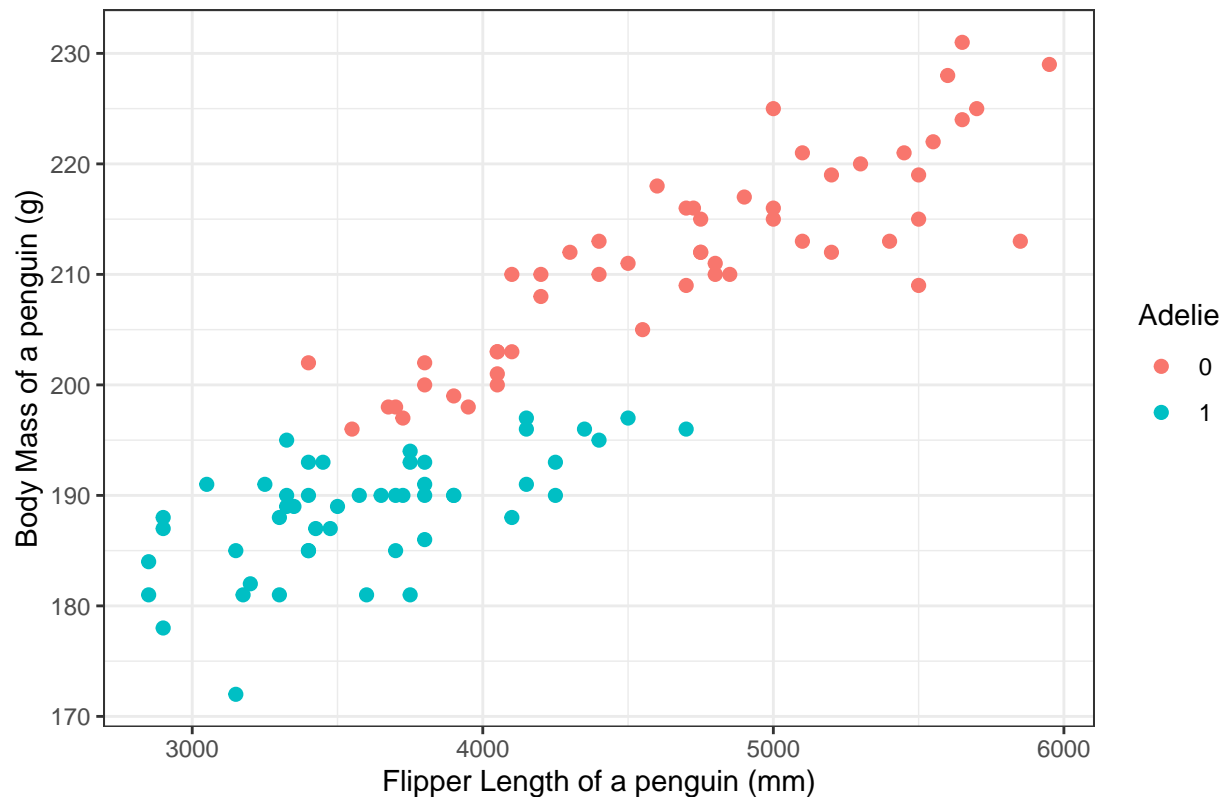
```

adelie_lr_df = bind_rows(mutate(test, pred_test_lr))
adelie_lr_df$pred_test_lr = as.factor(adelie_lr_df$pred_test_lr)

gg_p = ggplot(test, aes(x = body_mass_g, y=flipper_length_mm, color=adelie)) +
  geom_point(aes(x = body_mass_g, y=flipper_length_mm, colour=pred_test_lr),
             data=adelie_lr_df, size=2) + xlab("Flipper Length of a penguin (mm)") +
  labs(color = "Adelie") +
  ylab("Body Mass of a penguin (g)") +
  ggtitle("The classification of species using logistic regression") + theme_bw()
gg_p

```

The classification of species using logistic regression



Then, we use test dataset for testing the model. The plot illustrates the testing result.

ii) **Quadratic Discriminant Analysis** We do the same for classification using Quadratic Discriminant Analysis

```
qda_reg = qda(adelie ~ body_mass_g + flipper_length_mm, data = train)

prob_test_qda <- predict(qda_reg, newdata = test, type = "response")$posterior
pred_test_qda <- predict(qda_reg, newdata = test, type = "response")$class
```

Because QDA identifies the probabilities of having species = Adelie (class = 1), and having species = not Adelie (class = 0), predict() function will give both probabilities and already cut-off value (0.5) applied predictions. So, \$posteriors will give the probabilities, \$class will give 0.5 cut-off value applied corresponding classes.

