Machine Learning Homework 2

Latera Tesfaye Olana

08 February, 2023

Question 1a: The following table provides summary of the given data.

Table 1: Summary table of the data

diagnosis	observations
Benign	357
Malignant	212

The given data has 357 Benign outcome observations and 212 Malignant observations. The data has two predictors (p=2). Overall, the data has 569 observations. This data has no missing values.

Question 1b:
The data is divided into two, training and test with each containing 70.3 and 29.7, respectively.

Question 1c: My premises for answering this question is if the data points are well-separated into distinct groups and the groups form clear clusters, it is likely that the predictors radius and texture can accurately predict the outcome diagnosis. This indicates that there is a strong relationship between radius, texture and diagnosis, and a well-trained classification model can make accurate predictions.

On the other hand, if the data points are randomly scattered and there are no clear patterns, it is unlikely that the predictors radius, texture can accurately predict the outcome diagnosis.

This indicates that there is no significant relationship between radius, texture and diagnosis, and it would be difficult for a classification model to make accurate predictions.

Given this, as shown in figures 1 and 2, there seems to be a difference in distribution (grouping) of the outcome across both predictors. For instance, there seems to be a big difference in diagnosis status at a radius of about 14. However, it seems that the texture variable by itself might not be particularly useful. This is due to with different values of texture it is seems so as if the outcome (diagnosis) can be either Benign or Malignant. It does not seem it has a clear pattern.

Figure 3 and 4, also provide more information. Between the two outcome groups the radius density plot changes significantly, however, the changes for texture is small.

Finally, I would say both can be predictors, but radius surely has more magnitude as compared to texture. The texture variable by itself might not be particularly useful.

Question 1d: The table 2 shows the summary of the GLM fit using diagnosis as an outcome variable, whereas, radius and texture as predictors.

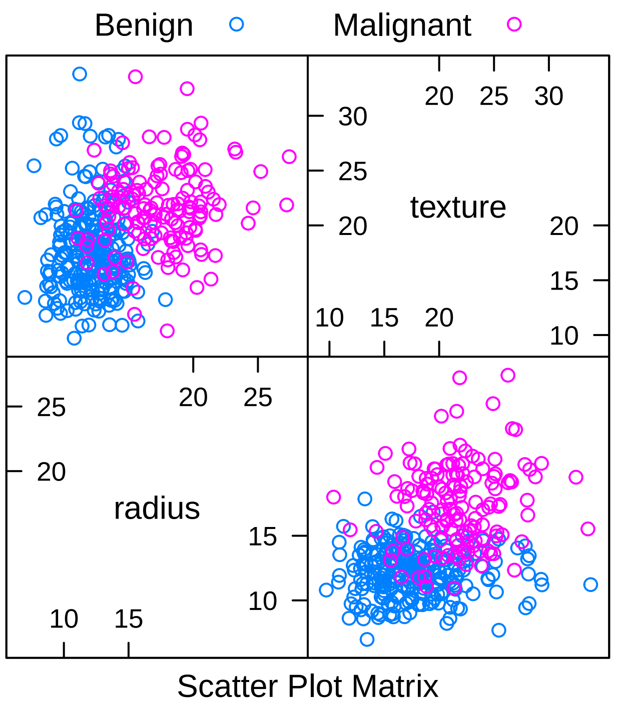

Figure 1: Scatter plot of the data

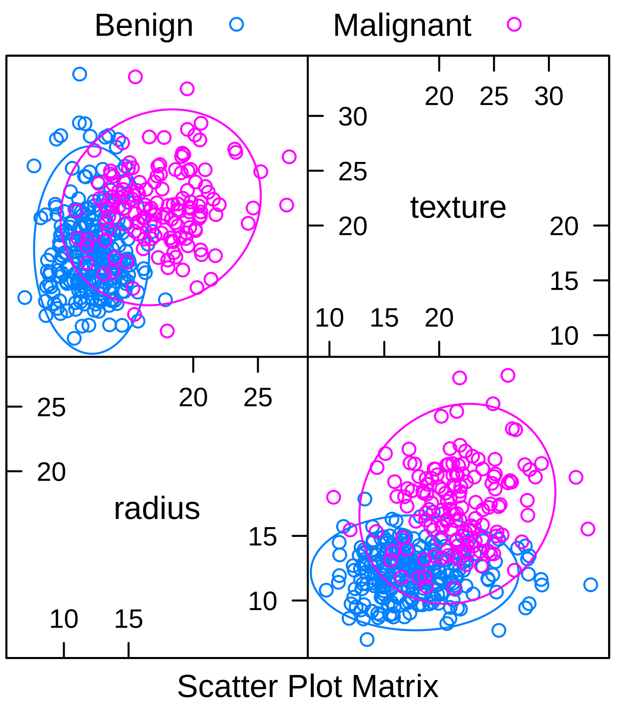

Figure 2: Scatter plot of the data with ellipse

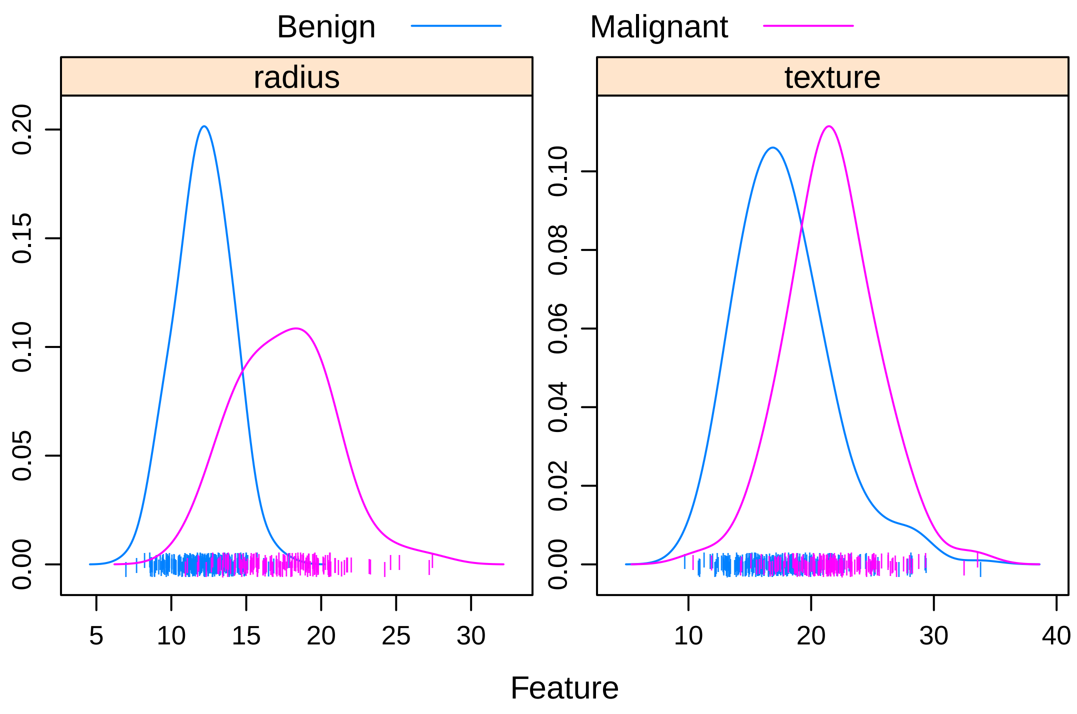

Figure 3: Density plot of outcome groups over the given features

Table 2: GLM fit summary

Variable	Coefficient	Standard Error	z-statistcis	P-value	95% CI Lower	95% CI Higher
(Intercept)	-19.383590	2.015225	-9.618572	0	-23.687124	-15.750290
radius	0.983655	0.111256	8.841395	0	0.783737	1.221895
texture	0.241305	0.045449	5.309381	0	0.154847	0.333955

The coefficients in the GLM fit represent the amount of change in the log-odds of the diagnosis outcome (Benign vs Malignant) for a one unit increase in the predictor variable. The (Intercept) coefficient represents the log-odds of the diagnosis outcome when both the predictors (radius and texture) are at their minimum value.

From the table, we can see that:

- The coefficient for the predictor variable radius is 0.98, which means that for a one unit increase in radius, the log-odds of the diagnosis outcome being malignant increase by 0.98 unit. This indicates that a higher radius value is associated with a higher probability of the diagnosis outcome being malignant. Since the p-value is very small we can reject the null hypothesis diagnosis is not dependent on radius.
- The coefficient for the predictor variable texture is 0.24, which means that for a one unit increase in texture, the log-odds of the diagnosis outcome being malignant increase by 0.24 unit. This indicates that a higher texture value is associated with a higher probability of the diagnosis outcome being malignant unit. Since the p-value is very small we can reject the null hypothesis diagnosis is not dependent on texture.

