EX.NO: 1

INSTALLATION OF R PROGRAM AND IMPORT PACKAGES

AIM:

To install R and import essential packages in R.

PROCEDURE:

o Install R from the official website: https://cran.r-project.org/

o Install the required package(s) using install.packages() in R.

o Use library() to load the package.

SOURCE CODE:

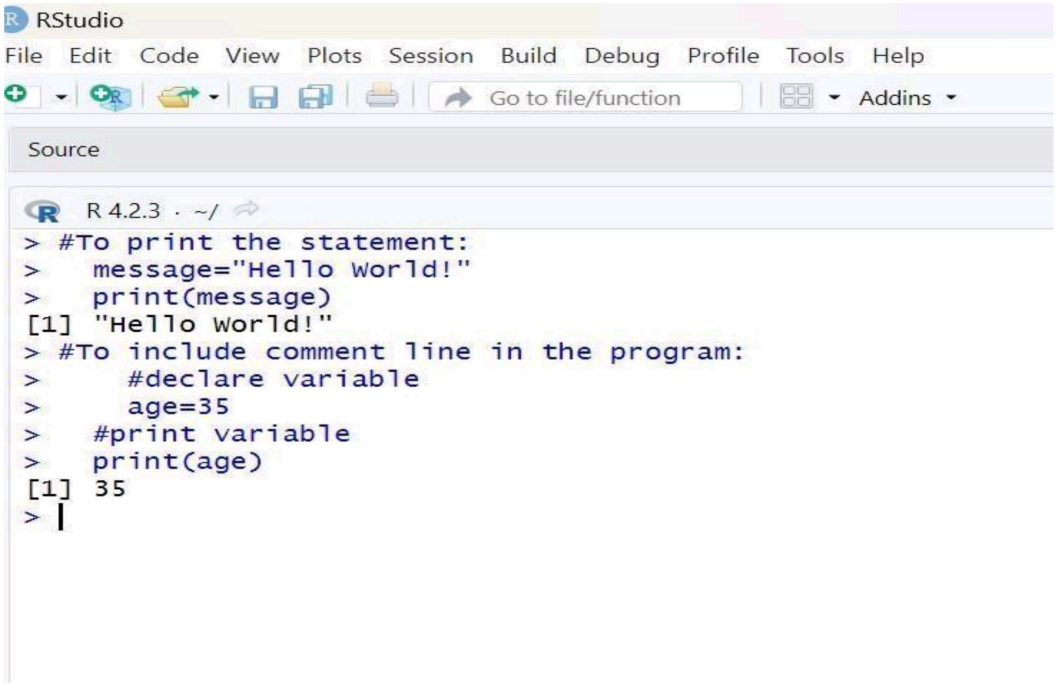
(i) To print the statement:

```
message="HelloWorld!"
print(message)
```

(ii) To include comment line in the program:

```
#declare
variable
age=35
#print
variable
print(age)
```

OUTPUT:



Result:

Thus, the basic R functions have been implemented successfully.

EX.NO: 2

IMPLEMENTATION OF DATA TYPES IN R

AIM:

To implement the basic data types in R programming.

PROCEDURE:

1. Define and Assign Values – Assign logical, numeric, integer, complex, and character values to variables.
2. Check Data Types – Use functions to verify the data type of each variable.
3. Execute the Program – Run the R script in RStudio or any R environment.
4. Verify Output – Observe and confirm the correct data types are displayed.

SOURCE CODE:

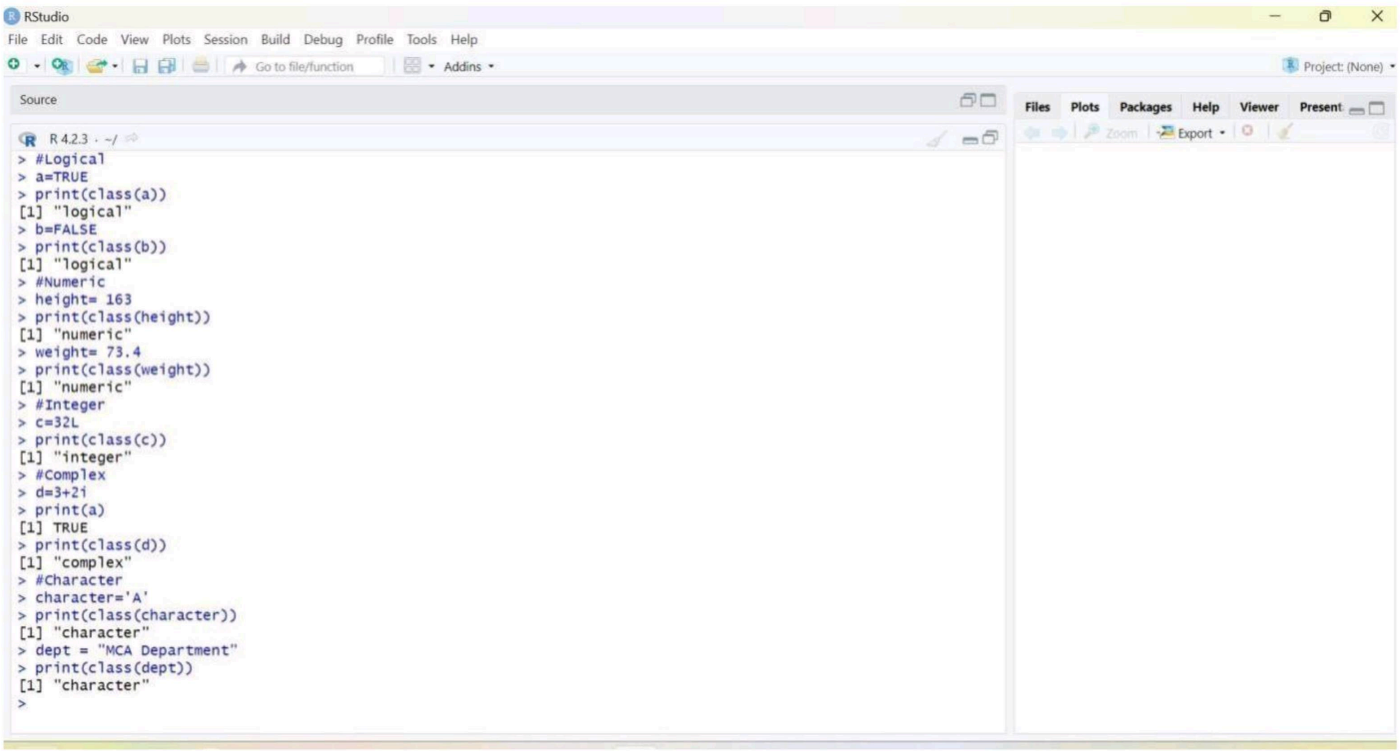
```
#Logical

    a=TRUE

    print(class(a))

    b=FALSE

    print(class(b))

#Numeric

    height= 163

    print(class(height))

    weight= 73.4

    print(class(weigh t))

#Integer

    c=32L

    print(class(c))

#Complex

    d=3+2i

    print(a)
```

```
print(class( d))
```

#Character

```
character='A'
print(class(character))
dept = "MCA Department"
print(class(dept))
```

OUTPUT:



RESULT:

Thus, the basic data types in R programming have been successfully implemented.

EX.NO: 3

IMPLEMENTATION OF DATA STRUCTURES IN R

AIM:

To implement the data structures in R programming.

PROCEDURE:

1. Define and Create Data Structures – Create different data structures, including vectors, lists, matrices, and data frames, by assigning appropriate values.
2. Use Functions to Initialize Data – Utilize functions such as c(), seq(), rep(), list(), matrix(), and data.frame() to define structured data.
3. Execute the Program – Run the R script in an R environment like RStudio to process the data structures.
4. Verify Output – Print the results to confirm the correct implementation of vectors, lists, matrices, and data frames.

SOURCE CODE:

#Vector

```
a=c(10,20,30,40,50)

print(a)

b=seq(1,10,1)
print(b)

c=rep("MCA"4)
print(c)
```