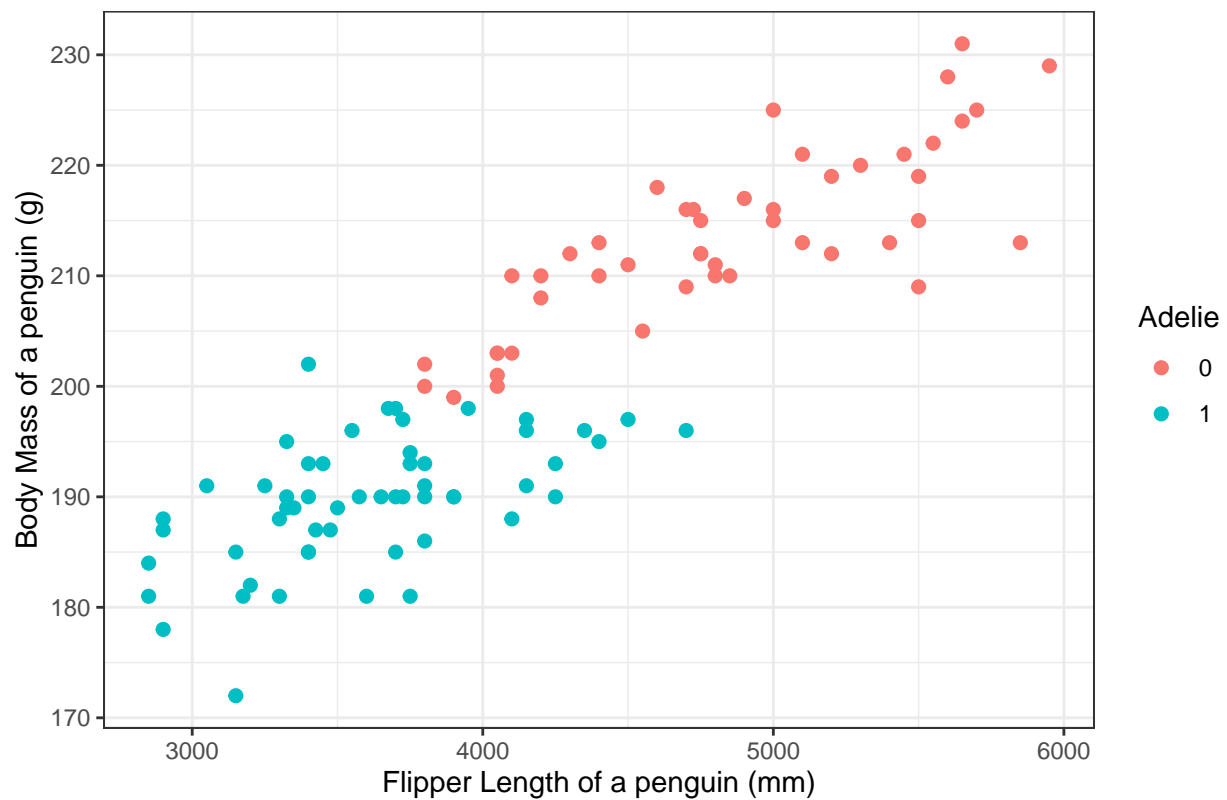
```
adelie_qda_df = bind_rows(mutate(test, pred_test_qda))
adelie_qda_df$pred_test_qda = as.factor(adelie_qda_df$pred_test_qda)

qda_plot = ggplot(test, aes(x=body_mass_g, y=flipper_length_mm, color=adelie)) +
  geom_point(aes(x = body_mass_g, y=flipper_length_mm, colour=pred_test_qda),
    data=adelie_qda_df, size=2) + xlab("Flipper Length of a penguin (mm)") +
  ylab("Body Mass of a penguin (g)") +
  labs(shape = "Adelie", color = "Adelie") +
  ggtitle("The classification of species using Quadratic Discriminat Analysis")

qda_plot
```

+ theme.

The classification of species using Quadratic Discriminat Analysis



Again, this plot illustrates the testing result.

```
knnMod = knn(train = train, test = test, cl = train$adelie, k = 25, prob = T)
knn_r <- data.frame(data.matrix(knnMod), attributes(knnMod)$prob)
```

iii) **K-nearest Neighbors** After having the model, we put the result in matrix form for better usage.

```
probKNN = ifelse(knnMod == 0, 1 - attributes(knnMod)$prob, attributes(knnMod)$prob)
predKNN = ifelse(probKNN > 0.5, 1, 0)
```