Question 1e: The formula is given by:

$$p(Y = k|X = x) = \frac{e^{(\beta_0+\beta_1 x_1+\beta_2 x_2+\cdots+\beta_p x_p)}}{1 + e^{(\beta_0+\beta_1 x_1+\beta_2 x_2+\cdots+\beta_p x_p)}}$$

Using the above summary table and formula we can re-written this question as:

$$p(Y = m|X = 10, 12) = \frac{e^{-19.38+0.98*10+0.24*12}}{1 + e^{-19.38+0.98*10+0.24*12}}$$

.

Finally, P(Y = M | (X1,X2) = (10, 12)) = 0.00129.

Using *predict()* function and manual calculation the predicted values are the same.

Question 1f: Let's start from predicted probabilities:

$$\hat{p}(x) = \hat{P}(Y = "Malignant"|X = x)$$

This following formula shows how the classification rule using 0.5 cutoff threshold.

$$\hat{D}(x) = \begin{cases} 1 & \hat{p}(x) > 0.5 \\ 0 & x \leq 0.5 \end{cases}$$

Where $\hat{p}(x) = \hat{P}(Y = "Malignant"|X = x)$.

Question 1f: The prediction accuracy of train data is 0.9, whereas the prediction accuracy for test data is 0.88. The following tables show the confusion matrix for both train and test datasets.

Table 3: Confussion matrix for train data

	Actual Benign	Actual Malignant
Benign	242	26
Malignant	15	117

Table 4: Confussion matrix for train data

	Percentages
Accuracy	0.89750
Sensitivity	0.81818
Specificity	0.94163
Pos Pred Value	0.88636
Neg Pred Value	0.90299
Prevalence	0.35750
Detection Rate	0.29250

Looking at the confusion matrix for train dataset from the above table, the true positive for classification was 242. The false negative is 15; the true negative 117 and finally the false positive is 26.

Table 5: Confussion matrix for test data

	Actual Benign	Actual Malignant
Benign	94	15
Malignant	6	54

Table 6: Confussion matrix for test data

	Percentages
Accuracy	0.87574
Sensitivity	0.78261
Specificity	0.94000
Pos Pred Value	0.90000
Neg Pred Value	0.86239
Prevalence	0.40828
Detection Rate	0.31953

Looking at the confusion matrix for test dataset from the above table, the true positive for classification was 94. The false negative is 6; the true negative 54 and finally the false positive is 15.
Comparing the train and test results, we can see that the true positive and the negative decreased rapidly on the test data. Accordingly, the sensitivity on test data is smaller than that of the train data. In both cases their accuracy and specificity are relatively close.

Question 1g: We can first create a prediction function with:

$$\hat{D}(x) = \begin{cases} 1 & \hat{p}(x) > c \\ 0 & x \leq c \end{cases}$$

Where c is the different cutoff levels.

Usually, sensitivity decreases as the cutoff increases. Conversely, specificity increases as the cutoff increases. Therefore, as it is shown in figure 4, as the threshold increases we can increasingly classify individuals who are benign. However, the mis-classification of those who are Malignant increases.

Question 1h: The figure 5 shows receiver operating characteristic curve (ROC) for test data.

Question 1i: As shown in the receiver operating characteristic curve (ROC) the area under the curve (AUC) is about 0.9535.

Question 2a: We fitted Linear Discriminant Analysis (LDA) using MASS library. The following table (table 7 and 8) shows both prior probabilities of groups and mean groups.

