Binary Classification on the Breast Cancer Diagnostic Data

Simple Logistic Regression, Linear/Quadratic discriminant Analysis Models & KNN

Makayla Tang

2023 Feb

Abstract

In this analysis, we perform binary classification on the Breast Cancer Wisconsin (Diagnostic) Data Set in the csv file wdbc.data. The dataset describes characteristics of the cell nuclei present in n (sample size) images. Each image has multiple attributes, which are described in detail in wdbc.names. We predict the attribute in column 2, which we denote by Y, representing the diagnosis (M = malignant, B = benign). Here, we focus on the attributes in columns {2:Diagnosis, 3:Average radius of the cell nuclei in each image, 4:Average texture of the cell nuclei in each image}. Specifically, our aim will be predicting a categorical variable Y (Diagnosis – column 2), from the quantitative attributes X1 (Average radius – column 3) and X2 (Average texture – column 4).

1.Data exploration + Logistic Regression

a.Describe the data: sample size n, number of predictors p, and number of observations in each class.

- n(sample size): 569
- number of predictors: 2
- number of observations in each class: M=212; B=357

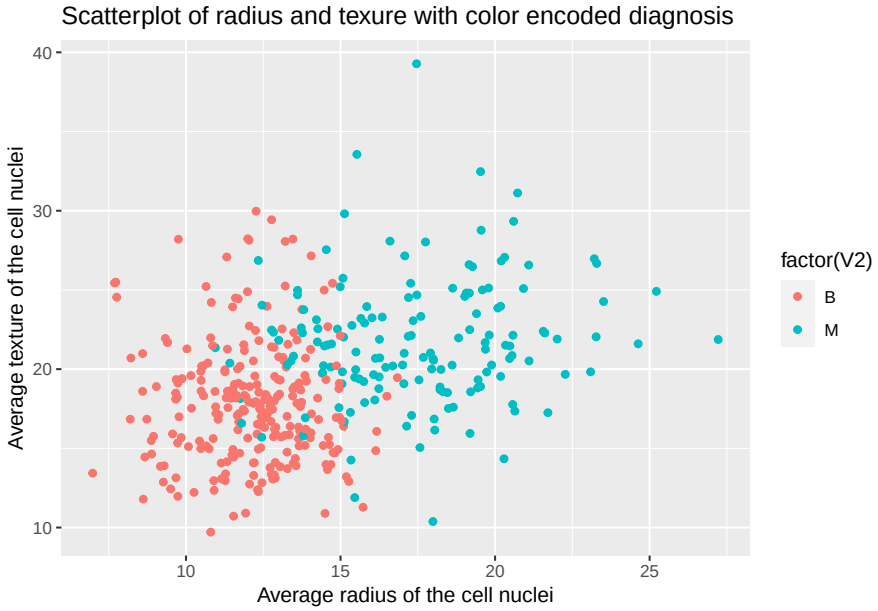
```
##-----------------------------------
# a.
wdbc %>% summarise(n=n())
wdbc %>% group_by(V2) %>% summarise(n=n(), )
glimpse(wdbc)
```

b.Divide the data into a training set of 400 observations, and a test set; from now on, unless specified, work only on the training set.

```
##------------------
## b.
## Divide the data into a training set of 400 obs
set.seed(101)
train_id_wdbc = sample(nrow(wdbc), 400)
train_wdbc = wdbc[train_id_wdbc, ]
test_wdbc  = wdbc[-train_id_wdbc, ]
```

c.Make a scatterplot displaying Y (color or shape encoded) and the predic- tors X1, X2 (on the x- and y-axis). Based on this scatterplot, do you think it will be possible to accurately predict the outcome from the predictors? Motivate your answer.

Based on the plot, we conclude that the average radius is a better predictor for the diagnosis as the blue points (diagnosis with malignant) tend to have higher values and the red points (diagnosis with benign) tend to have lower values, while the average texture does not seem to be a strong predictor as the red points and the blue points are spread evenly.



d.Fit a logistic regression model to predict Y and make a table, displaying the coefficient estimates, standard errors, and p-values (use command summary). Give an interpretation of the values of the coefficient estimates.

- The estimated expected log-odds of diagnosis with malignant for a population with average radium = 0 and average texture = 0 is -21.381
- The log-odds of diagnosis with malignant is estimated to be 1.123 times greater for a population with one more unit increase in the average radius of the cell nuclei, compared to a population of the same average texture of the cell nuclei.
- The log-odds of the diagnosis with malignant is estimated to be 0.247 times greater for a population with one more unit increase in the average texture of the cell nuclei, compared to a population of the same average radius of the cell nuclei.

```
##------------------
## logistic model
## transfer M variable from character to numeric (0,1)
train_wdbc$Dig <- ifelse (train_wdbc$V2=="M", 1, 0)

glm.wdbc = glm(Dig ~ V3 + V4, family = binomial(link = "logit"), data = train_wdbc)
summary(glm.wdbc)
```

```
kable(summary(glm.wdbc)$coefficients)
```

	Estimate	Std. Error	z value	Pr(>|z|)
(Intercept)	-21.3813316	2.3238811	-9.200699	0e+00
V3	1.1232747	0.1304202	8.612734	0e+00
V4	0.2474737	0.0485324	5.099144	3e-07

e.Use the coefficient estimates to manually calculate the predicted probabil- ity P(Y = M | (X1,X2) = (10,12)) writing explicitly every step. Compare your results with the prediction computed with predict.

- manually calculate: $prediction = -21.381 + 1.123 * 10 + 0.247 * 12 = -7.178901$, $probability = 1/(1 + exp(-7.178901) = 0.00076$
- predict(glm.wdbc, data.frame(V3 = 10, V4 = 12), type = "response") = 0.00076
- Therefore, the two results are the same.

```
##------------------
## e.
## extract the coefficients
beta_0 <- coef(glm.wdbc)[1]
beta_1 <- coef(glm.wdbc)[2]
beta_2 <- coef(glm.wdbc)[3]

## calculate the linear predictor
pred <- beta_0 + beta_1*10 + beta_2*12

## calculate the predicted probability
prob <- 1 / (1 + exp(-pred))
prob
# 0.0007619246

# compare with the prediction computed using predict
predict(glm.wdbc, data.frame(V3 = 10, V4 = 12), type = "response")
# 0.0007619246

# The two values are the same.
```

f.Use the glm previously fitted and the Bayes rule to compute the pre- dicted outcome Y^ from the associated probability estimates (computed with predict) both on the training and the test set. Then compute the confusion table and prediction accuracy (rate of correctly classified obser- vations) both on the training and test set. Comment on the results.

- training set: accuracy = 0.905
- testing set: accuracy = 0.882

The model accuracy for training data and the one for testing data are similar. It indicates that the good model performance.

```
##------------------
## f.
## Compute the predicted probabilities using the predict function
```

```r
train_probs <- predict(glm.wdbc, newdata = train_wdbc, type = "response")
test_probs <- predict(glm.wdbc, newdata = test_wdbc, type = "response")

## Use the Bayes rule to compute the predicted outcome
train_prediction <- ifelse(train_probs > 0.5, "M", "B")
test_prediction <- ifelse(test_probs > 0.5, "M", "B")

## Create confusion tables on the training and test sets
train_confusion_table <- table(Prediction = train_prediction, Actual = train_wdbc$V2)
test_confusion_table <- table(Prediction = test_prediction, Actual = test_wdbc$V2)

## Calculate prediction accuracy on the training and test sets
train_accuracy <- sum(diag(train_confusion_table)) / nrow(train_wdbc)
test_accuracy <- sum(diag(test_confusion_table)) / nrow(test_wdbc)
```

```r
kable(train_confusion_table,
      caption = "Confusion Table for training set")
```

Table 2: Confusion Table for training set

	B	M
B	233	23
M	15	129

```r
kable(test_confusion_table,
      caption = "Confusion Table for testing set")
```