#List

```
a=list("MCA",35,1:8,56.3,month.abb)

print(a)
```

#Matrix

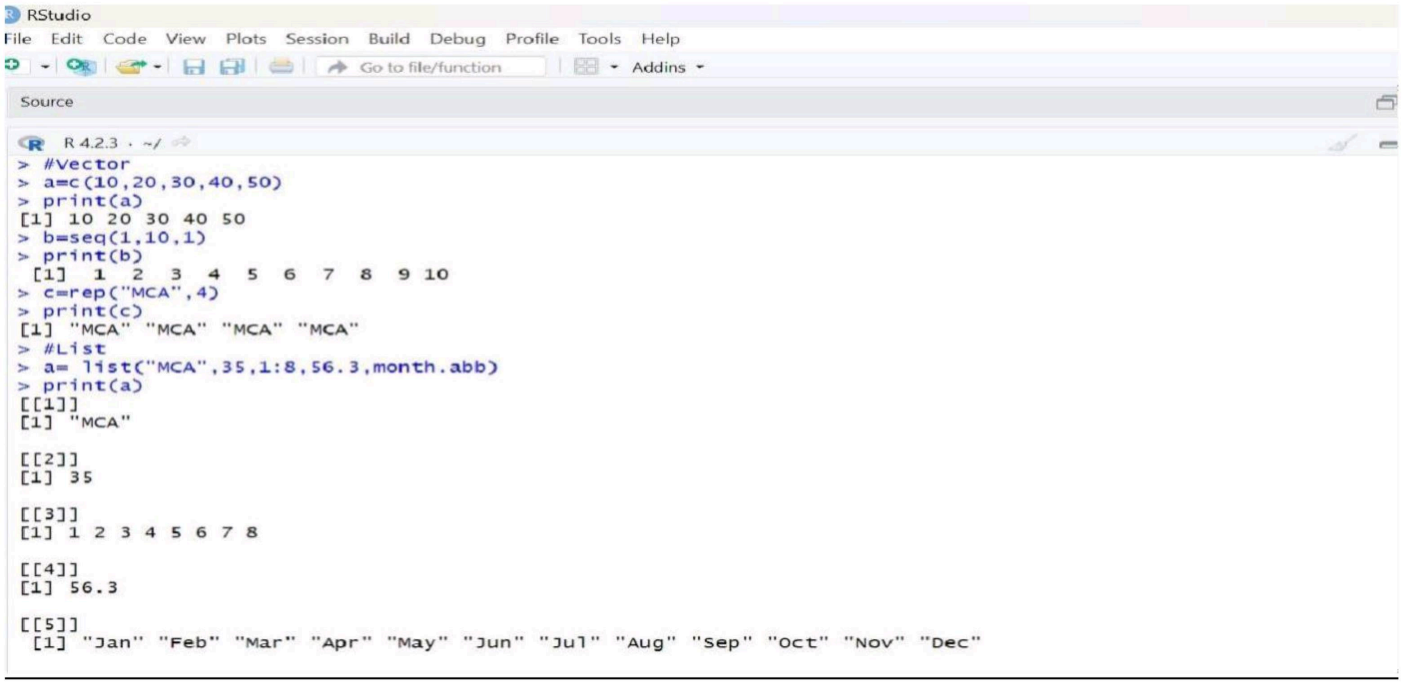
```
a=matrix(data=1:10,nrow=3)
print(a)
```

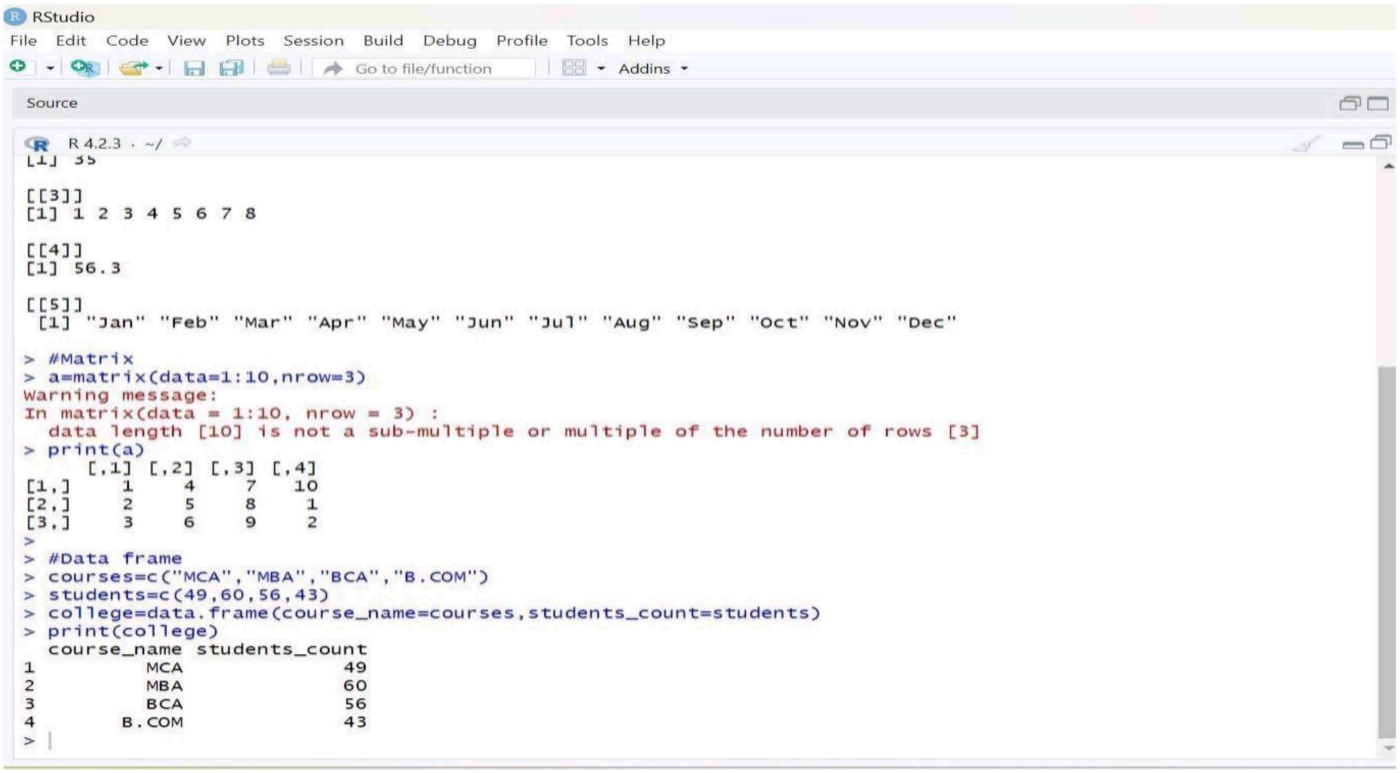
#Data frame

```
courses=c("MCA","MBA","BCA","B.COM")

students=c(49,60,56,43)
college=data.frame(course_name=courses,students_count=students)
```