This is the way to get the probabilities of species = Adelie (class = 1). Then, again we apply 0.5 cut-off value to get the predictions.

```
adelie_knn_df = bind_rows(mutate(test, probKNN, predKNN))
adelie_knn_df$predKNN = as.factor(adelie_knn_df$predKNN)

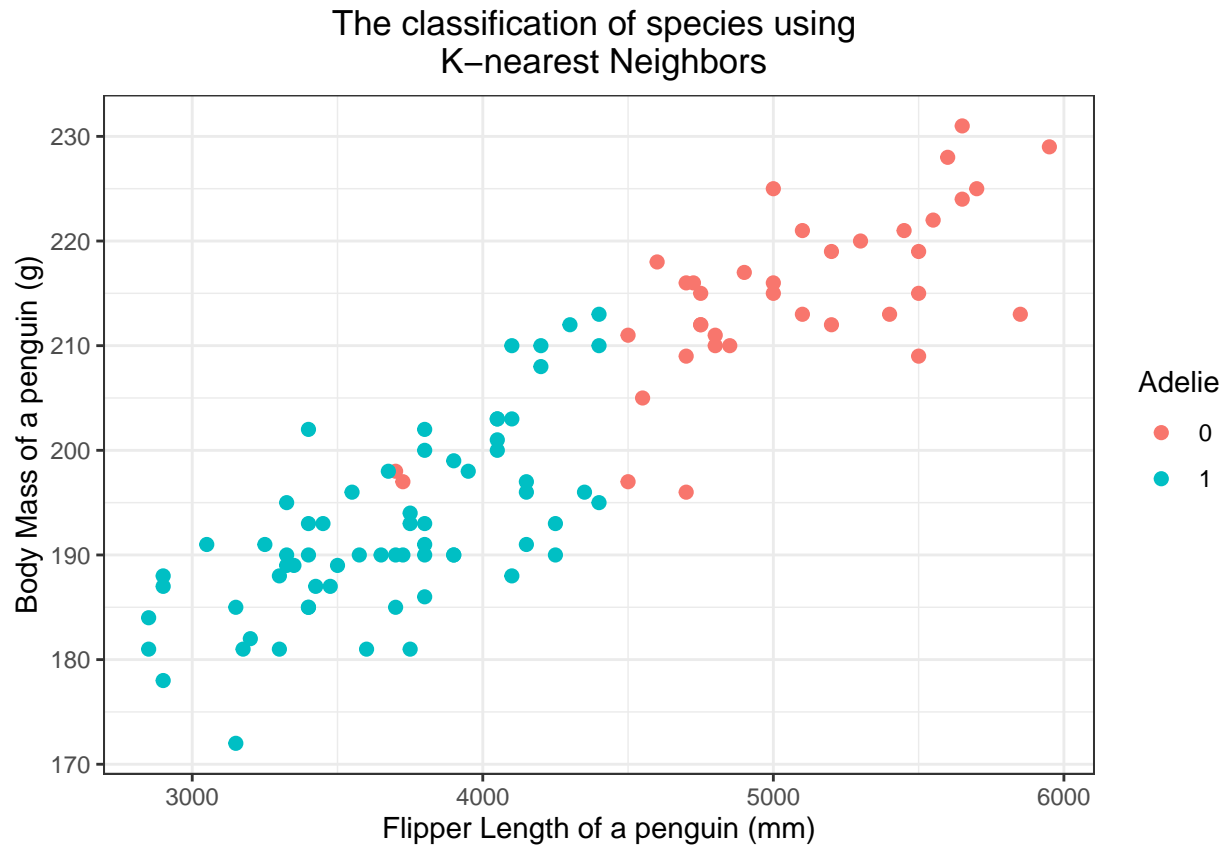
gg_knn = ggplot(test, aes(x=body_mass_g, y=flipper_length_mm,
  colour=adelie))+geom_point(aes(x=body_mass_g, y=flipper_length_mm,
  colour=predKNN), data=adelie_knn_df, size=2) +
```

```

xlab("Flipper Length of a penguin (mm)") + ylab("Body Mass of a penguin (g)") +
labs(shape = "Adelie", color = "Adelie") + ggtitle("The classification of species using \n K-nearest")
theme_bw() + theme(plot.title = element_text(hjust = 0.5))

```

gg_knn



iv) Sensitivity & specificity For Logistic regression

```

predictions_log_reg = data.frame(prob_test_lr, pred_test_lr, test[, "adelie"])
colnames(predictions_log_reg) = c("Estim. prob. of Y=1", "Predicted class",
                                "True class")
head(predictions_log_reg)

```

##	Estim. prob. of Y=1	Predicted class	True class
## 1	0.9616473	1	1
## 2	0.9028693	1	1
## 3	0.6321050	1	1
## 4	0.7859021	1	1
## 5	0.9324308	1	1
## 6	0.7688126	1	1

```

conf_mat_log_r = table(predicted = predictions_log_reg["Predicted class"][,1], true =
                        predictions_log_reg["True class"][,1] )