Table 3: Confusion Table for testing set

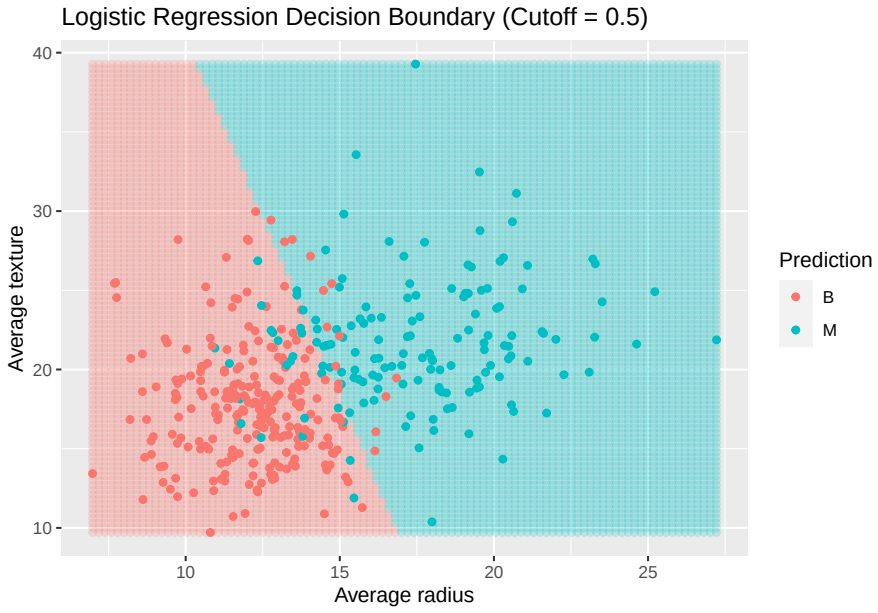
	B	M
B	101	12
M	8	48

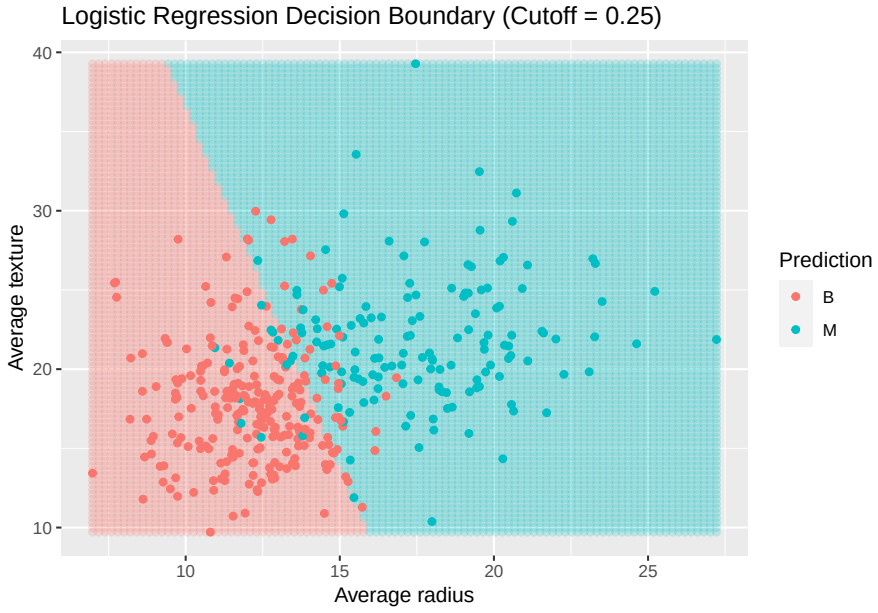
g.Plot an image of the decision boundary as follows:1.Generate a dense set (e.g. 10000 observations) of possible values for the predictors (X1,X2) within reasonable ranges; (the command expand.grid might come in handy); 2.Use the glm model previously fitted to predict the outcome probabil- ities for every observation you have generated and use Bayes rule to compute the predicted outcomes; 3.Plot predicted outcomes (color coded) and associated predictors in a 2D scatter plot together with the training set.Generate the same plot for probability cutoff values of 0.25 and 0.75. Comment on the results.

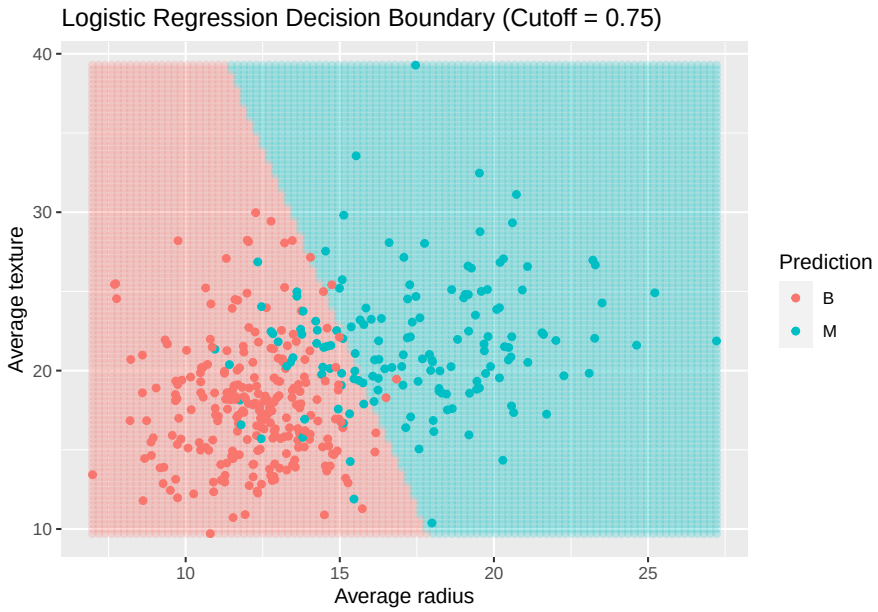
The increase in the red region (predicted as benign tumor) as the cutoff value increases. When the cutoff value is low, the model would be more conservative in classifying an observation as benign, while as the cutoff value increases, the model becomes more confident in classifying an observation as benign. Therefore, the red region would increase with higher cutoff value.

```r
## g.
## specify reasonable ranges for X1 and X2
x1_range <- c(min(train_wdbc$V3), max(train_wdbc$V3))
```

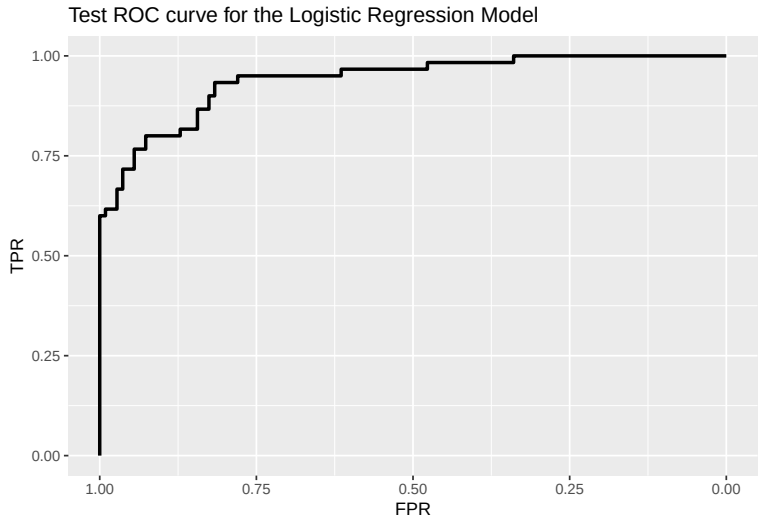
```r
x2_range <- c(min(train_wdbc$V4), max(train_wdbc$V4))

## create a dense set of possible values for X1 and X2 using expand.grid
dense_set <- expand.grid(V3 = seq(from = x1_range[1], to = x1_range[2], length.out = 100),
                         V4 = seq(from = x2_range[1], to = x2_range[2], length.out = 100))

## use the fitted GLM model to predict the outcome probabilities for each observation in the dense set
dense_set$probs <- predict(glm.wdbc, newdata = dense_set, type = "response")

## compute the predicted outcomes using Bayes rule
dense_set$prediction <- ifelse(dense_set$probs > 0.5, "M", "B")
```







h.Plot the ROC curve, computed on the test set.



i.Compute an estimate of the Area under the ROC Curve (AUC).

The AUC score is 0.9398.

2.Linear Discriminant Analysis Model

a.Now fit a linear discriminant analysis model to the training set you created in Exercise 1. Make a table displaying the estimated 'Prior probabilities of groups' and 'Group means'. Describe in words the meaning of these estimates and how they are related to the posterior probabilities.

The estimated group means are the average of the predictor variables (radius or texture) for each group (malignant or benign). These estimates are used to calculate the posterior probabilities which is the probability of an observation belonging to malignant or benign given its estimated predictor values. This calculation

is based on Bayes' theorem, which combines prior knowledge (prior probabilities) and new information (estimated predictor values) to estimate the probability of an event.

```
##------------------------------------ LDA
## a.
## linear discriminant model
lda.wdbc = lda(Dig ~ V3 + V4, data = train_wdbc, center = TRUE)

lda_table <- cbind(lda.wdbc$prior, lda.wdbc$mean)
colnames(lda_table) <- c("Prior prob of groups", "Group means(radius)", "Group means(texture)")
```

```
kable(lda_table)
```

	Prior prob of groups	Group means(radius)	Group means(texture)
0	0.62	12.15901	17.87661
1	0.38	17.46270	21.68230

b.Use the fitted model and Bayes rule to compute the predicted outcome Y^ from the predicted posterior probabilities, both on the training and test set. Then, compute the confusion table and prediction accuracy both on the training and test set. Comment on the results.

- Prediction accuracy on training set = 0.89
- Prediction accuracy on testing set = 0.876

The two prediction accuracy are similar, it indicates that the model has a good performance. The values are slightly smaller than the accuracy from fitting the logistic model. LDA finds the optimal linear combinations of predictors that maximize the separation between classes and then uses Bayes' theorem to make predictions, while logistic regression uses maximum likelihood estimation to fit a logistic function to the data. In this scenario, it seems that the logistic model is more appropriate because LDA with linear combination might be relative simpler for the data.

```
##------------------
## b
X_train <- train_wdbc %>% select(c(V3, V4))
X_test <- test_wdbc %>% select(c(V3, V4))

## fit the model and calculate post prob
train_pred.lda <- predict(lda.wdbc, X_train)$posterior
test_pred.lda <- predict(lda.wdbc, X_test)$posterior

## compute outcome Y based on the post prob
train_outcome.lda <- apply(train_pred.lda, 1, which.max)
## outcome variable = 1 or 2
## transfer them to B or M
train_outcome.lda <- data.frame(train_outcome.lda)
train_outcome.lda$train_outcome.lda <-
        ifelse(train_outcome.lda$train_outcome.lda==1, "B", "M")

## Do it again, but using testing set
test_outcome.lda <- apply(test_pred.lda, 1, which.max)
```

```r
## outcome variable = 1 or 2
## transfer them to B or M
test_outcome.lda <- data.frame(test_outcome.lda)
test_outcome.lda$test_outcome.lda <-
        ifelse(test_outcome.lda$test_outcome.lda==1, "B", "M")

## confusion matrix
train_confusion.lda <- table(Prediction = train_outcome.lda$train_outcome.lda,
                             Actual = train_wdbc$V2)
test_confusion.lda <- table(Prediction = test_outcome.lda$test_outcome.lda,
                            Actual = test_wdbc$V2)

## prediction accuracy
train_accuracy.lda <- sum(diag(train_confusion.lda)) / nrow(train_wdbc)
test_accuracy.lda <- sum(diag(test_confusion.lda)) / nrow(test_wdbc)
```

```r
kable(train_confusion.lda,
      caption = "Confusion table for training set - LDA")
```