print(college)

OUTPUT:





RESULT:

Thus, the data structures in R programming have been successfully implemented.

EX.NO: 4

IMPLEMENTATION OF CONTROL STATEMENTS IN R

AIM:

To implement the control statements in R.

PROCEDURE:

1. Define Conditional Statements – Use if and if-else statements to check conditions based on given values.
2. Apply Logical Conditions – Compare values using relational operators (e.g., >, <, >=, <=).
3. Execute the Program – Run the R script in an R environment like RStudio to evaluate the conditions.
4. Verify Output – Observe the printed results to ensure the correct execution of conditional statements.

SOURCE CODE:

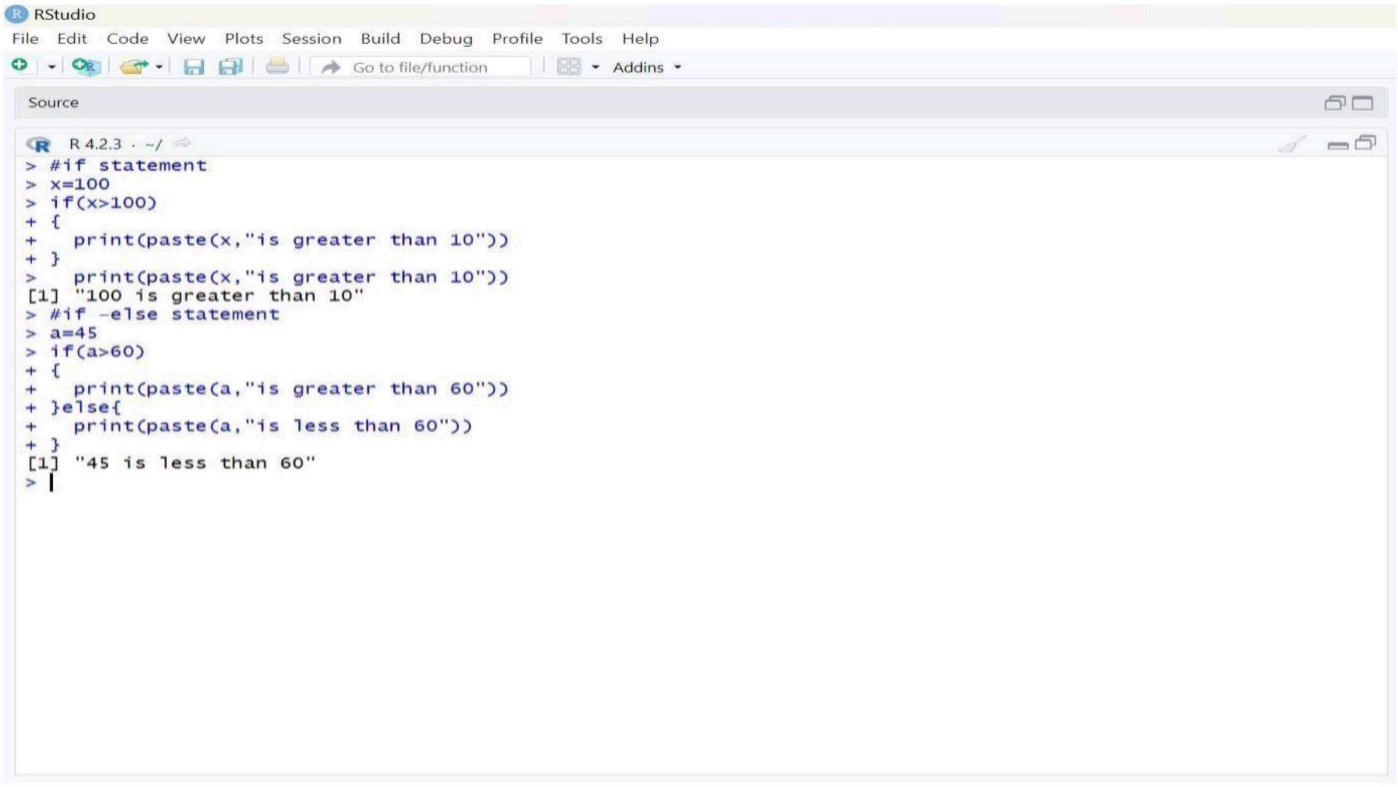
#if statement

```
x=100
if(x>100)
{
print(paste (x,"is greater than 10"))
}
```