```

```

print("The confusion matrix is:")

## [1] "The confusion matrix is:"

conf_mat_log_r

##           true
## predicted  0  1
##           0 52  1
##           1  8 42

test_error_rate_lr = (conf_mat_log_r[1, 2] + conf_mat_log_r[2, 1])/sum(conf_mat_log_r)
print(paste( "The test error for Logistic regression -", signif(test_error_rate_lr, digits = 4)))

## [1] "The test error for Logistic regression - 0.08738"

sensitivity_lr = conf_mat_log_r[2, 2]/(conf_mat_log_r[1, 2]+conf_mat_log_r[2, 2])
specificity_lr = conf_mat_log_r[1, 1]/(conf_mat_log_r[2, 1]+conf_mat_log_r[1, 1])

print(paste( "The sensitivity - ", signif(sensitivity_lr, digits=4)))

## [1] "The sensitivity - 0.9767"

print(paste( "The specificity - ", signif(specificity_lr, digits=4)))

## [1] "The specificity - 0.8667"

For Quadratic Discriminant Analysis

predictions_qda = data.frame(pred_test_qda, test[, "adelie"])
colnames(predictions_qda) = c("Predicted class", "True class")
head(predictions_qda)

##   Predicted class True class
## 1                1         1
## 2                1         1
## 3                1         1
## 4                1         1
## 5                1         1
## 6                1         1

conf_mat_qda = table(predicted = predictions_qda["Predicted class"][,1], true =
  predictions_qda["True class"][,1] )
print("The confusion matrix is:")

## [1] "The confusion matrix is:"

```

```
conf_mat_qda
```

```
##           true
## predicted  0  1
##           0 46  1
##           1 14 42
```

```
test_error_rate_qda = (conf_mat_qda[1, 2] + conf_mat_qda[2, 1])/sum(conf_mat_qda)
```

```
print(paste( "The test error for Quadratic Discriminant Analysis -", signif(test_error_rate_qda, digits
```

```
## [1] "The test error for Quadratic Discriminant Analysis - 0.1456"
```

```
sensitivity_qda = conf_mat_qda[2, 2]/(conf_mat_qda[1, 2]+conf_mat_qda[2, 2])
```

```
specificity_qda = conf_mat_qda[1, 1]/(conf_mat_qda[2, 1]+conf_mat_qda[1, 1])
```

```
print(paste( "The sensitivity - ", signif(sensitivity_qda, digits=4)))
```

```
## [1] "The sensitivity - 0.9767"
```

```
print(paste( "The specificity - ", signif(specificity_qda, digits=4)))
```

```
## [1] "The specificity - 0.7667"
```

For K-nearest Neighbors

```
predictions_knn = data.frame(knn_r[, 2], knn_r[, 1], test[, "adelie"])
```

```
colnames(predictions_knn) = c("Estim. prob. of Y=1", "Predicted class", "True class")
```

```
head(predictions_knn)
```

```
##   Estim. prob. of Y=1 Predicted class True class
## 1           0.6000000             1           1
## 2           0.6666667             1           1
## 3           0.6800000             1           1
## 4           0.6400000             1           1
## 5           0.6800000             1           1
## 6           0.6400000             1           1
```

```
conf_mat_knn = table(predicted = predictions_knn["Predicted class"][,1], true =
  predictions_knn["True class"][,1] )
```

```
print("The confusion matrix is:")
```

```
## [1] "The confusion matrix is:"
```

```
conf_mat_knn
```

```
##           true
## predicted  0  1
##           0 35  2
##           1 25 41
```

```
test_error_rate_knn = (conf_mat_knn[1, 2] + conf_mat_knn[2, 1])/sum(conf_mat_knn)
print(paste( "The test error for K-nearest Neighbors -", signif(test_error_rate_knn, digits = 4)))
```

```
## [1] "The test error for K-nearest Neighbors - 0.2621"
```

```
sensitivity_knn = conf_mat_knn[2, 2]/(conf_mat_knn[1, 2]+conf_mat_knn[2, 2])
specificity_knn = conf_mat_knn[1, 1]/(conf_mat_knn[2, 1]+conf_mat_knn[1, 1])

print(paste( "The sensitivity - ", signif(sensitivity_knn, digits=4)))
```

```
## [1] "The sensitivity - 0.9535"
```

```
print(paste( "The specificity - ", signif(specificity_knn, digits=4)))
```

```
## [1] "The specificity - 0.5833"
```

b)