Table 5: Confusion table for training set - LDA

	B	M
B	241	37
M	7	115

```r
kable(test_confusion.lda,
      caption = "Confusion table for testing set - LDA")
```

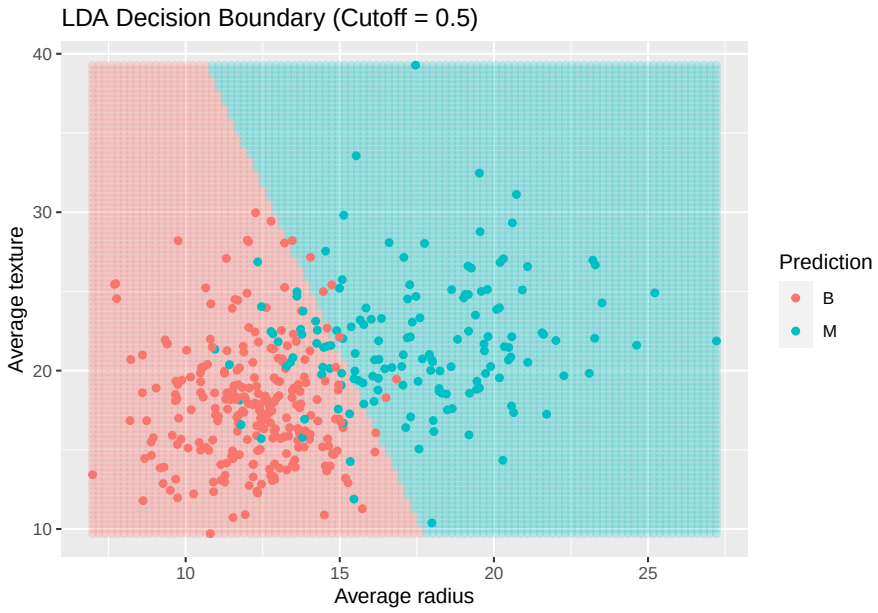
Table 6: Confusion table for testing set - LDA

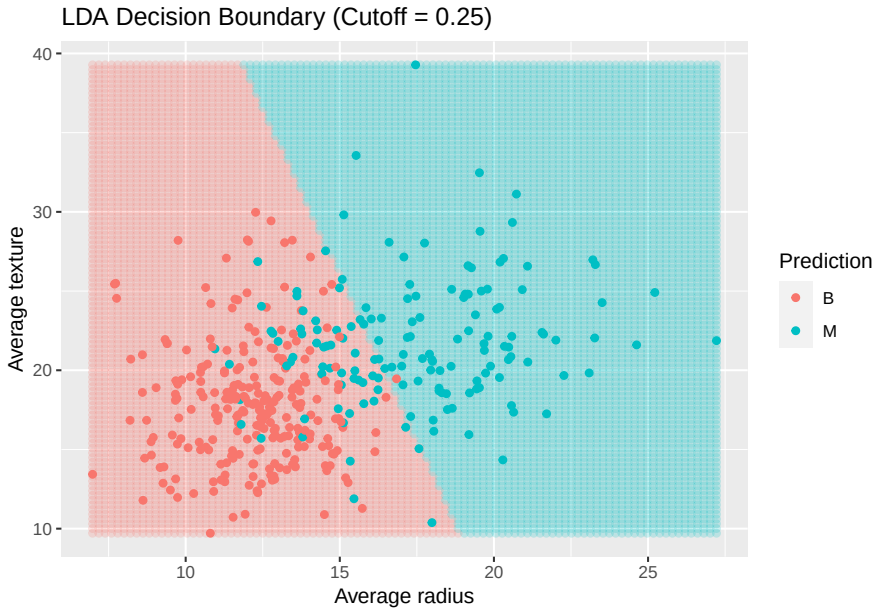
	B	M
B	104	16
M	5	44

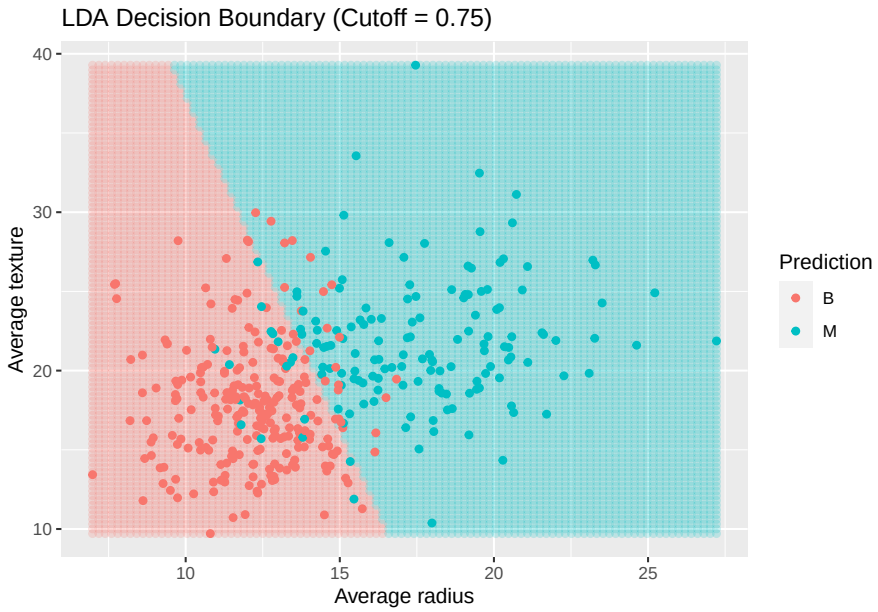
c.Plot an image of the LDA decision boundary (following the steps in 1(g)). Generate the same plot for cutoff values of 0.25 and 0.75. Comment on the results.

The increase in the red region (predicted as benign tumor) as the cutoff value decreases When the cutoff value is high, the model would be more conservative in classifying an observation as benign, while as the cutoff value decrease, the model becomes more confident in classifying an observation as benign.

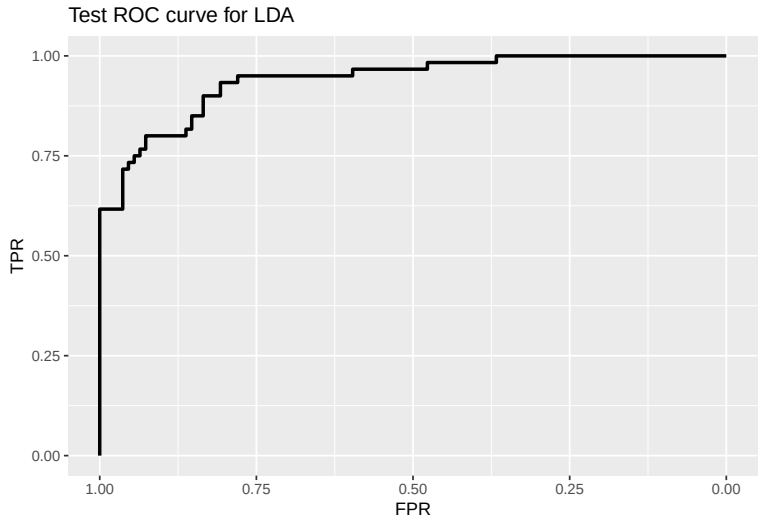
```r
##------------------
## c.
## use the fitted model to predict the outcome probabilities for each observation in the dense set
probs.lda <- predict(lda.wdbc, newdata = dense_set, type = "response")$posterior

# Compute the predicted outcomes using Bayes rule with different probability cutoff values
pred_outcomes_05 = ifelse(probs.lda[,1] > probs.lda[,2], "B", "M")
pred_outcomes_25 = ifelse(probs.lda[,1] > 0.25, "B", "M")
pred_outcomes_75 = ifelse(probs.lda[,1] > 0.75, "B", "M")
```


LDA Decision Boundary (Cutoff = 0.5)
Average texture
Average radius
40
30
20
10
10
15
20
25
Prediction
B
M


LDA Decision Boundary (Cutoff = 0.25)
Average texture
Average radius
40
30
20
10
10
15
20
25
Prediction
B
M



d.Plot the ROC curve, computed on the test set.



e.Compute an estimate of the Area under the ROC Curve (AUC).

The AUC score is 0.9396

3.Quadratic Discriminant Analysis Model.

a.Now fit a linear discriminant analysis model to the training set you created in Exercise 1. Make a table displaying the estimated 'Prior probabilities of groups' and 'Group means'. Describe in words the meaning of these estimates and how they are related to the posterior probabilities.