#if –else statement

```
a=45
if(a>60)
{
 print(paste(a,"is greater than 60"))
}else{
print(paste(a,"is less than 60"))
}
```

OUTPUT:



RESULT:

Thus, the data structures in R programming have been successfully implemented.

EX.NO: 5

IMPLEMENTATION OF LOOPING STATEMENTS IN R

AIM:

To implement the looping statements in R.

PROCEDURE:

1. Define a Looping Structure – Choose a loop type (for or while) based on the requirement.
2. Set Loop Conditions – Specify the range or condition for iteration.
3. Execute the Loop – Run the loop to iterate over elements or increment values until the condition is met.
4. Verify Output – Print and observe the loop execution to confirm correct iteration.

SOURCE CODE:

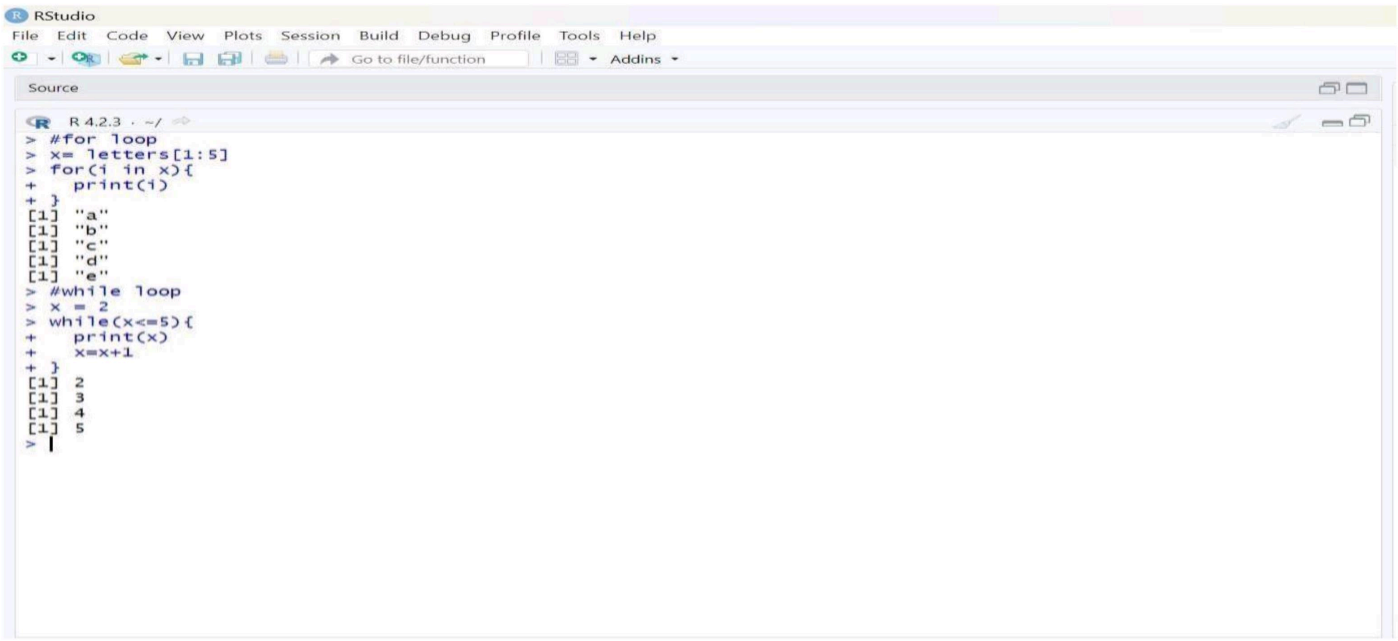
#for loop

```
x=letters[1:5]
for(i in x){
print(i)
}
```

#while loop

```
x = 2 while(x<=5)
{
 print(x)
x=x+1
}
```

OUTPUT:-



RESULT:

Thus, the looping statements in R programming have been successfully implemented.