Table 7: Group means

	radius	texture
Benign	12.146	17.769
Malignant	17.479	21.661

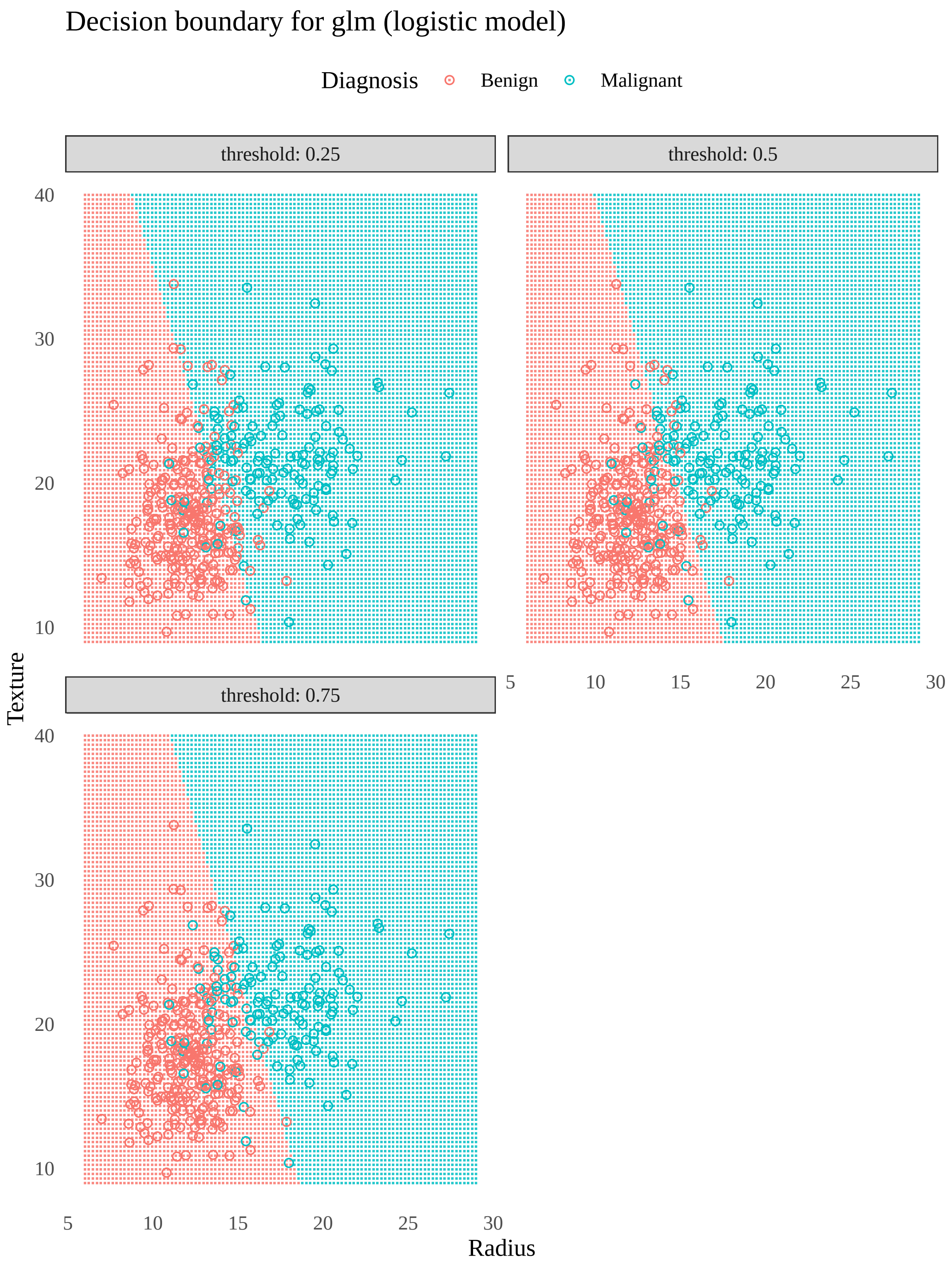

Figure 4: Boundary plot for logistic regression

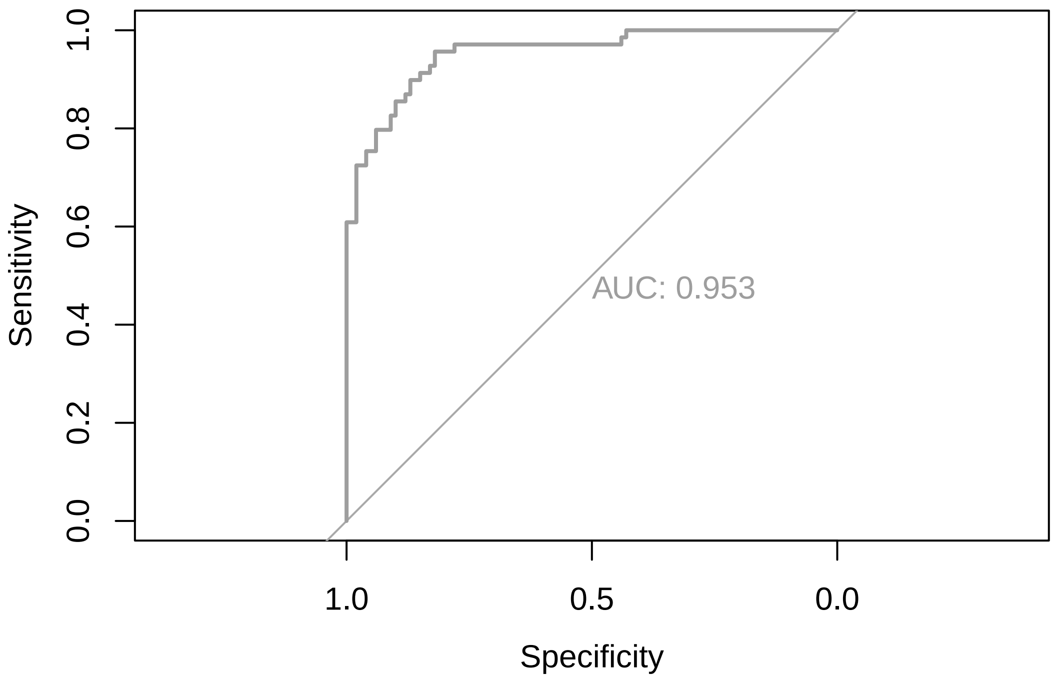

Figure 5: ROC plot for logistic classifcation

Table 8: Prior probabilities of groups

	Prior probabilities
Benign	0.642
Malignant	0.358

In LDA linear discriminant analysis, the prior probabilities of groups refer to the estimated probability of each group occurring in the population before any observations are made. This is calculated based on the number of observations of each group in the training data. Accordingly, about 35.8% of this observation are Malignant.

Group means, on the other hand, refer to the average value of each feature for each group in the training data. These are used to calculate the probability that a new observation belongs to a particular group based on the difference between the new observation's feature values and the group means.

The prior probabilities and group means are related to the posterior probabilities, which are the probabilities of each group given a new observation. The prior probabilities are combined with the group means and the difference between the new observation's feature values and the group means to calculate the posterior probabilities. The posterior probabilities represent the updated probability of each group occurring given the new observation and can be used for classification.

Question 2b: Using the Bayes rule (as indicated above), we computed the predicted outcome $\hat{Y}$ from the predicted posterior probabilities, both on the training and test set. As shown in the following table,

Table 9: Confussion matrix for train data

	Actual Benign	Actual Malignant
Benign	247	33
Malignant	10	110

Table 10: Confussion matrix for train data

	Percentages
Accuracy	0.89250
Sensitivity	0.76923
Specificity	0.96109
Pos Pred Value	0.91667
Neg Pred Value	0.88214
Prevalence	0.35750
Detection Rate	0.27500

the confusion matrix for train dataset, the true positive for classification is 247. The false negative is 10; the true negative is 110 and finally the false positive is 33.

Table 11: Confussion matrix for test data

	Actual Benign	Actual Malignant
Benign	98	21
Malignant	2	48

Table 12: Confussion matrix for train data

	Percentages
Accuracy	0.86391
Sensitivity	0.69565
Specificity	0.98000
Pos Pred Value	0.96000
Neg Pred Value	0.82353
Prevalence	0.40828
Detection Rate	0.28402

In addition, according to the confusion matrix for the test dataset, the true positive classification is 98. Whereas, the false negative is 2; the true negative is 48 and finally the false positive is 21.
Comparing the train and test results, we can see that the true positive and the negative decreased on the test data. There is considerable difference in sensitivity for both train and test dataset, with test dataset having smaller sensitivity. However, on both dataset the accuracy seems very close. The smaller sensitivity on the test data compared to the train data means that the model is not correctly identifying all the positive cases (malignant) on the test data.

Question 2c:

Similar to that of logistic fit, we can see here sensitivity decreases as the cutoff is increased. Conversely, specificity increases as the cutoff increases. Therefore, as it is shown figure 6, as the threshold increases

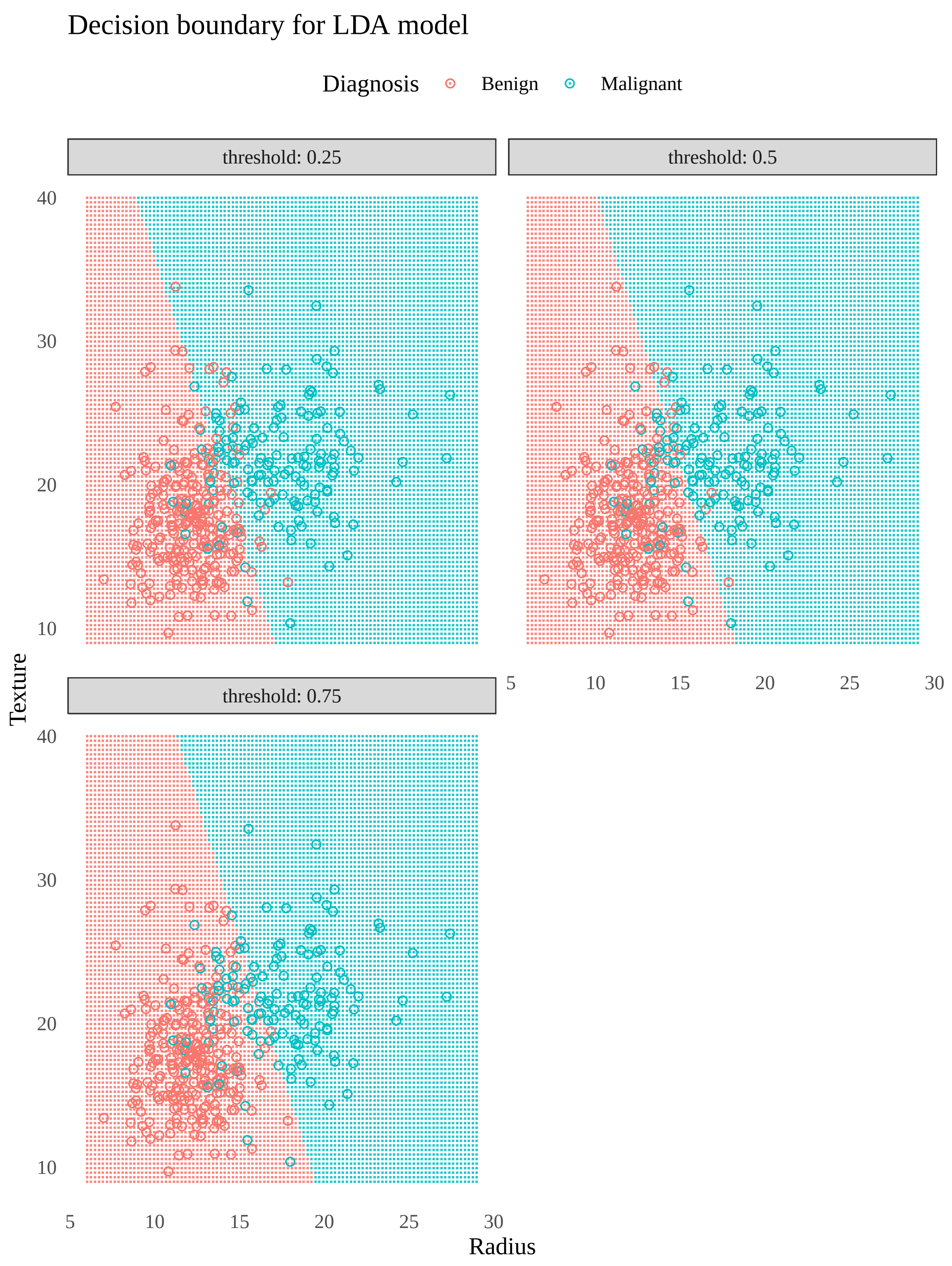

Figure 6: Boundary plot for LDA

we can increasingly classify individuals who are benign. However, the mis-classification of those who are Malignant increases.

Question 2d: Figure 7 shows ROC curve for LDA model.