The estimated group means are the average of the predictor variables (radius or texture) for each group (malignant or benign). These estimates are used to calculate the posterior probabilities which is the probability of an observation belonging to malignant or benign given its estimated predictor values. This calculation

is based on Bayes' theorem, which combines prior knowledge (prior probabilities) and new information (estimated predictor values) to estimate the probability of an event.

```
## a.
## quadratic discriminant analysis
qda.wdbc = qda(Dig ~ V3 + V4, data = train_wdbc, center = TRUE)
qda.wdbc
qda_table <- cbind(qda.wdbc$prior, qda.wdbc$mean)
colnames(qda_table) <- c("Prior prob of groups", "Group means(radius)", "Group means(texture)")
```

```
kable(qda_table)
```

	Prior prob of groups	Group means(radius)	Group means(texture)
0	0.62	12.15901	17.87661
1	0.38	17.46270	21.68230

b.Use the fitted model and Bayes rule to compute the predicted outcome Y^ from the predicted posterior probabilities, both on the training and test set. Then, compute the confusion table and prediction accuracy both on the training and test set. Comment on the results.

- Prediction accuracy on training set = 0.885
- Prediction accuracy on testing set = 0.870

The two prediction accuracy are similar, which indicates that the model fits well. The values are similar to the ones from LDA. It seems that using LDA or QDA is not so much different.

```
##------------------
## b.
## fit the model and calculate post prob
train_pred.qda <- predict(qda.wdbc, X_train)$posterior
test_pred.qda <- predict(qda.wdbc, X_test)$posterior

## compute outcome Y based on the post prob on training set
train_outcome.qda <- apply(train_pred.qda, 1, which.max)
## outcome variable = 1 or 2
## transfer them to B or M
train_outcome.qda <- data.frame(train_outcome.qda)
train_outcome.qda$train_outcome.qda <-
        ifelse(train_outcome.qda$train_outcome.qda==1, "B", "M")

## compute outcome Y based on the post prob on testing set
test_outcome.qda <- apply(test_pred.qda, 1, which.max)
## outcome variable = 1 or 2
## transfer them to B or M
test_outcome.qda <- data.frame(test_outcome.qda)
test_outcome.qda$test_outcome.qda <-
        ifelse(test_outcome.qda$test_outcome.qda==1, "B", "M")

## confusion matrix
train_confusion.qda <- table(Prediction = train_outcome.qda$train_outcome.qda,
```

```
                              Actual = train_wdbc$V2)
test_confusion.qda <- table(Prediction = test_outcome.qda$test_outcome.qda,
                            Actual = test_wdbc$V2)

## prediction accuracy
train_accuracy.qda <- sum(diag(train_confusion.qda)) / nrow(train_wdbc)
test_accuracy.qda <- sum(diag(test_confusion.qda)) / nrow(test_wdbc)
```

```
kable(train_confusion.qda,
      caption = "Confusion table for training set - QDA")
```

Table 8: Confusion table for training set - QDA

	B	M
B	236	34
M	12	118

```
kable(test_confusion.qda,
      caption = "Confusion table for testing set - QDA")
```

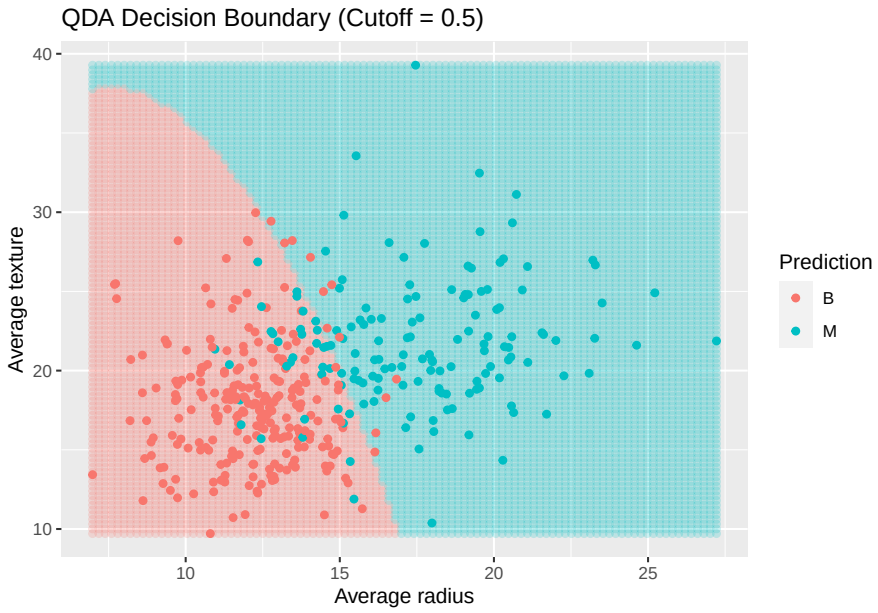
Table 9: Confusion table for testing set - QDA

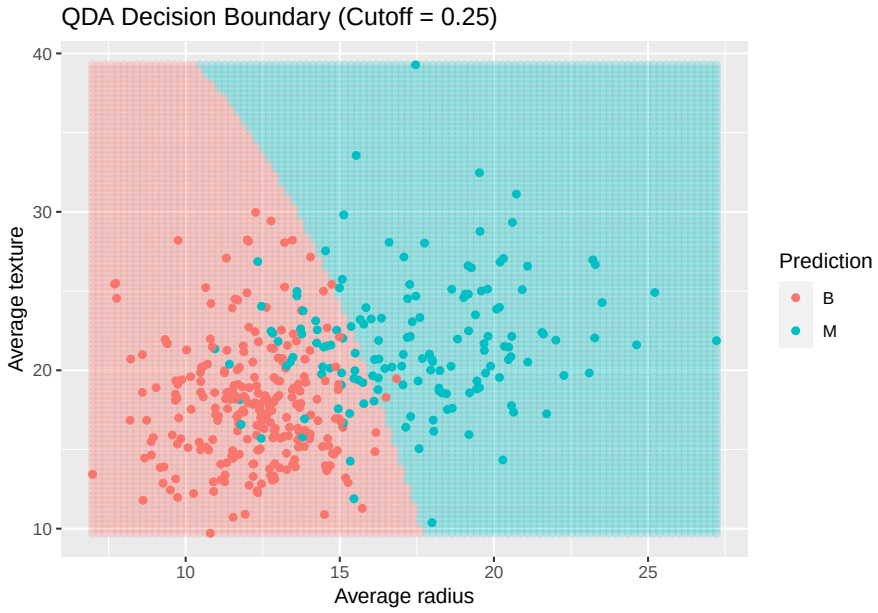
	B	M
B	101	14
M	8	46

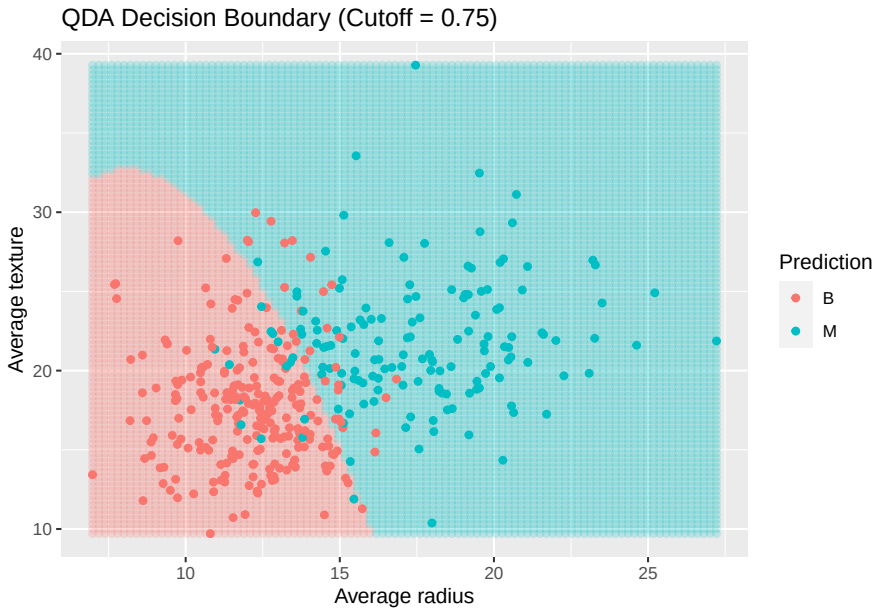
c.(c) Plot an image of the QDA decision boundary (following the steps in 1(g)). Generate the same plot for cutoff values of 0.25 and 0.75. Comment on the results.

The increase in the red region (predicted as benign tumor) as the cutoff value decreases When the cutoff value is high, the model would be more conservative in classifying an observation as benign, while as the cutoff value decrease, the model becomes more confident in classifying an observation as benign.