EX.NO: 6

IMPLEMENTATION OF DECISION TREE CLASSIFICATION ALGORITHM

AIM:

To classify the new instance (new sample) to a target data (grape fruit or orange) using decision tree algorithm in R Studio.

PROCEDURE:

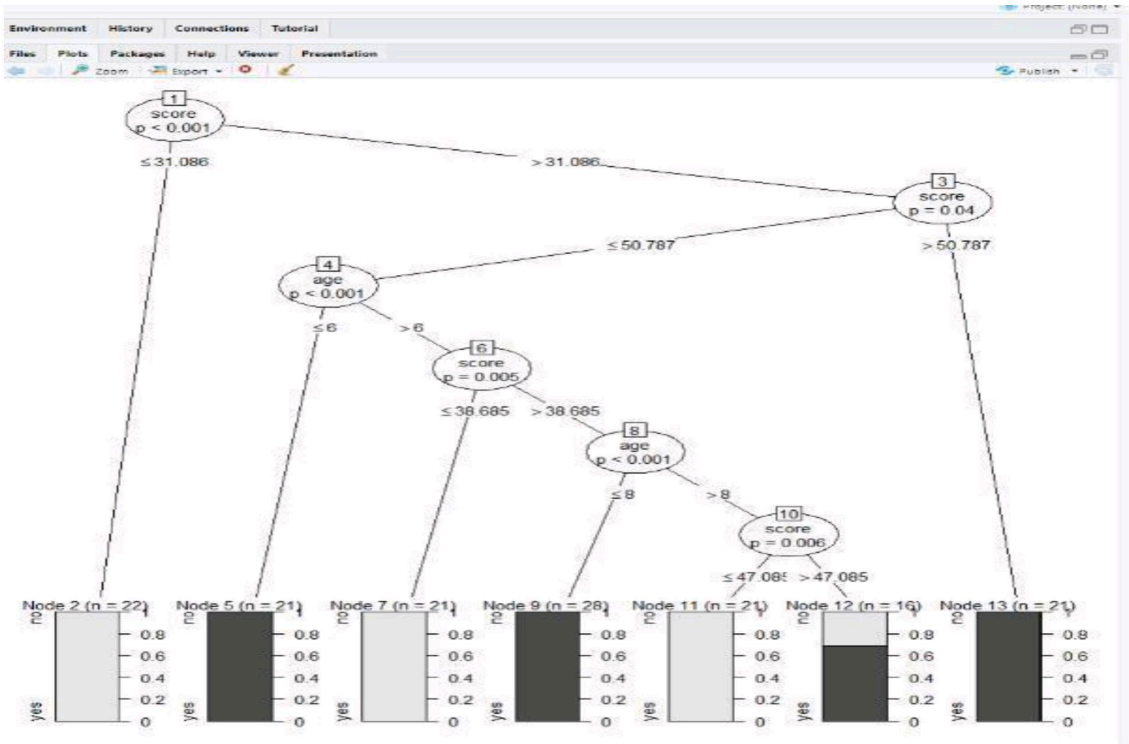
1. Install the necessary package in R Studio such as “rpart” and “rpart.plot”
2. Set the current working directory
3. Save the dataset in the current working directory
4. Read the dataset in the form of csv file using **read.csv** ()
5. Finally create the decision tree and predict the category for the new instance.

SOURCE CODE:

```
library(datasets)
library(caTools)
library(party)
library(dplyr)
library(magrittr)
data("readingSkills")
head(readingSkills)
sample_data = sample.split(readingSkills, SplitRatio = 0.8)
train_data <- subset(readingSkills, sample_data == TRUE)
test_data <- subset(readingSkills, sample_data == FALSE)
model<- ctree(nativeSpeaker ~ ., train_data)
plot(model)
ctree(formula,data)
```

