NAME:ZAROON TAHIR

REG NO :FA21-BSE-168

ASSIGNMENT:2

Part I: Classification Models

```
install.packages("caret")
```

```
install.packages(c("ggplot2", "lattice", "e1071", "class", "rpart"))
```

```
# Load necessary libraries
library(caret)      # For data splitting and model evaluation
library(rpart)      # For Decision Tree
library(e1071)      # For Naive Bayes
library(class)      # For KNN

# This step ensures all required libraries are loaded for building and evaluating the models.
```

Step 2: Load and Preprocess the Dataset

```
# Load the dataset
df <- read.csv("drug200.csv")

# Display the first few rows
head(df)

# Convert categorical variables to factors
df$Sex <- as.factor(df$Sex)
df$BP <- as.factor(df$BP)
df$Cholesterol <- as.factor(df$Cholesterol)
df$Drug <- as.factor(df$Drug)

# Split data into training (70%) and testing (30%)
set.seed(42)
trainIndex <- createDataPartition(df$Drug, p = 0.7, list = FALSE)
trainData <- df[trainIndex, ]
testData <- df[-trainIndex, ]

# This step loads the dataset, preprocesses it (factorizing categorical variables), and splits it into training and testing sets.
```

Step 3: Decision Tree Classifier

```
# Build the Decision Tree model
dt_model <- rpart(Drug ~ ., data = trainData, method = "class")

# Make predictions on the test set
dt_pred <- predict(dt_model, testData, type = "class")

# Evaluate the model
dt_conf_matrix <- confusionMatrix(dt_pred, testData$Drug)
print("Decision Tree Classifier Results:")
print(dt_conf_matrix)

# This cell builds a Decision Tree model, predicts test data, and evaluates the confusion matrix and accuracy.
```

Step 4: Naive Bayes Classifier

```
# Build the Naive Bayes model
nb_model <- naiveBayes(Drug ~ ., data = trainData)

# Make predictions on the test set
nb_pred <- predict(nb_model, testData)

# Evaluate the model
nb_conf_matrix <- confusionMatrix(nb_pred, testData$Drug)
print("Naive Bayes Classifier Results:")
print(nb_conf_matrix)

# This cell builds a Naive Bayes model, predicts test data, and evaluates the confusion matrix and accuracy.
```

Step 5: KNN Classifier

```
# Prepare data for KNN (numeric-only features)
trainX <- trainData[, -which(names(trainData) %in% c("Drug", "Sex", "BP", "Cholesterol"))] # Exclude non-numeric columns
testX <- testData[, -which(names(testData) %in% c("Drug", "Sex", "BP", "Cholesterol"))] # Exclude non-numeric columns
trainY <- trainData$Drug

# Standardize numeric features
trainX <- scale(trainX)
testX <- scale(testX)

# Build the KNN model
set.seed(42)
knn_pred <- knn(train = trainX, test = testX, cl = trainY, k = 3)

# Evaluate the model
knn_conf_matrix <- confusionMatrix(knn_pred, testData$Drug)
print("KNN Classifier Results:")
print(knn_conf_matrix)

# This cell builds a KNN model (with k=3), predicts test data, and evaluates the confusion matrix and accuracy.
```

Part II: R Functions for Dataset Analysis

Function 1: Calculate the Average Age of Males Who Used Drug C

```
# Function to calculate average age of males who used drug C
avg_age_males_drug_c <- function(data) {
  males_drug_c <- subset(data, Sex == "M" & Drug == "C")
  return(mean(males_drug_c$Age))
}

# Call the function
avg_age <- avg_age_males_drug_c(df)
print(paste("Average age of males who used Drug C:", avg_age))
```

Function 2: Females with HIGH BP and HIGH **Cholesterol**

```
# Function to find drugs used by females with HIGH BP and HIGH cholesterol
drugs_for_females_high <- function(data) {
  females_high <- subset(data, Sex == "F" & BP == "HIGH" & Cholesterol == "HIGH")
  return(unique(females_high$Drug))
}

# Call the function
drugs <- drugs_for_females_high(df)
```

```
print("Drugs for females with HIGH BP and HIGH cholesterol:")
print(drugs)
```

Function 3: Average Na-K Levels Based on BP Categories:

```
# Function to calculate average Na-K levels for different BP categories
avg_na_k_bp <- function(data) {
  avg_high <- mean(subset(data, BP == "HIGH")$Na_to_K)
  avg_low <- mean(subset(data, BP == "LOW")$Na_to_K)
  avg_normal <- mean(subset(data, BP == "NORMAL")$Na_to_K)

  return(list(
    High_BP = avg_high,
    Low_BP = avg_low,
    Normal_BP = avg_normal
  ))
}

# Call the function
avg_na_k <- avg_na_k_bp(df)
print("Average Na-K levels based on BP categories:")
print(avg_na_k)
```

Start coding or generate with AI.