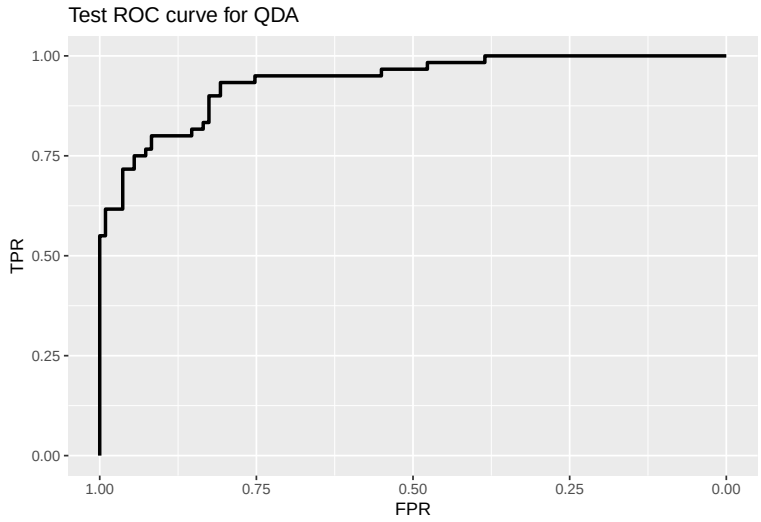
```
##------------------
## c.
## use the fitted GLM model to predict the outcome probabilities for each observation in the dense set
probs.qda <- predict(qda.wdbc, newdata = dense_set, type = "response")$posterior

# Compute the predicted outcomes using Bayes rule with different probability cutoff values
pred_outcomes_05.qda = ifelse(probs.qda[,1] > probs.qda[,2], "B", "M")
pred_outcomes_25.qda = ifelse(probs.qda[,1] > 0.25, "B", "M")
pred_outcomes_75.qda = ifelse(probs.qda[,1] > 0.75, "B", "M")
```


QDA Decision Boundary (Cutoff = 0.5)
Average texture
Average radius
Prediction
B
M
40
30
20
10
10
15
20
25


QDA Decision Boundary (Cutoff = 0.25)
Average texture
Average radius
Prediction
B
M
40
30
20
10
10
15
20
25



d.Plot the ROC curve, computed on the test set.



e.Compute an estimate of the Area under the ROC Curve (AUC).

The AUC score is 0.9361

4.KNN Classifier.

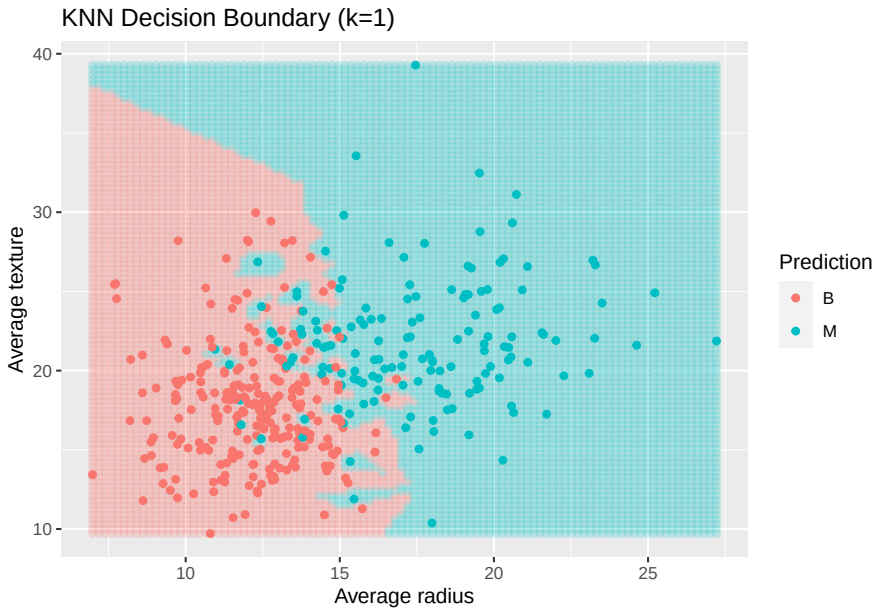
a.For all choices of k = {1, 2, 3, 4, 20} (number of neighbors), compute the predicted outcome Y^ for each observation in the training and test set. Then, compute the confusion table and prediction accuracy both on the training and test set. Comment on the results.

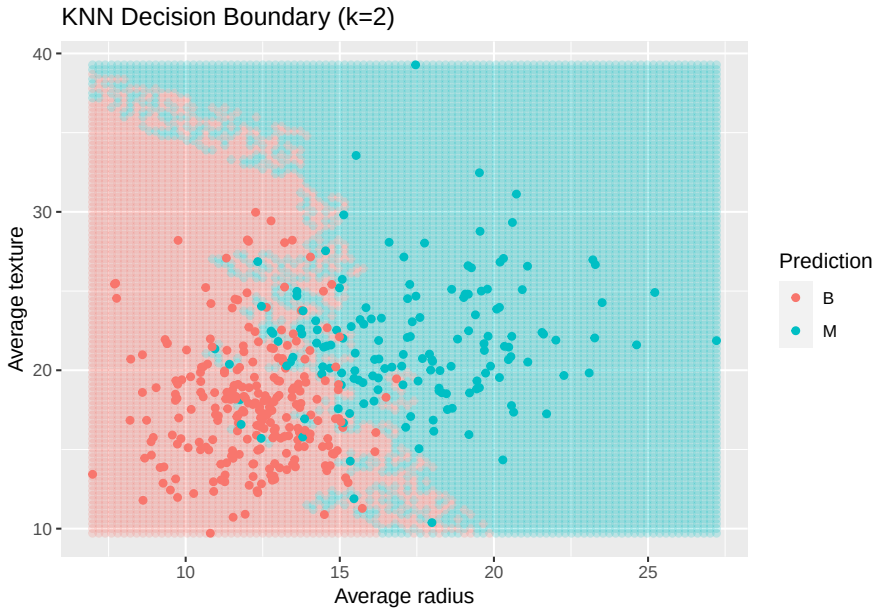
```
##----------------------------------- KNN
## a.
## knn
```

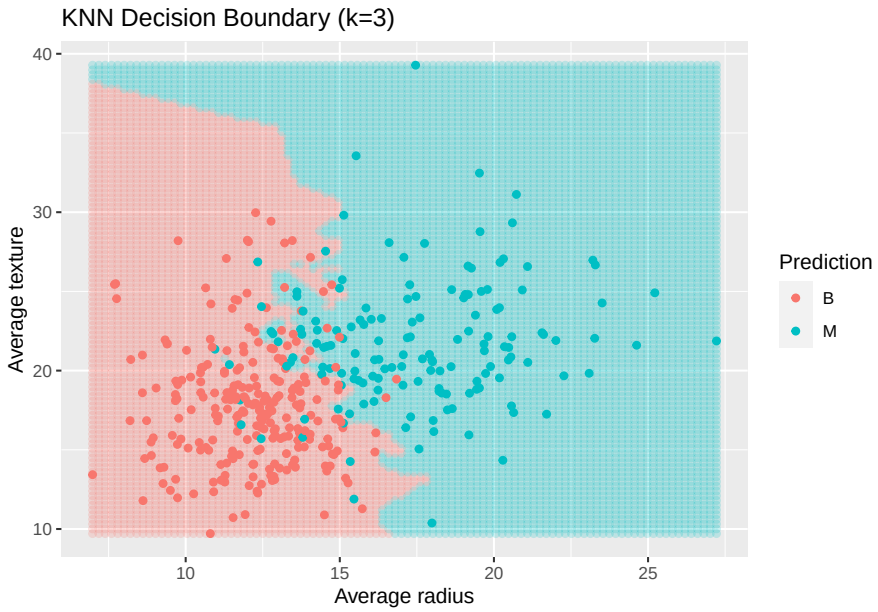
```r
k_values <- c(1, 2, 3, 4, 20)
for (k in k_values) {
        # train set
        knn.train.label <- knn(train = train_wdbc[,c("V3", "V4")],
                               test = train_wdbc[,c("V3", "V4")],
                               cl = train_wdbc$V2, k = k)

        # confusion table
        confusion_table_train <- table(knn.train.label, train_wdbc$V2)
        # prediction accuracy
        accuracy_train <- sum(diag(confusion_table_train)) / sum(confusion_table_train)

        # test set
        knn.test.label <- knn(train = train_wdbc[,c("V3", "V4")],
                              test = test_wdbc[,c("V3", "V4")],
                              cl = train_wdbc$V2, k = k)

        # confusion table - test set
        confusion_table_test <- table(knn.test.label, test_wdbc$V2)
        # prediction accuracy
        accuracy_test <- sum(diag(confusion_table_test)) / sum(confusion_table_test)

        # Print the results
        cat("For k =", k, ":\n")
        cat("Training set confusion table:\n")
        print(confusion_table_train)
        cat("Training set accuracy:", accuracy_train, "\n")
        cat("Test set confusion table:\n")
        print(confusion_table_test)
        cat("Test set accuracy:", accuracy_test, "\n\n")
}
```

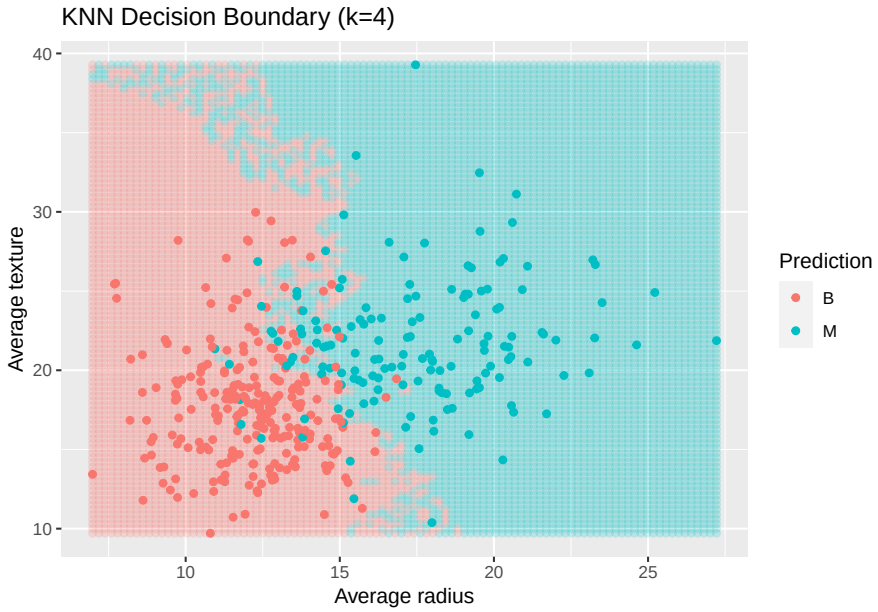
b.Plot an image of the decision boundary (following the steps in 1(g)), for k = {1, 2, 3, 4, 20} (number of neighbors). Comment on the results.