OUTPUT:

```
Warning in install.packages :
  package 'closure' is not available for this version of R

A version of this package for your version of R might be available elsewhere,
see the ideas at
https://cran.r-project.org/doc/manuals/r-patched/R-admin.html#Installing-packages
> library(datasets)
> library(caTools)
> library(party)
> library(dplyr)
> library(magrittr)
>
> data("readingSkills")
> head(readingSkills)
  nativeSpeaker age shoeSize    score
1           yes   5 24.83189 32.29385
2           yes   6 25.95238 36.63105
3            no  11 30.42170 49.60593
4           yes   7 28.66450 40.28456
5           yes  11 31.88207 55.46085
6           yes  10 30.07843 52.83124
> sample_data = sample.split(readingSkills, SplitRatio = 0.8)
> train_data <- subset(readingSkills, sample_data == TRUE)
> test_data <- subset(readingSkills, sample_data == FALSE)
> model<- ctree(nativeSpeaker ~ ., train_data)
> plot(model)
> ctree(formula, data)
Error in x$terms : object of type 'closure' is not subsettable
```



RESULT:

Thus the new instance has been classified to a target attribute (grape fruit) using Decision Tree algorithm.

EX.NO: 7

IMPLEMENTATION OF NAÏVE BAYES CLASSIFICATION ALGORITHM

AIM:

To classify the new instance (new sample) to a target data using Naïve Bayes classification algorithm in R Studio.

PROCEDURE:

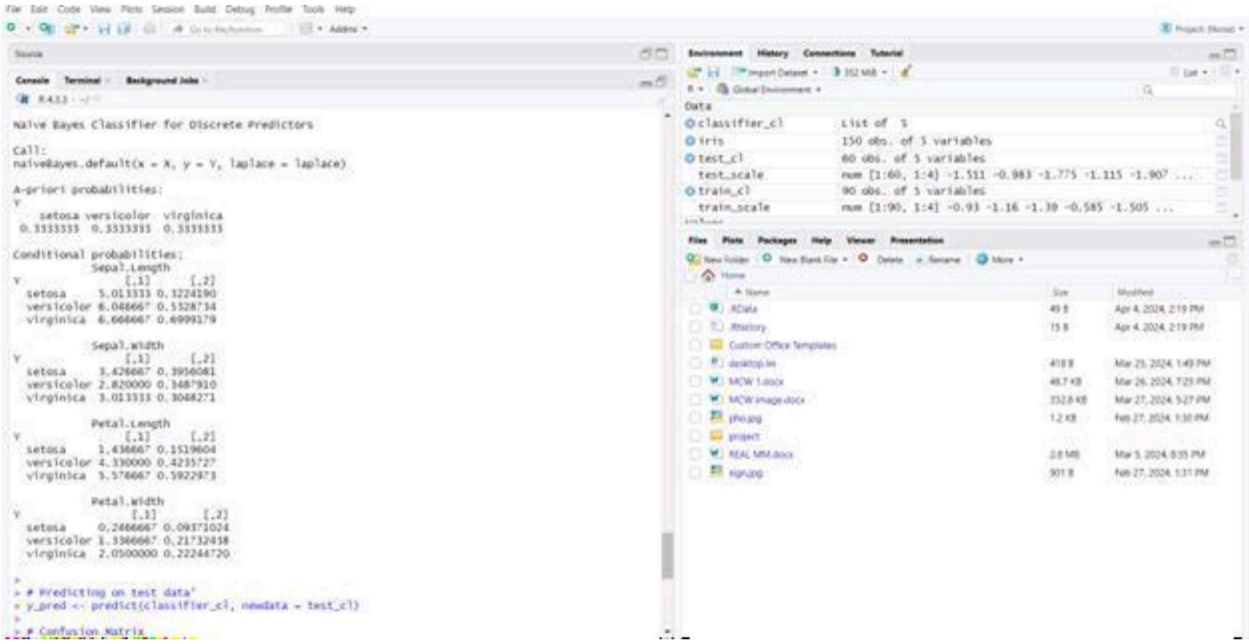
1. Install the necessary package in R Studio such as “e1071”
2. Set the current working directory
3. Save the dataset in the current working directory
4. Read the dataset in the form of csv file using **read.csv().**
5. Finally predict the category for the new instance using Naïve Bayes classifier.

SOURCE CODE:

```
# Loading package
library(e1071)
library(caTools)
library(caret)
# Splitting data into
train # and test data
split <- sample.split(iris, SplitRatio = 0.7)
train_cl <- subset(iris, split == "TRUE")
test_cl <- subset(iris, split == "FALSE")
 # Feature Scaling
train_scale <- scale(train_cl[, 1:4])
test_scale <- scale(test_cl[, 1:4])
```

```
# Fitting Naive Bayes Model

# to training dataset
set.seed(120)
# Setting Seed
classifier_cl <- naiveBayes(Species ~ ., data = train_cl) classifier_cl
# Predicting on test data'
y_pred <- predict(classifier_cl, newdata = test_cl)

# Confusion Matrix
cm <- table(test_cl$Species, y_pred) cm
# Model Evaluation
confusionMatrix(cm)
```

OUTPUT:



RESULT:

Thus, the new instance has been classified to a target attribute using Naïve Bayes classification algorithm.