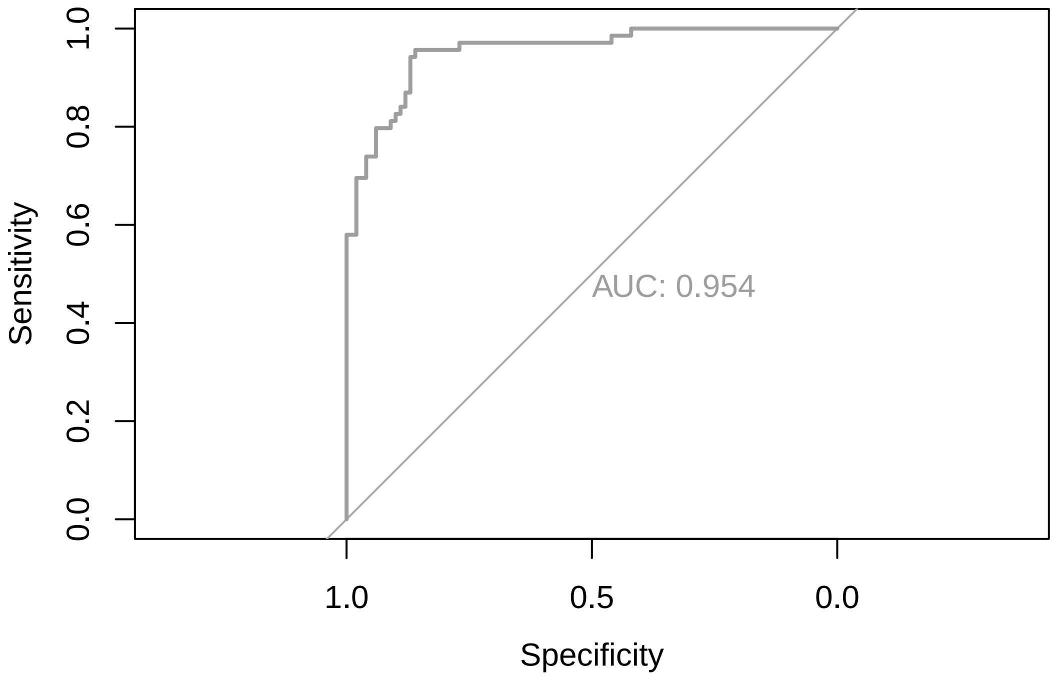

Figure 7: ROC curve for LDA model

Question 2e: As shown in the receiver operating characteristic curve (ROC) the area under the curve (AUC) is about 0.9536. There seems to be a slight increase in accuracy comparing logistic regression and LDA.

Question 3: We fitted Quadratic Discriminant Analysis (QDA) using MASS library. The following table shows both prior probabilities of groups and mean groups.

Table 13: Group means

	radius	texture
Benign	12.146	17.769
Malignant	17.479	21.661

Table 14: Prior probabilities of groups

	Prior probabilities
Benign	0.642
Malignant	0.358

In quadratic discriminant analysis (QDA), the “prior probabilities of groups” are the probabilities of each class or group in the data, before taking into account any other information. These prior probabilities are used to calculate the posterior probabilities, which are the probabilities of each class given the observed data.

The "group means" in QDA refer to the mean or average of the features for each class or group. These group means are used to calculate the discriminant functions, which are the functions that determine which class a new observation belongs to based on its feature values. The group means and the discriminant functions are related to the posterior probabilities because they are used to calculate the probabilities of each class given the observed data.

The posterior probabilities are calculated by combining the prior probabilities, the group means, and the discriminant functions. In QDA, each class has its own discriminant function, so the posterior probabilities are calculated differently for each class. This allows QDA to handle non-linear relationships between the features and the classes, making it a more flexible method compared to linear discriminant analysis (LDA).

Using the Bayes rule (as indicated above), we computed the predicted outcome $\hat{Y}$ from the predicted posterior probabilities, both on the training and test set. As shown in the following table of

Table 15: Confussion matrix for train data

	Actual Benign	Actual Malignant
Benign	244	31
Malignant	13	112

Table 16: Confussion matrix for train data

	Percentages
Accuracy	0.89000
Sensitivity	0.78322
Specificity	0.94942
Pos Pred Value	0.89600
Neg Pred Value	0.88727
Prevalence	0.35750
Detection Rate	0.28000

the confusion matrix for train dataset, the true positive for classification is 244. The false negative is 13; the true negative 112 and finally, the false positive is 31.

Table 17: Confussion matrix for train data

	Actual Benign	Actual Malignant
Benign	98	20
Malignant	2	49

Table 18: Confussion matrix for train data

	Percentages
Accuracy	0.86982
Sensitivity	0.71014
Specificity	0.98000
Pos Pred Value	0.96078
Neg Pred Value	0.83051

	Percentages
Prevalence	0.40828
Detection Rate	0.28994

As shown in the above table, the confusion matrix for test dataset, the true positive for classification is 98. The false negative is 2; the true negative 49 and finally the false positive is 20. Comparing the train and test results, we can see that the true positive and the negative decreased on the test data.

Figure 8 shows boundary plot for QDA model. As shown in the figure as the threshold increases the model tends to mis-classify more malignant cases.

The figure 9 shows ROC curve for QDA model.

As shown in the receiver operating characteristic curve (ROC) the area under the curve (AUC) is about 0.9562. There seems to be a slight increase in accuracy using QDA as compared to logistic regression and LDA. This suggests that the quadratic form assumed by QDA may capture the true relationship more accurately than the LDA and logistic regression.

