Q1)

Data Pre-processingWrite a python program to find all null values in a given data set and remove them.

(Download dataset from github.com) using R Studio

Answer:

#Total libraries = 1

library(tidyverse) install this

```
url <- "https://raw.githubusercontent.com/suneet10/DataPreprocessing/main/Data.csv"
dataset <- read.csv(url)

cat("Original Dataset:\n")
print(dataset)

cat("\nCount of null values in each column:\n")
null_values <- sapply(dataset, function(x) sum(is.na(x)))
print(null_values)
cleaned_dataset <- na.omit(dataset)

cat("\nDataset after removing null values:\n")
print(cleaned_dataset)
```

Q2)

Write a python program to implement complete data pre-processing in a given data set. missing value, encoding categorical value, Splitting thedataset into the training and test sets and feature scaling. (Download dataset fromgithub.com). Using R Studio

Answer :

1st

```
# Load necessary libraries
library(dplyr)
library(caTools)
library(e1071)

# Step 1: Load dataset from GitHub
url <- "https://raw.githubusercontent.com/suneet10/DataPreprocessing/main/Data.csv"
dataset <- read.csv(url)

# View the dataset
print("Original Dataset:")
print(head(dataset))

# Step 2: Handle missing values (e.g., replacing missing 'Age' and 'Salary' with mean)
dataset$Age[is.na(dataset$Age)] <- mean(dataset$Age, na.rm = TRUE)
dataset$Salary[is.na(dataset$Salary)] <- mean(dataset$Salary, na.rm = TRUE)
```

```
print("Dataset After Handling Missing Values:")
print(head(dataset))

# Step 3: Encode categorical values (Country and Purchased)
dataset$Country <- as.factor(dataset$Country)
dataset$Purchased <- as.factor(dataset$Purchased)

print("Dataset After Encoding Categorical Values:")
print(head(dataset))

# Step 4: Split the dataset into training and test sets (80% train, 20% test)
set.seed(123)
split <- sample.split(dataset$Purchased, SplitRatio = 0.8)
training_set <- subset(dataset, split == TRUE)
test_set <- subset(dataset, split == FALSE)

print("Training Set:")
print(head(training_set))

print("Test Set:")
print(head(test_set))

# Step 5: Apply feature scaling to numeric columns (Age, Salary)
training_set[, c('Age', 'Salary')] <- scale(training_set[, c('Age', 'Salary')])
```

```
test_set[, c('Age', 'Salary')] <- scale(test_set[, c('Age', 'Salary')])


print("Training Set After Scaling:")

print(head(training_set))


print("Test Set After Scaling:")

print(head(test_set))
```

2nd

```
# Load necessary libraries

library(dplyr)

library(caTools)


# Step 1: Load dataset from GitHub

url <- "https://raw.githubusercontent.com/suneet10/DataPreprocessing/main/Data.csv"

dataset <- read.csv(url)


# View the dataset

print("Original Dataset:")

print(head(dataset))


# Step 2: Handle missing values (replace missing 'Age' and 'Salary' with median)

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

dataset$Salary[is.na(dataset$Salary)] <- median(dataset$Salary, na.rm = TRUE)
```

```
print("Dataset After Handling Missing Values:")
print(head(dataset))

# Step 3: Encode categorical values (Convert Country and Purchased to numeric encoding)
dataset$Country <- as.numeric(factor(dataset$Country))
dataset$Purchased <- as.numeric(factor(dataset$Purchased))

print("Dataset After Encoding Categorical Values:")
print(head(dataset))

# Step 4: Split the dataset into training and test sets (80% train, 20% test)
set.seed(123)
split <- sample.split(dataset$Purchased, SplitRatio = 0.8)
training_set <- subset(dataset, split == TRUE)
test_set <- subset(dataset, split == FALSE)

print("Training Set:")
print(head(training_set))

print("Test Set:")
print(head(test_set))

# Step 5: Apply feature scaling to numeric columns (Age, Salary)
training_set[, c('Age', 'Salary')] <- scale(training_set[, c('Age', 'Salary')])
test_set[, c('Age', 'Salary')] <- scale(test_set[, c('Age', 'Salary')])
```

print("Training Set After Scaling:")

print(head(training_set))

print("Test Set After Scaling:")

print(head(test_set))

Q3)

Consider following dataset

weather=['Sunny','Sunny','Overcast','Rainy','Rainy','Rainy','Over

cast','S

unny','Sunny','Rainy','Sunny','Overcast','Overcast','Rainy']

temp=['Hot','Hot','Hot','Mild','Cool','Cool','Cool','Mild','Cool','Mi

ld','Mi

ld','Mild','Hot','Mild']

play=['No','No','Yes','Yes','Yes','No','Yes','No','Yes','Yes','Yes','Yes','Yes','No']. Use Naïve Bayes algorithm to predict[0:Overcast,

2:Mild]

tuple belongs to which class whether to play the sports or not. Using R studio

Answer :

 # Total Libraries = 1

library(e1071)

Create the dataset

weather <- c('Sunny', 'Sunny', 'Overcast', 'Rainy', 'Rainy', 'Rainy', 'Overcast',

