**Progressive Education Society's**

**Modern College of Engineering, Pune**

**MCA Department**

**A.Y.2023-24**

**(410908) : Data Science Laboratory**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Class: SY-MCA Shift / Div : B (S1) Roll Number : 52119**

**Name: Harsh Ghodke Assignment No: 3 Date of Implementation: 19/09/2024 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Q. 1) We have four things: grape, green bean, nuts and orange with two characteristics: sweetness (8, 3, 3, 7) and Crunchiness (5, 7, 6, 3). Among them two are fruits, one is**

**protein and one is vegetable. Suppose we wanted to classify tomato into one of the**

**classes. Is tomato a fruit, vegetable or protein? Tomato has the following**

**characteristics: sweetness = 6, crunchiness = 4. Let’s add Carrot with characteristics**

**sweetness = 4 and crunchiness = 9 keep k=1. Try for k=4 also.**

**Ans:-**

# Given data with characteristics: Sweetness and Crunchiness

items <- data.frame(

Name = c("Grape", "Green Bean", "Nuts", "Orange"),

Sweetness = c(8, 3, 3, 7),

Crunchiness = c(5, 7, 6, 3),

Class = c("Fruit", "Vegetable", "Protein", "Fruit")

)

# New items to classify

new\_items <- data.frame(

Name = c("Tomato", "Carrot"),

Sweetness = c(6, 4),

Crunchiness = c(4, 9)

)

# Function to calculate Euclidean distance

euclidean\_distance <- function(point1, point2) {

sqrt(sum((point1 - point2) ^ 2))

}

# Function to classify using k-NN

classify\_knn <- function(new\_item, k) {

distances <- apply(items[, 2:3], 1, euclidean\_distance, new\_item)

sorted\_indices <- order(distances)

nearest\_classes <- items$Class[sorted\_indices][1:k]

most\_common\_class <- names(sort(table(nearest\_classes), decreasing = TRUE))[1]

return(most\_common\_class)

}

# Classify new items for k = 1 and k = 4

results <- data.frame(Name = new\_items$Name)

for (k in c(1, 4)) {

results[[paste0("k\_", k)]] <- sapply(1:nrow(new\_items), function(i) {

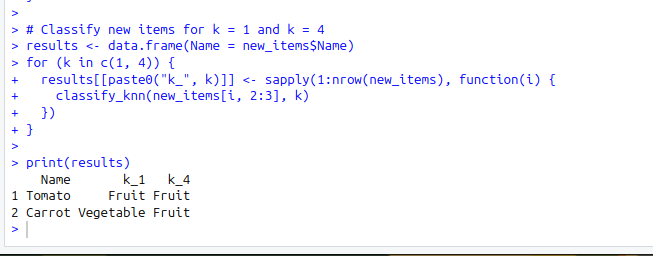
classify\_knn(new\_items[i, 2:3], k)

})

}

print(results)

**Output:-**

****

**Q.2) Using Titanic.CSV file, predict which people are more likely to survive after the collision with the iceberg using Decision Trees.**

**Ans:-**

# Load necessary libraries

library(rpart) # For building the decision tree

library(rpart.plot) # For plotting the decision tree

# Load the Titanic dataset

titanic\_data <- read.csv("Titanic.csv", stringsAsFactors = TRUE)

# View the first few rows of the dataset

head(titanic\_data)

# Data Preprocessing

# Removing columns that won't be helpful for prediction

titanic\_data <- titanic\_data[, c("Survived", "Pclass", "Sex", "Age", "SibSp", "Parch", "Fare", "Embarked")]

# Handle missing values in 'Age' with the median value

titanic\_data$Age[is.na(titanic\_data$Age)] <- median(titanic\_data$Age, na.rm = TRUE)

# Remove rows with missing 'Embarked' values

titanic\_data <- titanic\_data[!is.na(titanic\_data$Embarked), ]

# Convert categorical variables to factors

titanic\_data$Survived <- factor(titanic\_data$Survived)

titanic\_data$Pclass <- factor(titanic\_data$Pclass)

titanic\_data$Sex <- factor(titanic\_data$Sex)

titanic\_data$Embarked <- factor(titanic\_data$Embarked)

# Build the Decision Tree model

decision\_tree <- rpart(Survived ~ Pclass + Sex + Age + SibSp + Parch + Fare + Embarked,

data = titanic\_data, method = "class",

control = rpart.control(minsplit = 20, cp = 0.01))

# Display the decision tree

print(decision\_tree)

rpart.plot(decision\_tree, type = 4, extra = 101)

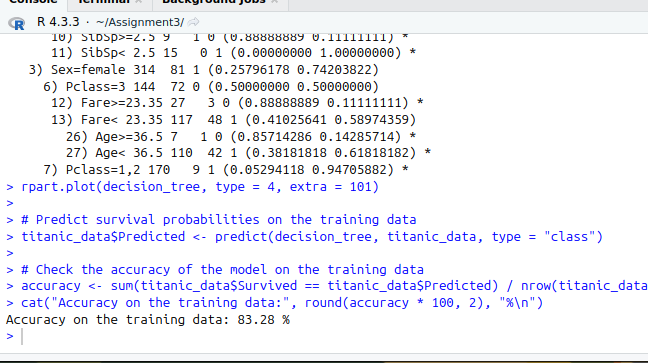
# Predict survival probabilities on the training data