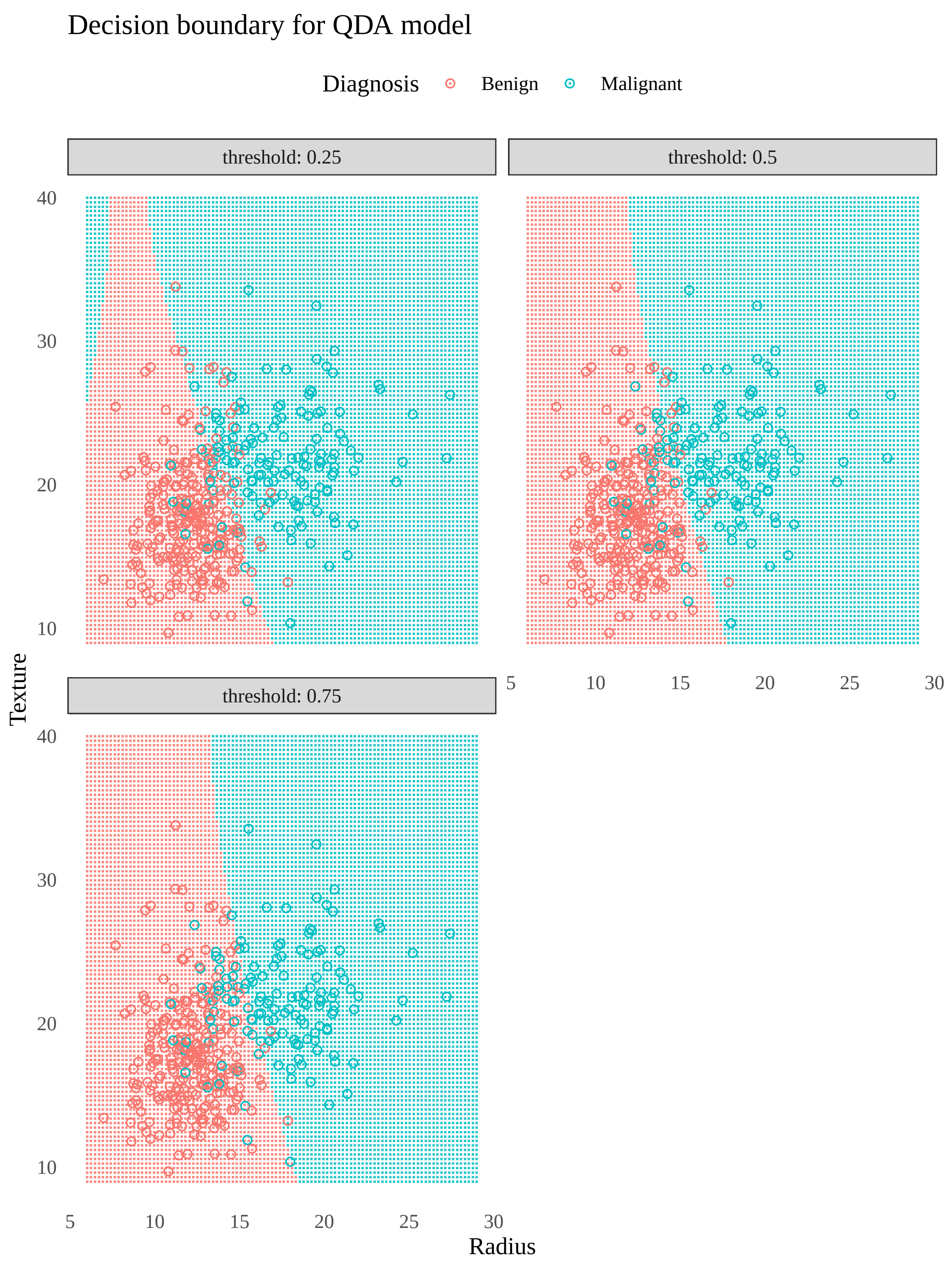

Figure 8: Boundary plot for QDA

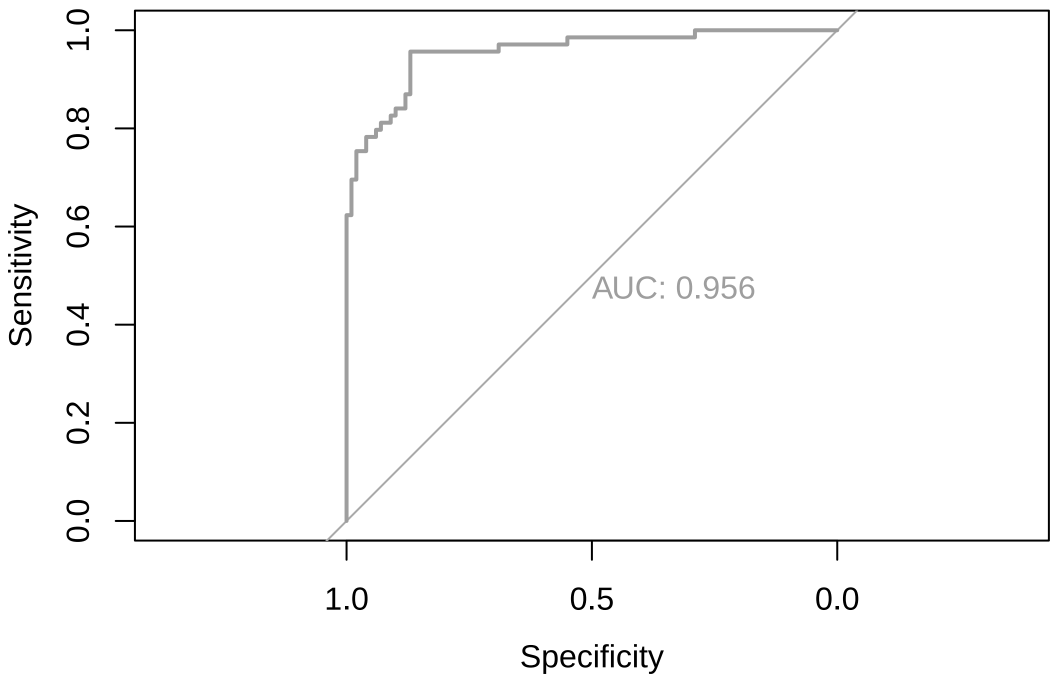

Figure 9: ROC for for QDA

Question 4a:

Table 19: Training kNN confusion matrix (k = 1)

Predicted -> Observed	Benign	Malignant
Benign	257	0
Malignant	0	143

Table 20: Training kNN confusion matrix (k = 2)

Predicted -> Observed	Benign	Malignant
Benign	243	14
Malignant	17	126

Table 21: Training kNN confusion matrix (k = 3)

Predicted -> Observed	Benign	Malignant
Benign	247	10
Malignant	23	120

Table 22: Training kNN confusion matrix (k = 4)

Predicted -> Observed	Benign	Malignant
Benign	247	10
Malignant	21	122

Table 23: Training kNN confusion matrix (k = 20)

Predicted -> Observed	Benign	Malignant
Benign	246	11
Malignant	22	121

Table 24: Test kNN confusion matrix (k = 1)

Predicted -> Observed	Benign	Malignant
Benign	94	6
Malignant	22	47

Table 25: Test kNN confusion matrix (k = 2)

Predicted -> Observed	Benign	Malignant
Benign	94	6
Malignant	20	49

Table 26: Test kNN confusion matrix (k = 3)

Predicted -> Observed	Benign	Malignant
Benign	100	0
Malignant	22	47

Table 27: Test kNN confusion matrix (k = 4)

Predicted -> Observed	Benign	Malignant
Benign	97	3
Malignant	21	48

Table 28: Test kNN confusion matrix (k = 20)

Predicted -> Observed	Benign	Malignant
Benign	98	2
Malignant	25	44

Looking at the above tables, $k = 1$ is perfectly classifying all observation (accuracy of train = 100%), but for larger k values we can observe an increased mis-classifications (the train accuracy slightly decreases with increasing k). However, as shown in the following table, the test accuracy increases with increasing k (not consistently). Across all k values the test accuracy increased except for $k = 20$.

Table 29: Predictive accuracy - kNN

k	training	test
1	1.000	0.834
2	0.922	0.846
3	0.917	0.870
4	0.922	0.858
20	0.917	0.840

Question 4b:

As k increases the boundary line becomes more and more smooth and represents more accurate boundaries between the outcome and the given predictors (figure 10).

Question 4c:

Figure 11 shows, the difference between test accuracy and train accuracy remained more or less consistent after k = 4. I plotted until k = 100 and the gap between test and train accuracy remained consistent without sign of decreasing. I found it to be challenging to decide which k value to use from this following predictive accuracy table.

Figure 12 shows test error rate with different k values. As k value increases the error approaches the test prevalence (0.41). We see that few values of k are tied for the lowest error rate, but we select the largest, as it is the least variable, and has the least chance of overfitting. Accordingly, k = 4 would be my choice.