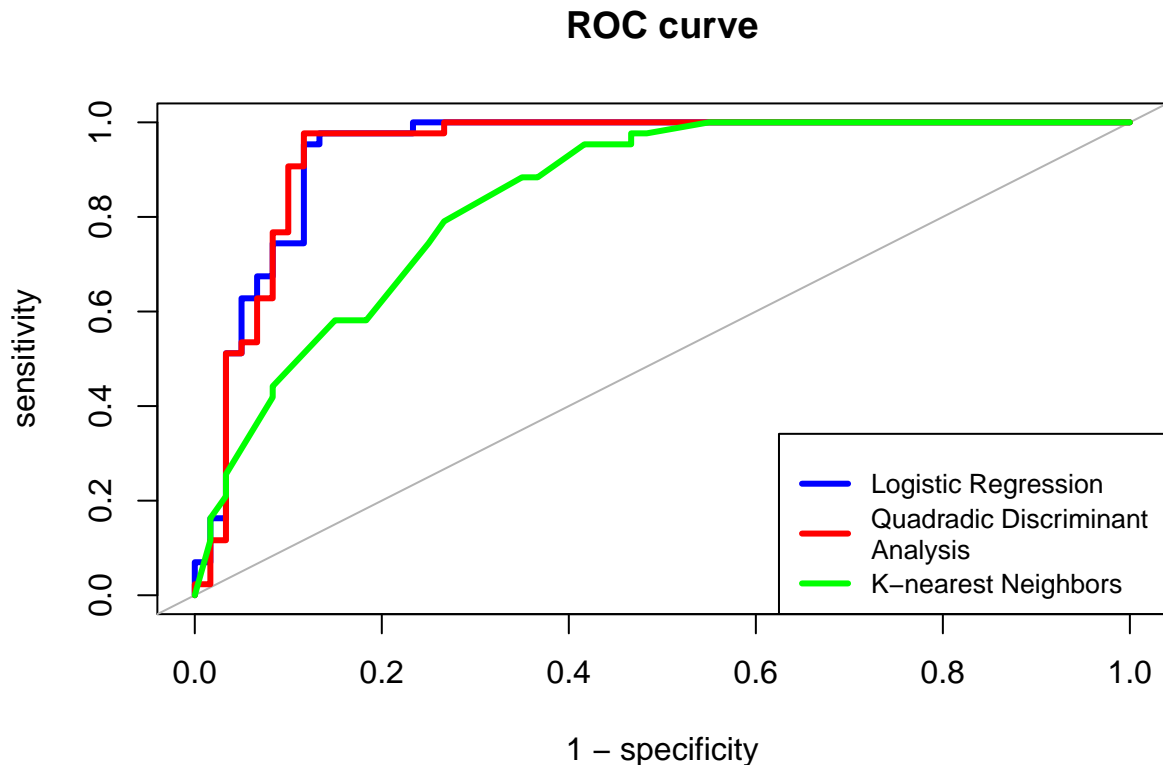
```
lr_roc = roc.curve(test$adelie, prob_test_lr, col = "blue", lwd=3,
                   xlab = "1 - specificity", ylab = "sensitivity")

qda_roc = roc.curve(test$adelie, prob_test_qda[,2], col = "red", lwd=3, add = T)

probKNN = ifelse(knnMod == 0, 1 - attributes(knnMod)$prob, attributes(knnMod)$prob)

knn_roc = roc.curve(test$adelie, probKNN, col = "green", lwd=3, add = T)

legend("bottomright", legend = c("Logistic Regression", "Quadratic Discriminant \nAnalysis", "K-nearest
                                col = c("blue", "red", "green"))
```



i)

Areas Under Curves:

```
print(paste( "AUC of Logistic Regression ", signif(lr_roc$auc, digits = 4)))
```

```
## [1] "AUC of Logistic Regression  0.9391"
```

```
print(paste( "AUC of Quadratic Discriminant Analysis ", signif(qda_roc$auc, digits = 4)))
```

```
## [1] "AUC of Quadratic Discriminant Analysis  0.938"
```

```
print(paste( "AUC of K-nearest Neighbors ", signif(knn_roc$auc, digits = 4)))
```

```
## [1] "AUC of K-nearest Neighbors  0.8417"
```

ii) ROC curve - Receiver operating characteristics curve displays the relation between False Positive rate and True Positive rate for all possible threshold values (0 to 1). Since True Positive rate is $\frac{\text{True Positives}}{\text{All Positives}}$, it is expressed as sensitivity, and False Positive rate as 1 - Specificity. If the model is ideal model, the sensitivity = 1, and specificity = 1, so, the curve will be on the top left. While a straight line will represent a model with random guesses on the outcome.

In order to check the overall performance of the model, AUC (Area Under Curve) values are compared, meaning that if AUC value is high, that means the curve is closer to the ideal situation.

The comparison of the models we have: Logistic regression, Quadratic Discriminant Analysis, K-nearest Neighbors

When we look at the ROC Curves, we clearly see that the model that performs worse than others is K-nearest Neighbors. This is proved by AUC area. (AUC area of KNN = 0.8417).

The performances of Logistic regression and Quadratic Discriminant Analysis are almost the same. Yet, there is a slight difference in AUC areas. Based on AUC areas, Logistic regression performs a bit better than Quadratic Discriminant Analysis (AUC area of LogReg = 0.9391, AUC area of QDA = 0.938). However, the overall conclusion is Logistic Regression and Quadratic Discriminant Analysis are similarly good models for interpreting the data.

iii) If the task is to create an interpretable model, the model that has to be chosen is Logistic Regression. The reason is Logistic Regression outputs a table that contains all the necessary information to analyze the relation between two variable. The features like Intercept, p-value, t-value have key roles in interpretation.