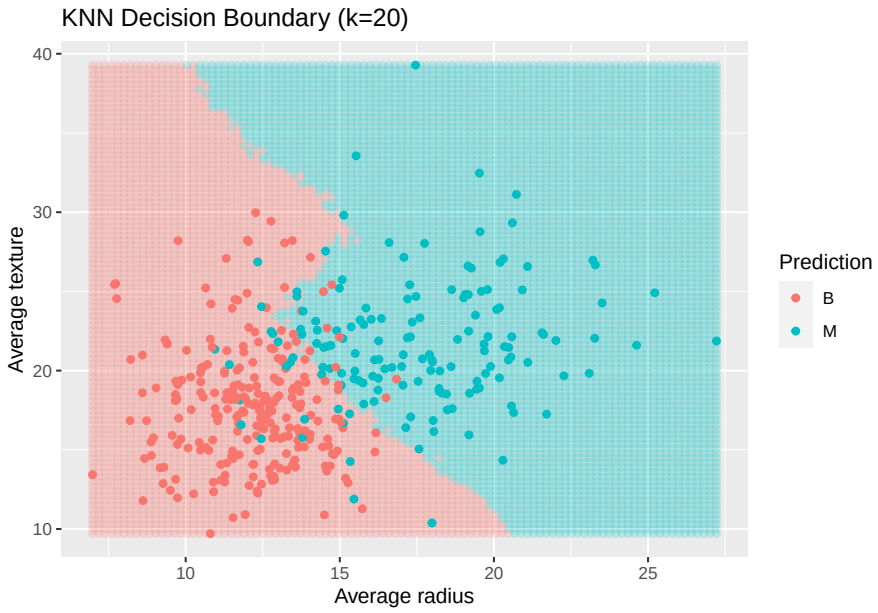
Based on the KNN decision boundary plots with different values of k, we can see that the boundaries are less specific than the boundaries using logistic regression, LDA, and QDA. It makes sense because KNN models make predictions based on the classes of the nearest neighbors, whereas logistic regression, LDA, and QDA are based on a more sophisticated mathematical model. In addition, according to the five plots with different values of k, we know that the decision boundary is more precise when k=3. It seems that k=3 might be more appropriate in this scenario.


KNN Decision Boundary (k=1)
Average texture
Average radius
40
30
20
10
10
15
20
25
Prediction
B
M


KNN Decision Boundary (k=2)
Average texture
Average radius
40
30
20
10
10
15
20
25
Prediction
B
M


KNN Decision Boundary (k=3)
Average texture
Average radius
40
30
20
10
10
15
20
25
Prediction
B
M

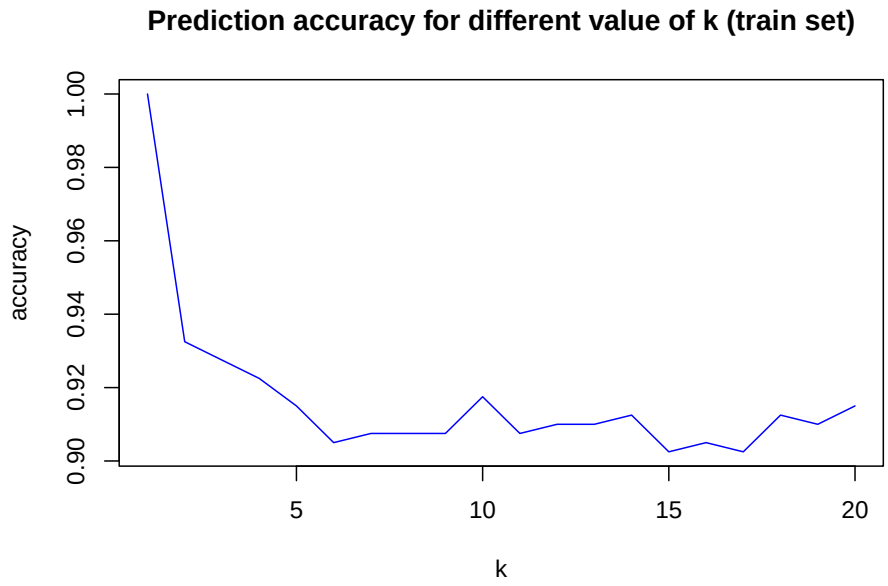

KNN Decision Boundary (k=4)
Average texture
Average radius
40
30
20
10
10
15
20
25
Prediction
B
M

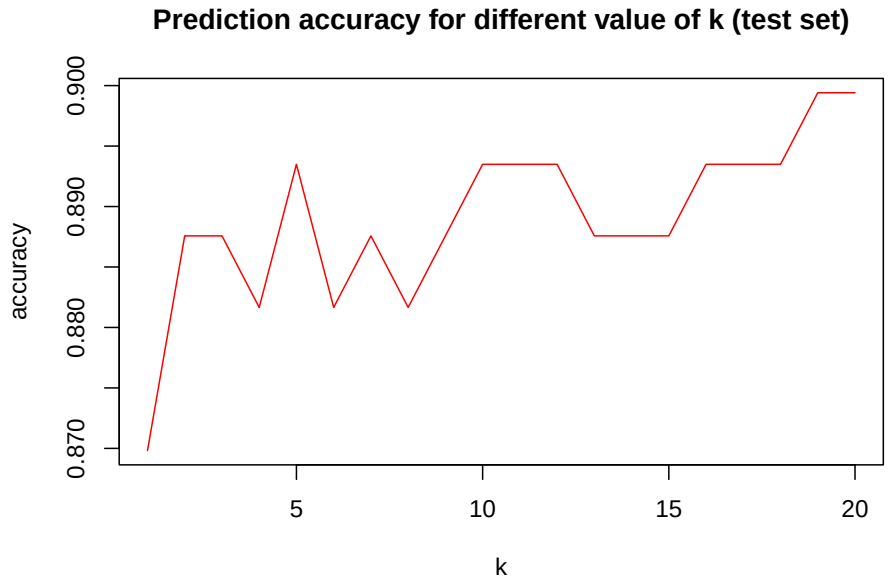


c.Compute and plot the prediction accuracy (on both the training and test set), for k = {1,2,. . . ,19,20} (number of neighbors). Which value of k would you choose? Comment on the results.

I would choose k=3. The followings are my reasons. Based on the plot of prediction accuracy for different value of k on the training set, we can see that the accuracy decrease as the k value increases. With the elbow method, the elbow point is at k=2 or k=3. In addition, based on the prediction accuracy for different values of k on the testing, we can see that the accuracy at k=3, k=6, k=17, k=18, k=19, and k=20 are relatively high. Therefore, I would choose k=3 as the best k value.

```
##------------------
## c.
## plot the prediction accuracy
train_accuracy = numeric(20)
test_accuracy = numeric(20)

for (i in 1:20) {
        knn.label <- knn(train = train_wdbc[,c("V3","V4")],
                         test = train_wdbc[,c("V3","V4")],
                         cl = train_wdbc$V2, k = i)
        train_accuracy[i] <- mean(knn.label == train_wdbc$V2)
        test_accuracy[i] <- mean(knn.label == test_wdbc$V2)
}

for (i in 1:20) {
        knn.label <- knn(train = train_wdbc[,c("V3", "V4")],
                          test = test_wdbc[,c("V3", "V4")],
                          cl = train_wdbc$V2, k = i)
        test_accuracy[i] <- mean(knn.label == test_wdbc$V2)
}
```