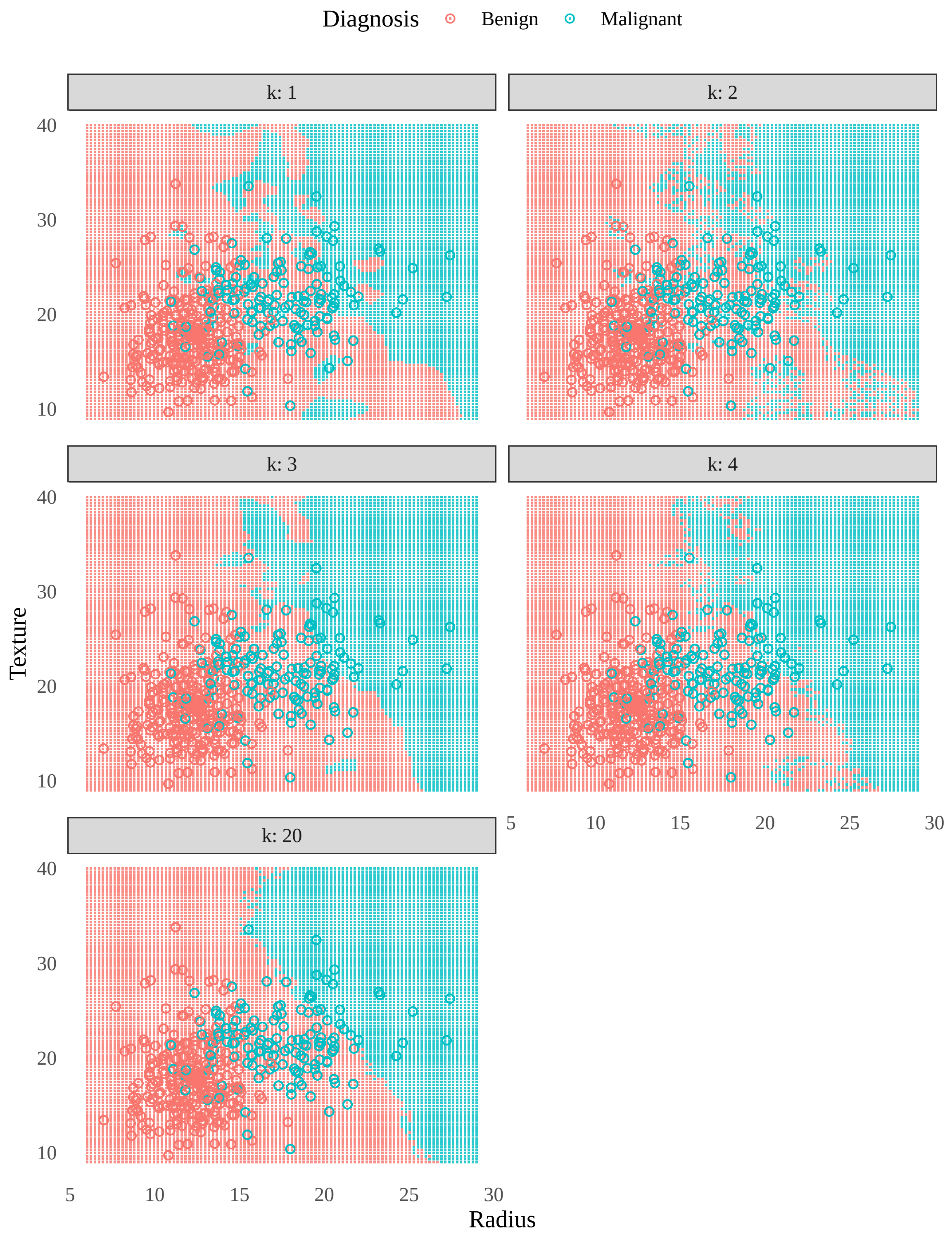

Figure 10: Boundary plot for KNN

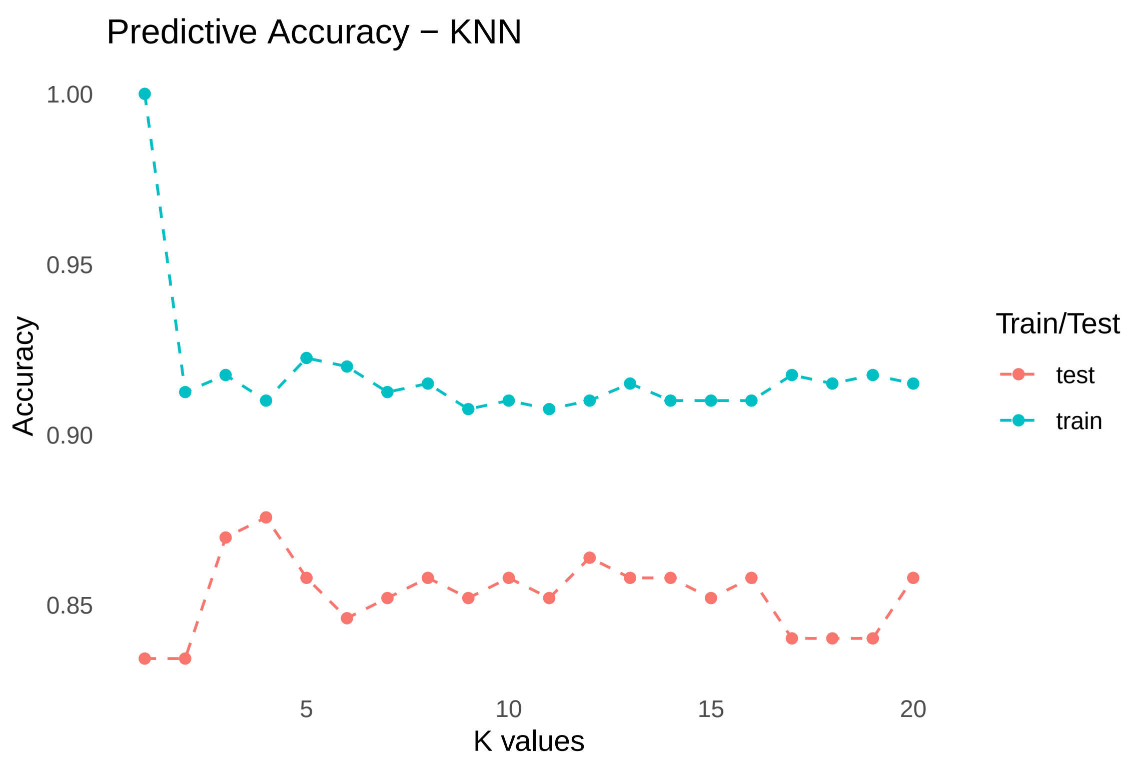

Figure 11: Train-test accuracy

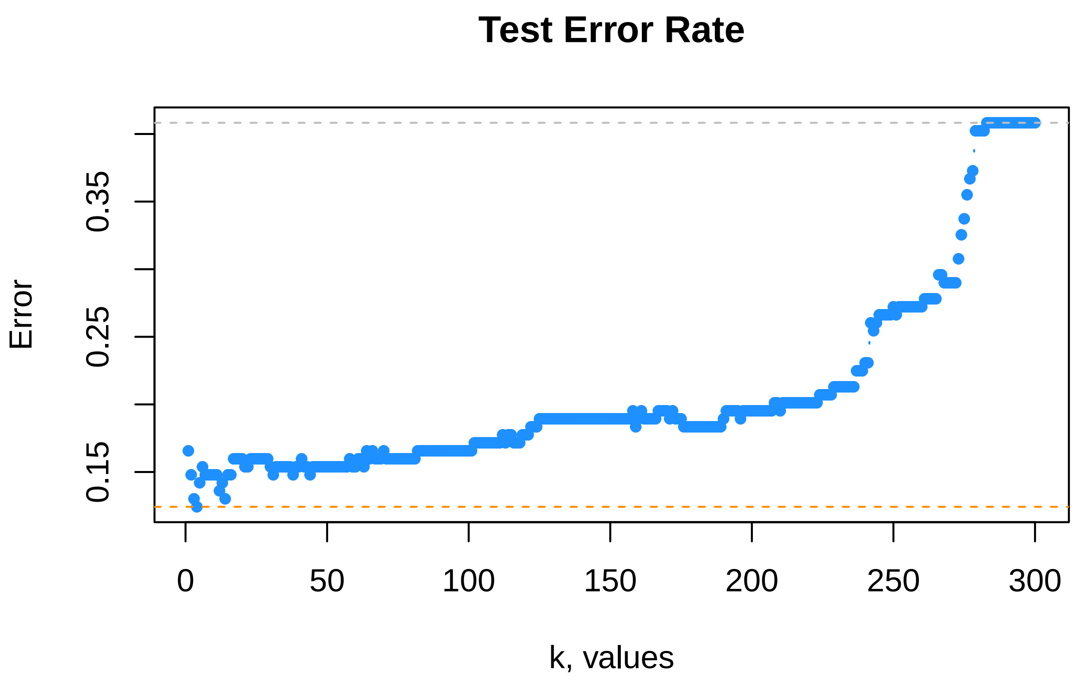

Figure 12: Test error for kNN

Code Appendix

```r
knitr::opts_chunk$set(fig.pos = 'H')
### Setting up the packages
library(knitr)
knitr::opts_chunk$set(echo = FALSE)
# check if packages are installed; if not, install them
packages <- c("tidyverse", "readr", "ggExtra", "plotly",
              "ggplot2","ggstatsplot","ggside","rigr","nlme","lmtest",
              "sandwich","gridExtra","broom","janitor","ellipse","caret",
              "pROC","MASS","class","purrr","tidyr")
not_installed <- setdiff(packages, rownames(installed.packages()))
if (length(not_installed)) install.packages(not_installed)

# load packages
library(sandwich)
library(janitor)
library(readr)
library(lmtest)
library(class)
library(pROC)
library(nlme)
library(ellipse)
library(broom)
library(ggstatsplot)
library(ggside)
library(caret)
library(rigr)
library(ggExtra)
library(gridExtra)
library(purrr)
library(plotly)
library(ggplot2)
library(MASS)
library(tidyverse)
library(tidyr)

### -----------------------------------------------------------
#Loading working directory of the raw data

#Please load your data/directory by changing it with your work directory
#Throughout this code module you will see a tone of places, where
#data is read and written, so please make sure to change them to your
#working directory folder format

working_directory_data <- setwd("C:/Users/latera/Desktop/ML_ass")

#loads the data on a variable df
study_data <- read.csv("data/wdbc.data", header=FALSE) %>%
  dplyr::select(diagnosis = V2, radius = V3, texture = V4) %>%
  mutate(diagnosis = factor(diagnosis, levels = c("B", "M"),
                            labels = c("Benign", "Malignant")))
```

```
#Describe the data
#as.tibble(study_data)


#Check if the outcome is factor
is.factor(study_data$diagnosis)

summary <- study_data %>%
  group_by(diagnosis) %>%
  summarise(observations = n())

knitr::kable(summary, caption = "Summary table of the data")
set.seed(0)
study_data_idx = sample(nrow(study_data), 400)
study_data_trn = study_data[study_data_idx, ]
study_data_tst = study_data[-study_data_idx, ]
featurePlot(x = study_data_trn[, c( "radius", "texture")],
y = study_data_trn$diagnosis,
plot = "pairs",
auto.key = list(columns = 2))

featurePlot(x = study_data_trn[, c("radius", "texture")],
y = study_data_trn$diagnosis,
plot = "ellipse",
auto.key = list(columns = 2))

featurePlot(x = study_data_trn[, c("radius", "texture")],
y = study_data_trn$diagnosis,
plot = "density",
scales = list(x = list(relation = "free"),
y = list(relation = "free")),
adjust = 1.5,pch = "|",
layout = c(2, 1),
auto.key = list(columns = 2))

model_glm = glm(diagnosis ~ radius + texture, data = study_data_trn,
                family = "binomial")

#I like tidy (broom)
knitr::kable(tidy(model_glm, conf.int = T),
      col.names = c('Variable',
                    "Coefficient",
                    "Standard Error",
                    "z-statistcis", "P-value",
                    "95% CI Lower",
                    "95% CI Higher"),
      caption = "GLM fit summary",
      digits = 6)
#summary(model_glm)
prob_numerator = exp(-19.383590  + (0.983655 * 10) + (0.241305   * 12))
prob_denomenator = 1 + prob_numerator
probablity_of_M_for_10_12 = (prob_numerator / prob_denomenator)
probablity_of_M_for_10_12
```

```
#predicting
pred_point <- function(.model, .type, ...) {
  predict(.model, type = .type, newdata = data.frame(...))
}
pred_point(model_glm, "response", radius = 10, texture = 12)
#print predicted probabilities
head(predict(model_glm, type = "response"))

calc_class_err = function(actual, predicted) {
mean(actual != predicted)
}

get_logistic_error = function(mod, data, res,
                              pos = 1, neg = 0, cut = 0.5) {
probs = predict(mod, newdata = data, type = "response")
preds = ifelse(probs > cut, pos, neg)
calc_class_err(actual = data[, res], predicted = preds)
}
train_error <- get_logistic_error(model_glm, data = study_data_trn,
res = "diagnosis", pos = "Malignant", neg = "Benign", cut = 0.5)

test_error <- get_logistic_error(model_glm, data = study_data_tst,
res = "diagnosis", pos = "Malignant", neg = "Benign", cut = 0.5)
model_glm_pred = ifelse(predict(model_glm, data=study_data_trn,
                                type = "response") > 0.5,
                        "Malignant", "Benign")

train_tab = table(predicted = model_glm_pred, actual = study_data_trn$diagnosis)
train_con_mat = confusionMatrix(train_tab, positive = "Malignant")



knitr::kable(train_tab,
      col.names = c("Actual Benign", "Actual Malignant"),
      digits = 5,caption = "Confussion
matrix for train data")

knitr::kable(c(train_con_mat$overall["Accuracy"],
train_con_mat$byClass["Sensitivity"],
train_con_mat$byClass["Specificity"],
train_con_mat$byClass["Pos Pred Value"],
train_con_mat$byClass["Neg Pred Value"],
train_con_mat$byClass["Prevalence"],
train_con_mat$byClass["Detection Rate"]),
      col.names = c("Percentages"),
      digits = 5,caption = "Confussion matrix for
train data")

model_glm_pred_tst = ifelse(predict(model_glm, newdata = study_data_tst,
                                type = "response") > 0.5,
                         "Malignant", "Benign")

tst_tab = table(predicted = model_glm_pred_tst,
```

```r
                 actual = study_data_tst$diagnosis)

tst_con_mat = confusionMatrix(tst_tab, positive = "Malignant")

knitr::kable(tst_tab,
      col.names = c("Actual Benign", "Actual Malignant"),
      digits = 5, caption = "Confussion matrix for test data")

knitr::kable(c(tst_con_mat$overall["Accuracy"],
tst_con_mat$byClass["Sensitivity"],