c)

$$\text{odds} = \frac{p_i}{1-p_i} = \frac{P(Y_i=1|X=x)}{P(Y_i=0|X=x)} = \exp(\beta_0) \cdot \exp(\beta_1 x_1) \cdot \exp(\beta_2 x_2)$$

$$\text{odds ratio} = \frac{\text{odds}[Y_i=1|X=x_1+1000]}{\text{odds}[Y_i=1|X=x_1]} = \frac{\exp(\beta_0) \cdot \exp(\beta_1(x_1+1000)) \cdot \exp(\beta_2 x_2)}{\exp(\beta_0) \cdot \exp(\beta_1 x_1) \cdot \exp(\beta_2 x_2)} = \exp(\beta_1 * 1000)$$

We find β_1 from coefficient table of Logistic regression

```
summary(log_reg)$coef
```

```
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  37.7618776  5.1761640773  7.295340 2.979055e-13
## body_mass_g    0.0007120  0.0004619996  1.541127 1.232859e-01
## flipper_length_mm -0.2055804  0.0324291723 -6.339367 2.307116e-10
```

So, $\beta_1 = 0.000712$

```
print(paste( "The odds will be multiplied by ", signif(exp(coef(log_reg)[2]*1000), digits = 4)))
```

```
## [1] "The odds will be multiplied by  2.038"
```

The answer is III.

d)

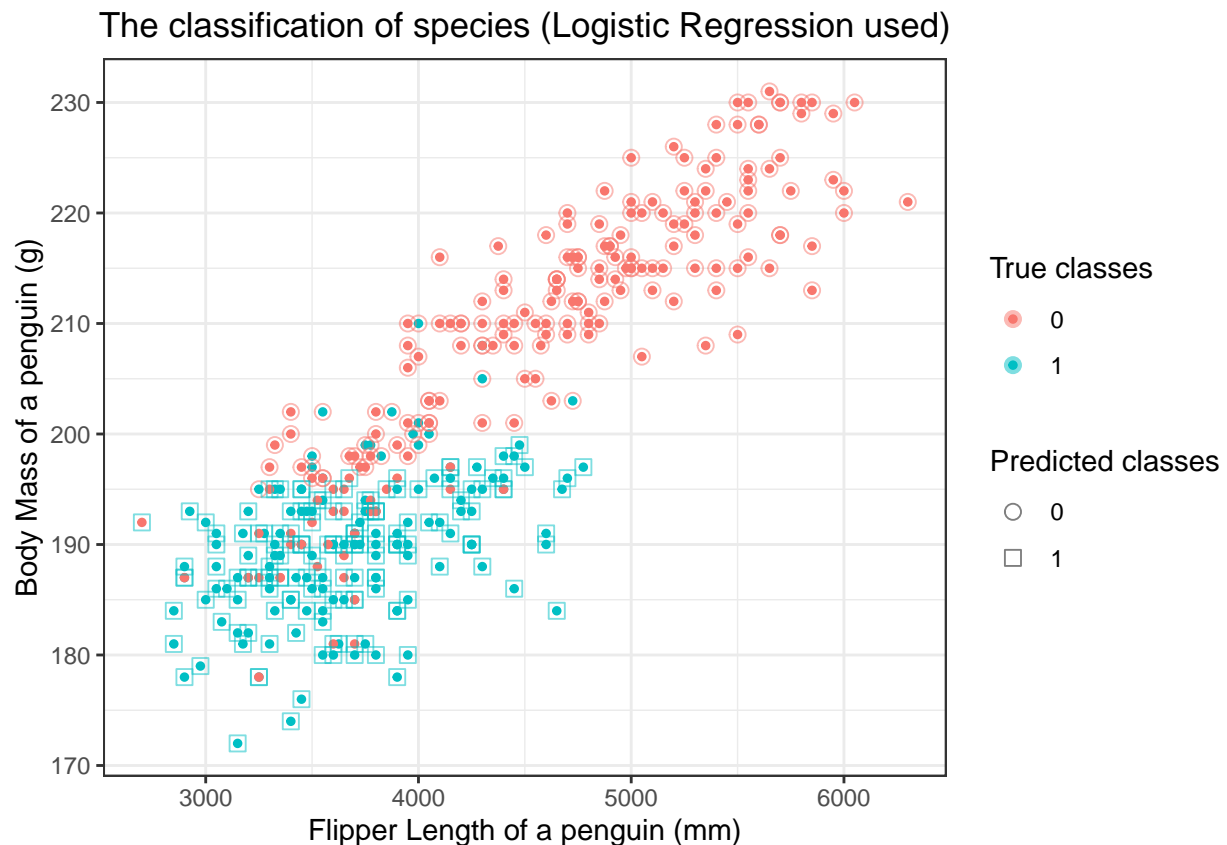
```
prob_whole_lr <- predict(log_reg, newdata = Penguins_reduced, type = "response")
pred_whole_lr <- ifelse(prob_whole_lr > 0.5, 1, 0)
pred_whole_lr_sh = pred_whole_lr

whole_lr_df = bind_rows(mutate(Penguins_reduced, pred_whole_lr, pred_whole_lr_sh))
whole_lr_df$pred_whole_lr = as.factor(whole_lr_df$pred_whole_lr)
whole_lr_df$pred_whole_lr_sh = as.factor(whole_lr_df$pred_whole_lr_sh)
Penguins_reduced$adelie = as.factor(Penguins_reduced$adelie)
```

```
data = ggplot(Penguins_reduced, aes(x=body_mass_g, y=flipper_length_mm, color=adelie)) +
  geom_point(size=1) + theme_bw()
```

```
data_plot = data +
  geom_point(aes(x = body_mass_g, y=flipper_length_mm, colour = pred_whole_lr,
    shape = pred_whole_lr_sh), data=whole_lr_df,
    size=2.5, alpha = 1/2) + scale_shape_manual(values=c(1, 0))+
  labs(shape = "Predicted classes", color = "True classes") +
  xlab("Flipper Length of a penguin (mm)") +
  ylab("Body Mass of a penguin (g)") +
  ggtitle("The classification of species (Logistic Regression used)") +
  theme_bw() + theme(plot.title = element_text(hjust = 0.5))
```