Code Appendix

```
knitr::opts_chunk$set(echo = TRUE)
knitr::opts_chunk$set(warning = FALSE, message = FALSE)
## load the libraries
library(knitr)
library(ggplot2)
library(MASS)  # Implements LDA & QDA
library(caret) # confusionMatrix
library(pROC)  # ROC & AUC
library(class) # Implements KNN
library(dplyr)
library(GGally)
```

```r
##-----------------------------------
## load the dataset
wdbc <- read.csv("wdbc.data", header = FALSE)
head(wdbc)
## select column {2,3,4}
wdbc <- wdbc %>% select(c("V2", "V3", "V4"))

##-----------------------------------
# a.
wdbc %>% summarise(n=n())
wdbc %>% group_by(V2) %>% summarise(n=n(), )
glimpse(wdbc)
##------------------
## b.
## Divide the data into a training set of 400 obs
set.seed(101)
train_id_wdbc = sample(nrow(wdbc), 400)
train_wdbc = wdbc[train_id_wdbc, ]
test_wdbc  = wdbc[-train_id_wdbc, ]
##------------------
## c.
## scatterplot
train_wdbc %>% ggplot(aes(x = V3, y = V4, color = factor(V2))) +
        geom_point() +
        xlab("Average radius of the cell nuclei") + ylab("Average texture of the cell nuclei") +
        ggtitle("Scatterplot of radius and texure with color encoded diagnosis")

##------------------
## logistic model
## transfer M variable from character to numeric (0,1)
train_wdbc$Dig <- ifelse (train_wdbc$V2=="M", 1, 0)

glm.wdbc = glm(Dig ~ V3 + V4, family = binomial(link = "logit"), data = train_wdbc)
summary(glm.wdbc)

kable(summary(glm.wdbc)$coefficients)
##------------------
## e.
## extract the coefficients
beta_0 <- coef(glm.wdbc)[1]
beta_1 <- coef(glm.wdbc)[2]
beta_2 <- coef(glm.wdbc)[3]

## calculate the linear predictor
pred <- beta_0 + beta_1*10 + beta_2*12

## calculate the predicted probability
prob <- 1 / (1 + exp(-pred))
prob
# 0.0007619246

# compare with the prediction computed using predict
predict(glm.wdbc, data.frame(V3 = 10, V4 = 12), type = "response")
```

```r
# 0.0007619246

# The two values are the same.
##------------------
## f.
## Compute the predicted probabilities using the predict function
train_probs <- predict(glm.wdbc, newdata = train_wdbc, type = "response")
test_probs <- predict(glm.wdbc, newdata = test_wdbc, type = "response")

## Use the Bayes rule to compute the predicted outcome
train_prediction <- ifelse(train_probs > 0.5, "M", "B")
test_prediction <- ifelse(test_probs > 0.5, "M", "B")

## Create confusion tables on the training and test sets
train_confusion_table <- table(Prediction = train_prediction, Actual = train_wdbc$V2)
test_confusion_table <- table(Prediction = test_prediction, Actual = test_wdbc$V2)

## Calculate prediction accuracy on the training and test sets
train_accuracy <- sum(diag(train_confusion_table)) / nrow(train_wdbc)
test_accuracy <- sum(diag(test_confusion_table)) / nrow(test_wdbc)

kable(train_confusion_table,
      caption = "Confusion Table for training set")
kable(test_confusion_table,
      caption = "Confusion Table for testing set")
## g.
## specify reasonable ranges for X1 and X2
x1_range <- c(min(train_wdbc$V3), max(train_wdbc$V3))
x2_range <- c(min(train_wdbc$V4), max(train_wdbc$V4))

## create a dense set of possible values for X1 and X2 using expand.grid
dense_set <- expand.grid(V3 = seq(from = x1_range[1], to = x1_range[2], length.out = 100),
                         V4 = seq(from = x2_range[1], to = x2_range[2], length.out = 100))

## use the fitted GLM model to predict the outcome probabilities for each observation in the dense set
dense_set$probs <- predict(glm.wdbc, newdata = dense_set, type = "response")

## compute the predicted outcomes using Bayes rule
dense_set$prediction <- ifelse(dense_set$probs > 0.5, "M", "B")
## Plot the predicted outcomes for probability cutoff of 0.5
ggplot(data = dense_set, aes(x = V3, y = V4, color = factor(prediction))) +
        geom_point(alpha = 0.2) +
        geom_point(data = train_wdbc, aes(x = V3, y = V4, color = factor(V2))) +
        scale_color_discrete(name = "Prediction") +
        ggtitle("Logistic Regression Decision Boundary (Cutoff = 0.5)") +
        xlab("Average radius") + ylab("Average texture")
dense_set$prediction_25 <- ifelse(dense_set$probs > 0.25, "M", "B")
ggplot(data = dense_set, aes(x = V3, y = V4, color = factor(prediction_25))) +
        geom_point(alpha = 0.2) +
        geom_point(data = train_wdbc, aes(x = V3, y = V4, color = factor(V2))) +
        scale_color_discrete(name = "Prediction") +
        ggtitle("Logistic Regression Decision Boundary (Cutoff = 0.25)") +
        xlab("Average radius") + ylab("Average texture")
```

```
dense_set$prediction_75 <- ifelse(dense_set$probs > 0.75, "M", "B")
ggplot(data = dense_set, aes(x = V3, y = V4, color = factor(prediction_75))) +
        geom_point(alpha = 0.2) +
        geom_point(data = train_wdbc, aes(x = V3, y = V4, color = factor(V2))) +
        scale_color_discrete(name = "Prediction") +
        ggtitle("Logistic Regression Decision Boundary (Cutoff = 0.75)") +
        xlab("Average radius") + ylab("Average texture")
##------------------
## h.
test_probs <- predict(glm.wdbc, newdata = test_wdbc, type = "response")
## calculate AUC score
roc_score = roc(response = test_wdbc$V2, predictor = test_probs)
#roc_score

## plot the ROC curve
ggroc(roc_score, linetype=1, size = 1) +
        xlab("FPR") + ylab("TPR") +
        ggtitle("Test ROC curve for the Logistic Regression Model")

##----------------------------------- LDA
## a.
## linear discriminant model
lda.wdbc = lda(Dig ~ V3 + V4, data = train_wdbc, center = TRUE)

lda_table <- cbind(lda.wdbc$prior, lda.wdbc$mean)
colnames(lda_table) <- c("Prior prob of groups", "Group means(radius)", "Group means(texture)")

kable(lda_table)
##------------------
## b
X_train <- train_wdbc %>% select(c(V3, V4))
X_test <- test_wdbc %>% select(c(V3, V4))

## fit the model and calculate post prob
train_pred.lda <- predict(lda.wdbc, X_train)$posterior
test_pred.lda <- predict(lda.wdbc, X_test)$posterior

## compute outcome Y based on the post prob
train_outcome.lda <- apply(train_pred.lda, 1, which.max)
## outcome variable = 1 or 2
## transfer them to B or M
train_outcome.lda <- data.frame(train_outcome.lda)
train_outcome.lda$train_outcome.lda <-
        ifelse(train_outcome.lda$train_outcome.lda==1, "B", "M")

## Do it again, but using testing set
test_outcome.lda <- apply(test_pred.lda, 1, which.max)
## outcome variable = 1 or 2
## transfer them to B or M
test_outcome.lda <- data.frame(test_outcome.lda)
test_outcome.lda$test_outcome.lda <-
        ifelse(test_outcome.lda$test_outcome.lda==1, "B", "M")
```

```r
## confusion matrix
train_confusion.lda <- table(Prediction = train_outcome.lda$train_outcome.lda,
                             Actual = train_wdbc$V2)
test_confusion.lda <- table(Prediction = test_outcome.lda$test_outcome.lda,
                            Actual = test_wdbc$V2)

## prediction accuracy
train_accuracy.lda <- sum(diag(train_confusion.lda)) / nrow(train_wdbc)
test_accuracy.lda <- sum(diag(test_confusion.lda)) / nrow(test_wdbc)

kable(train_confusion.lda,
      caption = "Confusion table for training set - LDA")

kable(test_confusion.lda,
      caption = "Confusion table for testing set - LDA")

##------------------
## c.
## use the fitted model to predict the outcome probabilities for each observation in the dense set
probs.lda <- predict(lda.wdbc, newdata = dense_set, type = "response")$posterior

# Compute the predicted outcomes using Bayes rule with different probability cutoff values
pred_outcomes_05 = ifelse(probs.lda[,1] > probs.lda[,2], "B", "M")
pred_outcomes_25 = ifelse(probs.lda[,1] > 0.25, "B", "M")
pred_outcomes_75 = ifelse(probs.lda[,1] > 0.75, "B", "M")

## Plot the predicted outcomes for probability cutoff of 0.5
ggplot(data = dense_set, aes(x = V3, y = V4, color = factor(pred_outcomes_05))) +
        geom_point(alpha = 0.2) +
        geom_point(data = train_wdbc, aes(x = V3, y = V4, color = factor(V2))) +
        scale_color_discrete(name = "Prediction") +
        ggtitle("LDA Decision Boundary (Cutoff = 0.5)") +
        xlab("Average radius") + ylab("Average texture")
# Plot the predicted outcomes for a probability cutoff of 0.25
ggplot(data = dense_set, aes(x = V3, y = V4, color = factor(pred_outcomes_25))) +
        geom_point(alpha = 0.2) +
        geom_point(data = train_wdbc, aes(x = V3, y = V4, color = factor(V2))) +
        scale_color_discrete(name = "Prediction") +
        ggtitle("LDA Decision Boundary (Cutoff = 0.25)") +
        xlab("Average radius") + ylab("Average texture")
# Plot the predicted outcomes for a probability cutoff of 0.75
ggplot(data = dense_set, aes(x = V3, y = V4, color = factor(pred_outcomes_75))) +
        geom_point(alpha = 0.2) +
        geom_point(data = train_wdbc, aes(x = V3, y = V4, color = factor(V2))) +
        scale_color_discrete(name = "Prediction") +
        ggtitle("LDA Decision Boundary (Cutoff = 0.75)") +
        xlab("Average radius") + ylab("Average texture")
##------------------
## d.
test.lda_probs <- predict(lda.wdbc, newdata = test_wdbc,  type = "response")$posterior[, 2]
## calculate AUC score
roc.lda_score = roc(response = test_wdbc$V2, predictor = test.lda_probs)
#roc.lda_score
```

```r