tst_con_mat$byClass["Specificity"],
tst_con_mat$byClass["Pos Pred Value"],
tst_con_mat$byClass["Neg Pred Value"],
tst_con_mat$byClass["Prevalence"],
tst_con_mat$byClass["Detection Rate"]),
      col.names = c("Percentages"),
      digits = 5,caption = "Confussion matrix for test data")

#since this problem has to run three times, I will create a function
plot_decision_boundary <- function(data, model_fit, thresholds) {

  grid_points <- expand.grid(
    radius = seq(floor(min(study_data$radius)),
                 ceiling(max(study_data$radius)),
                 length.out = 100),
    texture = seq(floor(min(study_data$texture)),
                  ceiling(max(study_data$texture)),
                  length.out = 100)
  )

  data_rep <- thresholds %>%
    purrr::map_dfr(~dplyr::mutate(data, threshold = .x))

  grid_classes <- model_fit(grid_points, thresholds)

  ggplot(grid_classes, aes(x = radius, y = texture, color = classes_predicted)) +
    geom_point(shape = ".", alpha = .84) +
    geom_point(data = data_rep, aes(color = diagnosis), shape = 1) +
    facet_wrap(vars(threshold), ncol = 2, labeller = "label_both") +
    theme_bw(base_family = "serif") +
    theme(legend.position = "top") +
    theme(axis.line = element_line(colour = "white"),
        axis.ticks = element_blank(),
    panel.grid.major = element_blank(),
    panel.grid.minor = element_blank(),
    panel.border = element_blank(),
    panel.background = element_blank())+
    labs(
      x = "Radius",
      y = "Texture",
      color = "Diagnosis"
    )
```

```r
}

create_bayes_classifier <- function(model_fit) {

  function(data, thresholds) {

    add_class <- function(threshold) {

      classes_predicted <- factor(
        model_fit(data) > threshold,
        levels = c(FALSE, TRUE),
        labels = c("Benign", "Malignant")
      )

      data %>% dplyr::mutate(
        classes_predicted = classes_predicted,
        threshold = threshold
      )

    }

    thresholds %>% purrr::map_dfr(add_class)
  }

}

log_bayes_pred <- create_bayes_classifier(
  function(data) predict(model_glm, data, type = "response")
)
grid_plot <-
  plot_decision_boundary(study_data_trn, log_bayes_pred, c(0.25, 0.5, 0.75)) +
  labs(
    title = "Decision boundary for glm (logistic model)",
  )
plot(grid_plot)
test_prob = predict(model_glm, newdata = study_data_tst,
                    type = "response")
test_roc = roc(study_data_tst$diagnosis ~ test_prob, plot = TRUE,
               print.auc = TRUE, col=8)
lda_model = lda(diagnosis~., data = study_data_trn, center = TRUE,
                scale = TRUE)


knitr::kable(lda_model$means,
      caption = "Group means",
      digits = 3)
knitr::kable(lda_model$prior,
      col.names = c("Prior probabilities"),
      caption = "Prior probabilities
of groups",
      digits = 3)
bayes_lda <- create_bayes_classifier(
  function(data) predict(lda_model, data)$posterior[, "Malignant"])
```

```r
train_pred = bayes_lda(study_data_trn, .5)
tst_pred = bayes_lda(study_data_tst, .5)

#Computing confusion table and prediction accuracy for train
train_tab = table(train_pred$classes_predicted, study_data_trn$diagnosis)
train_con_mat = confusionMatrix(train_tab, positive = "Malignant")

knitr::kable(train_tab,
      col.names = c("Actual Benign", "Actual Malignant"),
      digits = 5,caption = "Confussion matrix for train data")

knitr::kable(c(train_con_mat$overall["Accuracy"],
train_con_mat$byClass["Sensitivity"],
train_con_mat$byClass["Specificity"],
train_con_mat$byClass["Pos Pred Value"],
train_con_mat$byClass["Neg Pred Value"],
train_con_mat$byClass["Prevalence"],
train_con_mat$byClass["Detection Rate"]),
      col.names = c("Percentages"),
      digits = 5,caption = "Confussion matrix for
train data")
#Computing confusion table and prediction accuracy for train
tst_tab = table(tst_pred$classes_predicted, study_data_tst$diagnosis)
tst_con_mat = confusionMatrix(tst_tab, positive = "Malignant")


knitr::kable(tst_tab,
      col.names = c("Actual Benign", "Actual Malignant"),
      digits = 5, caption = "Confussion matrix for test data")

knitr::kable(c(tst_con_mat$overall["Accuracy"],
tst_con_mat$byClass["Sensitivity"],
tst_con_mat$byClass["Specificity"],
tst_con_mat$byClass["Pos Pred Value"],
tst_con_mat$byClass["Neg Pred Value"],
tst_con_mat$byClass["Prevalence"],
tst_con_mat$byClass["Detection Rate"]),
      col.names = c("Percentages"),
      digits = 5,caption = "Confussion matrix
for train data")

grid_plot <-
  plot_decision_boundary(study_data_trn, bayes_lda, c(0.25, 0.5, 0.75)) +
  labs(
    title = "Decision boundary for LDA model",
  )
plot(grid_plot)
test_prob = predict(lda_model, newdata=study_data_tst)
roc <- roc(response = study_data_tst$diagnosis, predictor =
             test_prob$posterior[, 1], levels =
             levels(study_data_tst$diagnosis),plot = TRUE,
           print.auc = TRUE, col=8)
```

```r
qda_model = qda(diagnosis~., data = study_data_trn, center = TRUE,
                scale = TRUE)


knitr::kable(qda_model$means,
      caption = "Group means",
      digits = 3)
knitr::kable(qda_model$prior,
      col.names = c("Prior probabilities"),
      caption = "Prior probabilities
of groups",
      digits = 3)
bayes_qda <- create_bayes_classifier(
  function(data) predict(qda_model, data)$posterior[, "Malignant"])
train_pred = bayes_qda(study_data_trn, .5)
tst_pred = bayes_qda(study_data_tst, .5)

#Computing confusion table and prediction accuracy for train
train_tab = table(train_pred$classes_predicted, study_data_trn$diagnosis)
train_con_mat = confusionMatrix(train_tab, positive = "Malignant")

knitr::kable(train_tab,
      col.names = c("Actual Benign", "Actual Malignant"),
      digits = 5,caption = "Confussion matrix
      for train data")

knitr::kable(c(train_con_mat$overall["Accuracy"],