'Sunny', 'Sunny', 'Rainy', 'Sunny', 'Overcast', 'Overcast', 'Rainy')

```
temp <- c('Hot', 'Hot', 'Hot', 'Mild', 'Cool', 'Cool', 'Cool', 'Mild', 'Cool',
        'Mild', 'Mild', 'Mild', 'Hot', 'Mild')
play <- c('No', 'No', 'Yes', 'Yes', 'Yes', 'No', 'Yes', 'No', 'Yes', 'Yes',
        'Yes', 'Yes', 'Yes', 'No')



# Combine into a data frame
data <- data.frame(Weather = weather, Temperature = temp, Play = play)

# Train the Naive Bayes model
model <- naiveBayes(Play ~ Weather + Temperature, data = data)

# Create new data for prediction [Weather = 'Overcast', Temperature = 'Mild']
new_data <- data.frame(Weather = 'Overcast', Temperature = 'Mild')

# Predict using the trained model
prediction <- predict(model, new_data)

# Display the prediction
cat("Prediction for Weather = Overcast and Temperature = Mild:\n")
#cat("Play =", prediction, "\n")
cat("Play =", as.character(prediction), "\n")
View(data)
```

Q4)

Association Rules

Write a Python Programme to read the dataset (“Iris.csv”). dataset download from (https://archive.ics.uci.edu/ml/datasets/iris) and apply Apriori algorithm.

Answer ->

Total libraries = 2

- library(arules) : A package that provides functions for association rule mining, including theApriori algorithm.
- library(arulesViz) : A package used to visualize the association rules generated by the Apriorialgorithm.

```
# Load the dataset

url <- "https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"

iris_data <- read.csv(url, header = FALSE)

colnames(iris_data) <- c("Sepal.Length", "Sepal.Width", "Petal.Length", "Petal.Width", "Species")


# Preprocessing the dataset - Discretize numeric features into categories

iris_data$Sepal.Length <- cut(iris_data$Sepal.Length, breaks = 3, labels = c("Short", "Medium", "Long"))

iris_data$Sepal.Width <- cut(iris_data$Sepal.Width, breaks = 3, labels = c("Narrow", "Medium", "Wide"))

iris_data$Petal.Length <- cut(iris_data$Petal.Length, breaks = 3, labels = c("Short", "Medium", "Long"))

iris_data$Petal.Width <- cut(iris_data$Petal.Width, breaks = 3, labels = c("Narrow", "Medium", "Wide"))


# Convert the data to transactions format suitable for the Apriori algorithm

iris_transactions <- as(iris_data, "transactions")
```

```
# Step 5: Apply the Apriori algorithm with defined support and confidence thresholds

rules <- apriori(iris_transactions, parameter = list(supp = 0.2, conf = 0.8))


# Inspect the rules

inspect(rules)


# Visualize the top rules based on lift

plot(rules, method = "graph", control = list(type = "items"))


#Optional : Sort and inspect top rules based on lift

top_rules <- sort(rules, by = "lift", decreasing = TRUE)

inspect(top_rules[1:5])
```

Q5)

Write a Python program to read “StudentsPerformance.csv” file. Solve following:

To display the shape of dataset.

To display the top rows of the dataset with their columns.

To display the number of rows randomly.

To display the number of columns and names of the columns. Note: Download dataset from following link :

(https://www.kaggle.com/spscientist/students-performance-in-exams?select=StudentsPerformance.csv)

Main Link →

Link - https://www.kaggle.com/datasets/spscientist/students-performance-in-exams

Answer ->

Total libraries = 2

```
# library(dplyr)

# library(readr)


dataset <- read_csv("StudentsPerformance.csv")

cat("Shape of the dataset:\n")

cat("Number of rows: ", nrow(dataset), "\n")

cat("Number of columns: ", ncol(dataset), "\n")

cat("\nTop rows of the dataset:\n")

print(head(dataset))

set.seed(123)

cat("\nRandom sample of rows:\n")

random_rows <- dataset %>% sample_n(5) # Display 5 random rows

print(random_rows)

cat("\nNumber of columns: ", ncol(dataset), "\n")

cat("Names of the columns:\n")

print(colnames(dataset))
```

Q6)

Regression Analysis and Outlier Detection

Consider following observations/data. And apply simple linear regression and find

out estimated coefficients b1 and b1 Also analyse the performance of the model

(Use sklearn package)

x = np.array([1,2,3,4,5,6,7,8])

y = np.array([7,14,15,18,19,21,26,23])

Answer ->

Total libraries = 1

```
# library(ggplot2)


#Define the data

x <- c(1, 2, 3, 4, 5, 6, 7, 8)

y <- c(7, 14, 15, 18, 19, 21, 26, 23)



# Combine into a Data frame

data <- data.frame(x = x, y = y)


# Fit the Linear Model

model <- lm(y ~ x, data = data)
```

```
# Display the summary of the Model
summary(model)

# Extract Coefficients
coefficients <- coef(model)
b0 <- coefficients["(Intercept)"]
b1 <- coefficients["x"]

cat("Estimated coefficients:\n")
cat("Intercept (b0): ", b0, "\n")
cat("Slope (b1): ", b1, "\n")

# Make Predictions
data$predicted <- predict(model, data)

# Calculate Residuals
data$residuals <- data$y - data$predicted


# Display Residuals
cat("\nResiduals:\n")
print(data$residuals)

# Plot the Data and Regression Line
ggplot(data, aes(x = x, y = y)) +
```

```
geom_point(color = "blue") +
geom_smooth(method = "lm", color = "red") +
labs(title = "Simple Linear Regression",
    x = "X",
    y = "Y") +
theme_minimal()
```