# 0.9396

## plot the ROC curve
ggroc(roc.lda_score, linetype=1, size = 1) +
        xlab("FPR") + ylab("TPR") +
        ggtitle("Test ROC curve for LDA")

roc.lda_score
# 0.9396
## a.
## quadratic discriminant analysis
qda.wdbc = qda(Dig ~ V3 + V4, data = train_wdbc, center = TRUE)
qda.wdbc
qda_table <- cbind(qda.wdbc$prior, qda.wdbc$mean)
colnames(qda_table) <- c("Prior prob of groups", "Group means(radius)", "Group means(texture)")

kable(qda_table)
##-------------------
## b.
## fit the model and calculate post prob
train_pred.qda <- predict(qda.wdbc, X_train)$posterior
test_pred.qda <- predict(qda.wdbc, X_test)$posterior

## compute outcome Y based on the post prob on training set
train_outcome.qda <- apply(train_pred.qda, 1, which.max)
## outcome variable = 1 or 2
## transfer them to B or M
train_outcome.qda <- data.frame(train_outcome.qda)
train_outcome.qda$train_outcome.qda <-
        ifelse(train_outcome.qda$train_outcome.qda==1, "B", "M")

## compute outcome Y based on the post prob on testing set
test_outcome.qda <- apply(test_pred.qda, 1, which.max)
## outcome variable = 1 or 2
## transfer them to B or M
test_outcome.qda <- data.frame(test_outcome.qda)
test_outcome.qda$test_outcome.qda <-
        ifelse(test_outcome.qda$test_outcome.qda==1, "B", "M")

## confusion matrix
train_confusion.qda <- table(Prediction = train_outcome.qda$train_outcome.qda,
                             Actual = train_wdbc$V2)
test_confusion.qda <- table(Prediction = test_outcome.qda$test_outcome.qda,
                            Actual = test_wdbc$V2)

## prediction accuracy
train_accuracy.qda <- sum(diag(train_confusion.qda)) / nrow(train_wdbc)
test_accuracy.qda <- sum(diag(test_confusion.qda)) / nrow(test_wdbc)

kable(train_confusion.qda,
      caption = "Confusion table for training set - QDA")

kable(test_confusion.qda,
```

```
      caption = "Confusion table for testing set - QDA")

##------------------
## c.
## use the fitted GLM model to predict the outcome probabilities for each observation in the dense set
probs.qda <- predict(qda.wdbc, newdata = dense_set, type = "response")$posterior

# Compute the predicted outcomes using Bayes rule with different probability cutoff values
pred_outcomes_05.qda = ifelse(probs.qda[,1] > probs.qda[,2], "B", "M")
pred_outcomes_25.qda = ifelse(probs.qda[,1] > 0.25, "B", "M")
pred_outcomes_75.qda = ifelse(probs.qda[,1] > 0.75, "B", "M")

## Plot the predicted outcomes for probability cutoff of 0.5
ggplot(data = dense_set, aes(x = V3, y = V4, color = factor(pred_outcomes_05.qda))) +
        geom_point(alpha = 0.2) +
        geom_point(data = train_wdbc, aes(x = V3, y = V4, color = factor(V2))) +
        scale_color_discrete(name = "Prediction") +
        ggtitle("QDA Decision Boundary (Cutoff = 0.5)") +
        xlab("Average radius") + ylab("Average texture")
## Plot the predicted outcomes for probability cutoff of 0.25
ggplot(data = dense_set, aes(x = V3, y = V4, color = factor(pred_outcomes_25.qda))) +
        geom_point(alpha = 0.2) +
        geom_point(data = train_wdbc, aes(x = V3, y = V4, color = factor(V2))) +
        scale_color_discrete(name = "Prediction") +
        ggtitle("QDA Decision Boundary (Cutoff = 0.25)") +
        xlab("Average radius") + ylab("Average texture")
## Plot the predicted outcomes for probability cutoff of 0.75
ggplot(data = dense_set, aes(x = V3, y = V4, color = factor(pred_outcomes_75.qda))) +
        geom_point(alpha = 0.2) +
        geom_point(data = train_wdbc, aes(x = V3, y = V4, color = factor(V2))) +
        scale_color_discrete(name = "Prediction") +
        ggtitle("QDA Decision Boundary (Cutoff = 0.75)") +
        xlab("Average radius") + ylab("Average texture")
##------------------
## d.
test.qda_probs <- predict(qda.wdbc, newdata = test_wdbc, type = "response")$posterior
## calculate AUC score
roc.qda_score = roc(response = test_wdbc$V2, predictor = test.qda_probs[, 2])
#roc.qda_score
# 0.9361

## plot the ROC curve
ggroc(roc.qda_score, linetype=1, size = 1) +
        xlab("FPR") + ylab("TPR") +
        ggtitle("Test ROC curve for QDA")

roc.qda_score
# 0.9361
##---------------------------------- KNN
## a.
## knn
k_values <- c(1, 2, 3, 4, 20)
for (k in k_values) {
```

```
        # train set
        knn.train.label <- knn(train = train_wdbc[,c("V3", "V4")],
                               test = train_wdbc[,c("V3", "V4")],
                               cl = train_wdbc$V2, k = k)

        # confusion table
        confusion_table_train <- table(knn.train.label, train_wdbc$V2)
        # prediction accuracy
        accuracy_train <- sum(diag(confusion_table_train)) / sum(confusion_table_train)

        # test set
        knn.test.label <- knn(train = train_wdbc[,c("V3", "V4")],
                              test = test_wdbc[,c("V3", "V4")],
                              cl = train_wdbc$V2, k = k)

        # confusion table - test set
        confusion_table_test <- table(knn.test.label, test_wdbc$V2)
        # prediction accuracy
        accuracy_test <- sum(diag(confusion_table_test)) / sum(confusion_table_test)

        # Print the results
        cat("For k =", k, ":\n")
        cat("Training set confusion table:\n")
        print(confusion_table_train)
        cat("Training set accuracy:", accuracy_train, "\n")
        cat("Test set confusion table:\n")
        print(confusion_table_test)
        cat("Test set accuracy:", accuracy_test, "\n\n")
}

##------------------
## b.
## k=1
model.knn.1 = knn(train = train_wdbc[,c("V3", "V4")],
            test = dense_set[,c("V3", "V4")],
            cl = train_wdbc$V2,
            k = 1)

## Plot
ggplot(data = dense_set, aes(x = V3, y = V4, color = factor(model.knn.1))) +
        geom_point(alpha = 0.2) +
        geom_point(data = train_wdbc, aes(x = V3, y = V4, color = factor(V2))) +
        scale_color_discrete(name = "Prediction") +
        ggtitle("KNN Decision Boundary (k=1)") +
        xlab("Average radius") + ylab("Average texture")

## k=2
model.knn.2 = knn(train = train_wdbc[,c("V3", "V4")],
                  test = dense_set[,c("V3", "V4")],
                  cl = train_wdbc$V2,
                  k = 2)

## Plot
```

```r
ggplot(data = dense_set, aes(x = V3, y = V4, color = factor(model.knn.2))) +
        geom_point(alpha = 0.2) +
        geom_point(data = train_wdbc, aes(x = V3, y = V4, color = factor(V2))) +
        scale_color_discrete(name = "Prediction") +
        ggtitle("KNN Decision Boundary (k=2)") +
        xlab("Average radius") + ylab("Average texture")

## k=3
model.knn.3 = knn(train = train_wdbc[,c("V3", "V4")],
                  test = dense_set[,c("V3", "V4")],
                  cl = train_wdbc$V2,
                  k = 3)

## Plot
ggplot(data = dense_set, aes(x = V3, y = V4, color = factor(model.knn.3))) +
        geom_point(alpha = 0.2) +
        geom_point(data = train_wdbc, aes(x = V3, y = V4, color = factor(V2))) +
        scale_color_discrete(name = "Prediction") +
        ggtitle("KNN Decision Boundary (k=3)") +
        xlab("Average radius") + ylab("Average texture")

## k=4
model.knn.4 = knn(train = train_wdbc[,c("V3", "V4")],
                  test = dense_set[,c("V3", "V4")],
                  cl = train_wdbc$V2,
                  k = 4)

## Plot
ggplot(data = dense_set, aes(x = V3, y = V4, color = factor(model.knn.4))) +
        geom_point(alpha = 0.2) +
        geom_point(data = train_wdbc, aes(x = V3, y = V4, color = factor(V2))) +
        scale_color_discrete(name = "Prediction") +
        ggtitle("KNN Decision Boundary (k=4)") +
        xlab("Average radius") + ylab("Average texture")

## k=20
model.knn.20 = knn(train = train_wdbc[,c("V3", "V4")],
                  test = dense_set[,c("V3", "V4")],
                  cl = train_wdbc$V2,
                  k = 20)

## Plot
ggplot(data = dense_set, aes(x = V3, y = V4, color = factor(model.knn.20))) +
        geom_point(alpha = 0.2) +
        geom_point(data = train_wdbc, aes(x = V3, y = V4, color = factor(V2))) +
        scale_color_discrete(name = "Prediction") +
        ggtitle("KNN Decision Boundary (k=20)") +
        xlab("Average radius") + ylab("Average texture")

##------------------
## c.
## plot the prediction accuracy
train_accuracy = numeric(20)
```

```
test_accuracy = numeric(20)

for (i in 1:20) {
        knn.label <- knn(train = train_wdbc[,c("V3","V4")],
                         test = train_wdbc[,c("V3","V4")],
                         cl = train_wdbc$V2, k = i)
        train_accuracy[i] <- mean(knn.label == train_wdbc$V2)
        test_accuracy[i] <- mean(knn.label == test_wdbc$V2)
}

for (i in 1:20) {
        knn.label <- knn(train = train_wdbc[,c("V3", "V4")],
                          test = test_wdbc[,c("V3", "V4")],
                          cl = train_wdbc$V2, k = i)
        test_accuracy[i] <- mean(knn.label == test_wdbc$V2)
}

## train set
plot(1:20, train_accuracy, type="l", col="blue",
     xlab="k", ylab="accuracy",
     main="Prediction accuracy for different value of k (train set)")

## test set
plot(1:20, test_accuracy, type="l", col="red",
     xlab="k", ylab="accuracy",
     main="Prediction accuracy for different value of k (test set)")
```