train_con_mat$byClass["Sensitivity"],
train_con_mat$byClass["Specificity"],
train_con_mat$byClass["Pos Pred Value"],
train_con_mat$byClass["Neg Pred Value"],
train_con_mat$byClass["Prevalence"],
train_con_mat$byClass["Detection Rate"]),
      col.names = c("Percentages"),
      digits = 5,caption = "Confussion matrix
for train data")
#Computing confusion table and prediction accuracy for train
tst_tab = table(tst_pred$classes_predicted, study_data_tst$diagnosis)
tst_con_mat = confusionMatrix(tst_tab, positive = "Malignant")


knitr::kable(tst_tab,
      col.names = c("Actual Benign", "Actual Malignant"),
      digits = 5,caption = "Confussion matrix
      for train data")

knitr::kable(c(tst_con_mat$overall["Accuracy"],
tst_con_mat$byClass["Sensitivity"],
tst_con_mat$byClass["Specificity"],
tst_con_mat$byClass["Pos Pred Value"],
tst_con_mat$byClass["Neg Pred Value"],
tst_con_mat$byClass["Prevalence"],
tst_con_mat$byClass["Detection Rate"]),
      col.names = c("Percentages"),
      digits = 5,caption = "Confussion matrix for
train data")

grid_plot <-
  plot_decision_boundary(study_data_trn, bayes_qda, c(0.25, 0.5, 0.75)) +
  labs(
    title = "Decision boundary for QDA model",
  )
plot(grid_plot)
test_prob = predict(qda_model, newdata=study_data_tst)
roc <- roc(response = study_data_tst$diagnosis, predictor =
             test_prob$posterior[, 1], levels =
             levels(study_data_tst$diagnosis),plot = TRUE,
           print.auc = TRUE, col=8)

confusion_matrix <- function(data, classes_ob, classes_predicted) {

  data %>%
    dplyr::group_by({{classes_ob}}, {{classes_predicted}}, .add = TRUE) %>%
    dplyr::summarise(count = n()) %>%
    tidyr::pivot_wider(
      names_from = {{classes_predicted}},
      values_from = count,
      values_fill = 0
    ) %>%
    dplyr::ungroup({{classes_ob}}) %>%
    dplyr::rename(`Predicted -> Observed` = {{classes_ob}})

}

prediction_knn <- function(data, k_neighbors) {
  # Scale the features in the training data
  df_train_scale <- study_data_trn %>%
    mutate(across(radius:texture, ~scale(as.numeric(.x))))

  # Scale the features in the test data
  df_test_scale <- data %>%
    mutate(across(radius:texture, ~scale(as.numeric(.x))))

  # Function to add predictions based on a given number of nearest neighbors
  add_preds <- function(k) {
    data %>%
        mutate(
          classes_predicted = knn(
            train = dplyr::select(df_train_scale, radius:texture),
            test = dplyr::select(df_test_scale, radius:texture),
            cl = df_train_scale$diagnosis,
            k = k
          ),
          k = k
        )
  }
  # Apply the add_preds function to each number of neighbors in k_neighbors
  k_neighbors %>% map_dfr(add_preds)
}

classes <- list(
  train = prediction_knn(study_data_trn, c(1, 2, 3, 4, 20)) %>% group_by(k),
  test = prediction_knn(study_data_tst, c(1, 2, 3, 4, 20)) %>% group_by(k)
)
confusion_matrix_knn <- list(
  train = classes$train %>% confusion_matrix(diagnosis, classes_predicted),
  test = classes$test %>% confusion_matrix(diagnosis, classes_predicted)
)
accuracy_knn <- list(
  train = classes$train %>% summarise(accuracy = mean(diagnosis == classes_predicted)),
  test = classes$test %>% summarise(accuracy = mean(diagnosis ==
                                                     classes_predicted))
)

print_table <- function(k, data, type) {
  print(knitr::kable(
    x = data,
    caption = sprintf("%s kNN confusion matrix (k = %d)", type, k)
  ))
}
confusion_matrix_knn$train %>%
  tidyr::nest(data = -k) %>%
  purrr::pwalk(print_table, type = "Training")
confusion_matrix_knn$test %>%
  tidyr::nest(data = -k) %>%
  purrr::pwalk(print_table, type = "Test")

left_join_result <- left_join(accuracy_knn$train, accuracy_knn$test, by = "k")

knitr::kable(left_join_result,
             caption = "Predictive accuracy - kNN",
             col.names = c("k", "training", "test"),
             digits = 3)
plot_knn_grid <-
  plot_decision_boundary(study_data_trn, prediction_knn, c(1, 2, 3, 4, 20)) +
  facet_wrap(vars(k), ncol = 2, labeller = "label_both")
plot(plot_knn_grid)
pred_acc_knn <-
  list(train = study_data_trn, test = study_data_tst) %>%
  purrr::map(~prediction_knn(.x, 1:20) %>% group_by(k)) %>%
  purrr::map_dfr(
    ~summarise(.x, accuracy = mean(diagnosis == classes_predicted)),
    .id = "data_set"
  )
plot_pred_acc_knn <-
  ggplot(pred_acc_knn, aes(x = k, y = accuracy, color = data_set)) +
  geom_line(lty = "dashed") +
  geom_point() +
  theme_bw() +
  theme(axis.line = element_line(colour = "white"),
        axis.ticks = element_blank(),
    panel.grid.major = element_blank(),
    panel.grid.minor = element_blank(),
    panel.border = element_blank(),
    panel.background = element_blank())+
  theme(legend.background = element_rect
        (fill = "transparent"))+
  labs(
    title = "Predictive Accuracy - KNN",
    x = "K values",
    y = "Accuracy",
    color = "Train/Test"
  )

plot(plot_pred_acc_knn)
study_new_train = study_data_trn[, -1]
study_new_train_y = study_data_trn$diagnosis

study_new_tst = study_data_tst[, -1]
study_new_tst_y = study_data_tst$diagnosis

set.seed(42)
k_to_try = 1:300
err_k = rep(x = 0, times = length(k_to_try))
for (i in seq_along(k_to_try)) {
pred = knn(train = scale(study_new_train),
test = scale(study_new_tst),
cl = study_new_train_y,
k = k_to_try[i])
err_k[i] = calc_class_err(study_new_tst_y, pred)
}


# plot error vs choice of k
plot(err_k, type = "b", col = "dodgerblue", cex = 1, pch = 20,
xlab = "k, values", ylab = "Error",
main = "Test Error Rate")
# add line for min error seen
abline(h = min(err_k), col = "darkorange", lty = 2)
# add line for minority prevalence in test set
abline(h = mean(study_new_tst_y == "Malignant"), col = "grey", lty = 2)
```