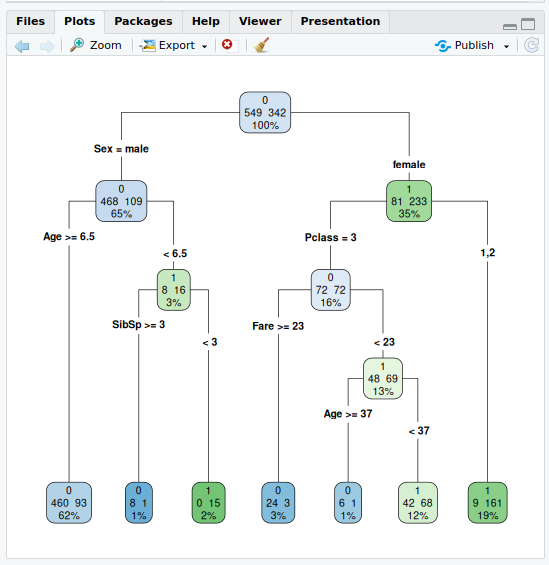
titanic\_data$Predicted <- predict(decision\_tree, titanic\_data, type = "class")

# Check the accuracy of the model on the training data

accuracy <- sum(titanic\_data$Survived == titanic\_data$Predicted) / nrow(titanic\_data)

cat("Accuracy on the training data:", round(accuracy \* 100, 2), "%\n")

**output:-**

****

**Q.3) Load the tissue\_gene\_expression dataset. Run a k-means clustering on the data with K=7. Make a table comparing the identified clusters to the actual tissue types. Run the algorithm several times to see how the answer changes.**

**Ans:-**

# Load necessary libraries

library(dslabs)

library(dplyr)

# Load the tissue\_gene\_expression dataset

data("tissue\_gene\_expression")

# Extract the expression data and tissue labels

gene\_data <- tissue\_gene\_expression$x

tissue\_labels <- tissue\_gene\_expression$y

# Perform k-means clustering with K = 7

set.seed(123) # Setting a seed for reproducibility

kmeans\_result <- kmeans(gene\_data, centers = 7)

# Create a table comparing the identified clusters to the actual tissue types

comparison\_table <- table(Cluster = kmeans\_result$cluster, Tissue = tissue\_labels)

print(comparison\_table)

# Check how many times the answer changes by running the algorithm multiple times

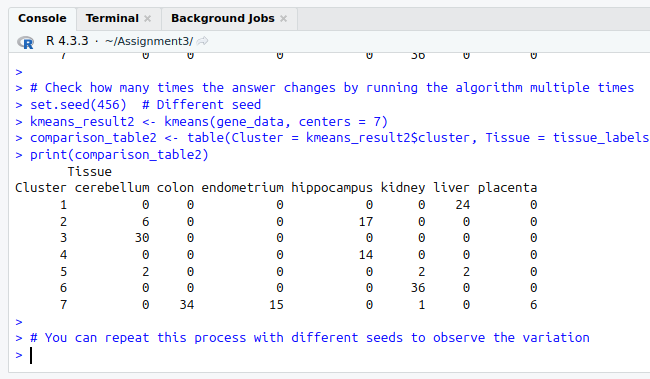
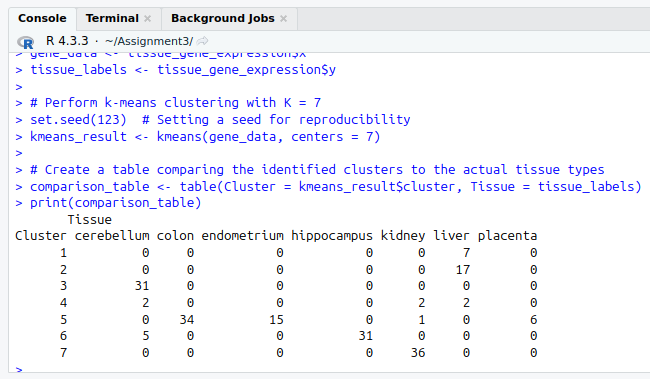
set.seed(456) # Different seed

kmeans\_result2 <- kmeans(gene\_data, centers = 7)

comparison\_table2 <- table(Cluster = kmeans\_result2$cluster, Tissue = tissue\_labels)

print(comparison\_table2)

**Output:-**



**Q.4) Plot the distribution of distances between data points and their fifth nearest neighbors**

**using the kNNdistplot function from the dbscan package.**

**Examine the plot and find a tentative threshold at which distances start increasing**

**quickly. On the same plot, draw a horizontal line at the level of the threshold (use Iris**

**dataset)**

**Ans:-**

# Load necessary libraries

library(dbscan) # For kNNdistplot

library(datasets) # For the Iris dataset

# Load the Iris dataset

data("iris")

# Remove the Species column since we are only interested in numerical features

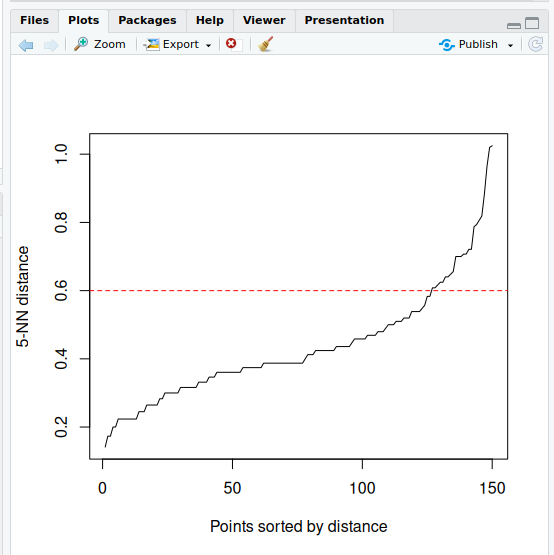
iris\_data <- iris[, -5]

# Use kNNdistplot from the dbscan package to plot distances to the 5th nearest neighbors

k <- 5

kNNdistplot(iris\_data, k = k)

abline(h = 0.6, col = "purple", lty = 2) # Draw a horizontal line at the tentative threshold



**Output:-**