data_plot



Problem 4

a)

Answers for the questions are as follows: (i) True; (ii) False; (iii) True; (iv) False;

b)

Heart disease (chd) probability for a non-smoking male with systolic blood pressure (sbp) 150 is 0.10096. The R code for the problem solution is provided below:

```
#Accessing the data

id<-"1chRpybM5cJn4Eow3-_xwDKPKyddL9M2N"
d.chd<-read.csv(sprintf("https://docs.google.com/uc?id=%s&export=download",id))

#Creating a predicting model

glm_fit = glm(chd~sex+sbp+smoking, data = d.chd, family = binomial)

#Predicting the disease probability

temp_d <- data.frame(sex=1, sbp=150, smoking=0)
predict(glm_fit,temp_d,type = 'response')
```

```
##          1
## 0.10096
```

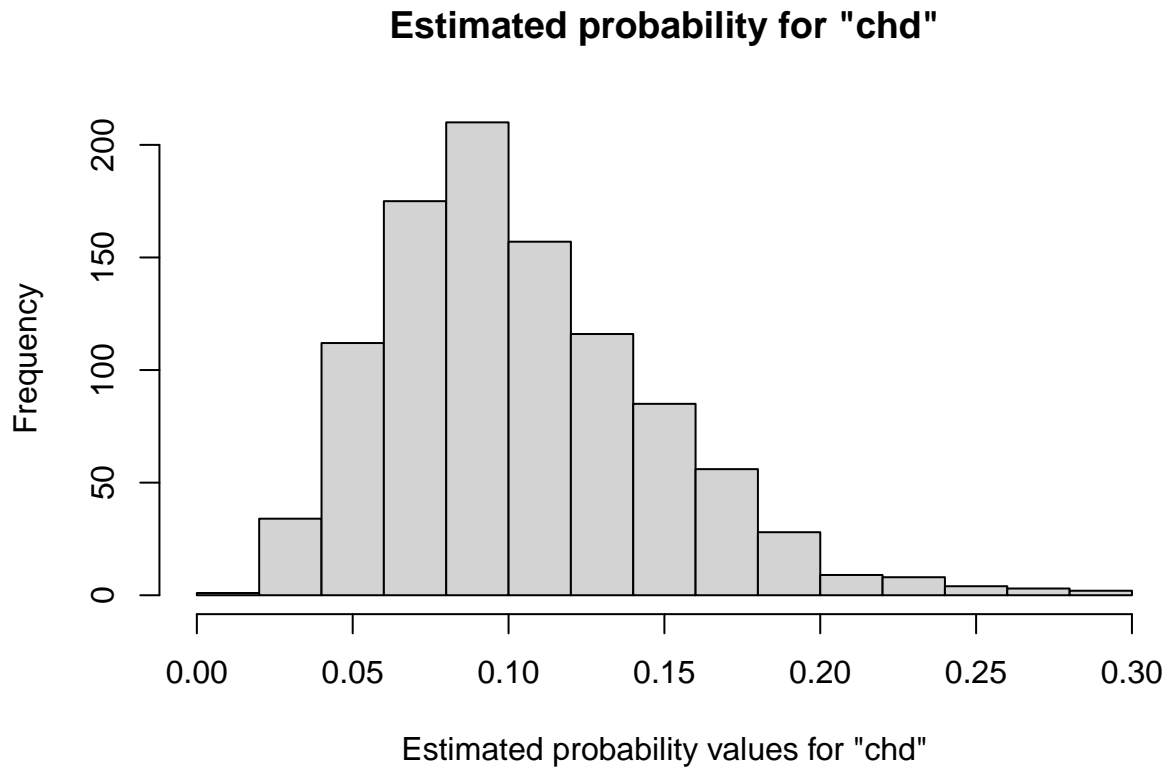
c)

The R code generating solution for the first problem can be found below. As one can see most of the prediction values are between 0.05 and 0.15.

```
#Create function of interest
func_interest <- function(used_data){
  glm_fit = glm(chd~sex+sbp+smoking, data = used_data, family = binomial)
  predict(glm_fit,data.frame(sex=1, sbp=150, smoking=0),type = 'response')
}

#Implement bootstrap for 1000 iterations and save the results in A
B = 1000
A = replicate(B,0)
for (i in 1:B){
  A[i]=func_interest(d.chd[sample(1:nrow(d.chd), 500, replace = T),])
}

hist(A, main = 'Estimated probability for "chd"', xlab = 'Estimated probability values for "chd"')
```



The standard error derived from predictions above equals to 0.00134.

One can find the R code generated for solving the problem [here](#).

```
#Calculating standard error
std_error_func <- function(x) sd(x) / sqrt(length(x))
Std_error <- std_error_func(A)
print(Std_error)
```

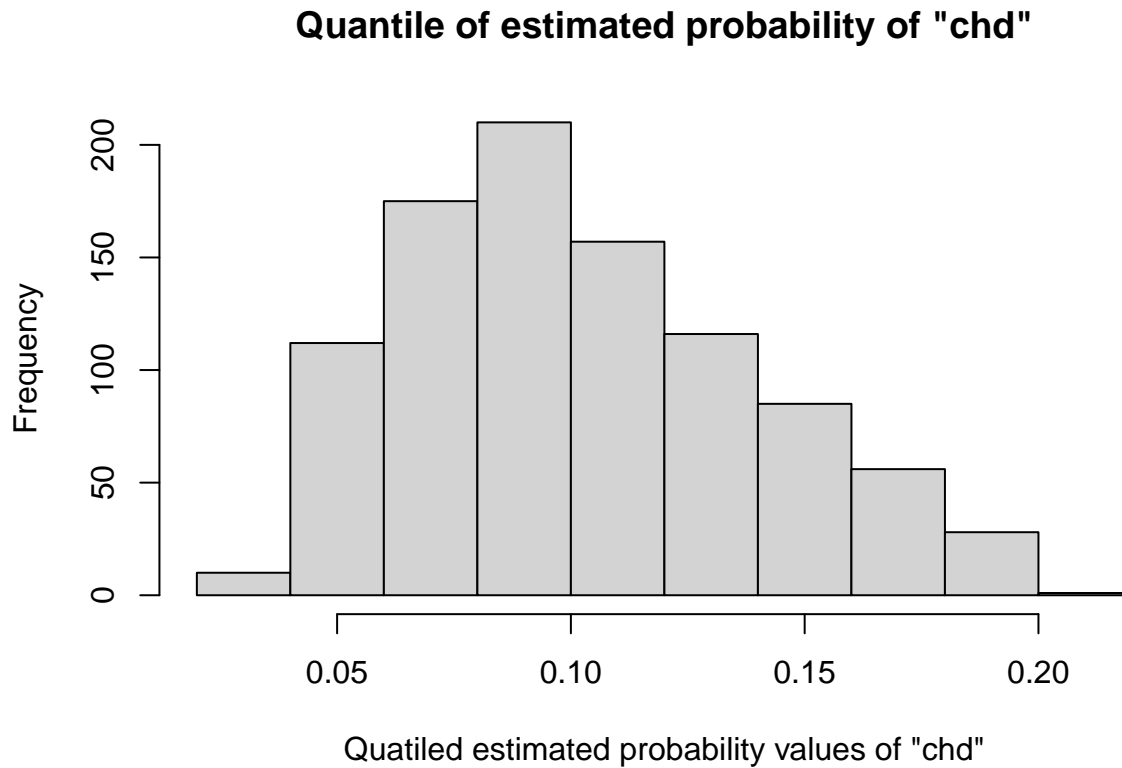
```
## [1] 0.001387751
```

The 95 % quantile interval for the bootstrap samples lies between 0.03915905 and 0.20153356. One can see how they distributed in the histogram below.

```
#Deriving quantiles and sorting bootstrap results within them
q = quantile(A, probs = c(0.025, 0.975))

After_q <- A[A < q[2]]
After_q <- After_q[After_q > q[1]]

hist(After_q, main = 'Quantile of estimated probability of "chd"', xlab = 'Quatiled estimated probabili
```



The expected probability value is 0.1036039. In my opinion the range of plausible values are from 0.06 to 0.12. The R code for solving this problem is here

```
#Calculating values probabilities and expected probability value
```

```
EP = 0
for(i in 1:length(After_q)){
  EP = EP+After_q[i]*((1/950))
}
print(EP)
```

```
## [1] 0.1025979
```

d)

The answers for the questions in problem d) are as follows. (i) True, (ii) False, (iii) True, (iv) False.