EX.NO: 8

IMPLEMENTATION OF K-NN CLASSIFICATION ALGORITHM

AIM:

To classify the test data to a target attribute based on the model built using the training data with the help K-NN classification algorithm.

PROCEDURE:

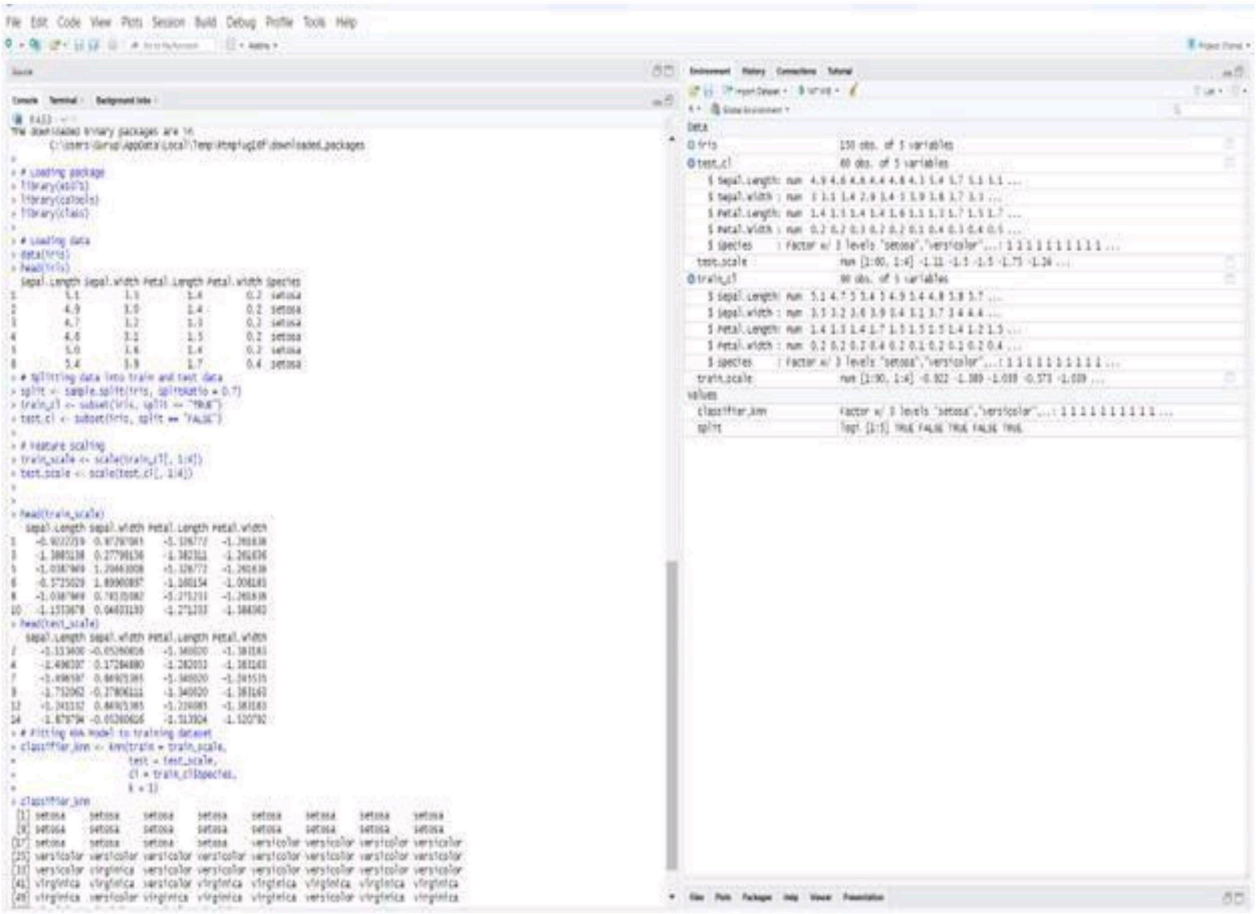
1. Install the necessary package in R Studio such as “class”
2. Set the current working directory
3. Save the dataset in the current working directory
4. Read the dataset in the form of csv file using **read.csv ()**
5. Finally create the model using kNN algorithm with the help of training data.

SOURCE CODE:

```
# Loading data
data(iris)
# Structure str(iris)
# Installing Packages
Install.Packages("e1071")
install.packages("caTools")
install.packages("class")
# Loading package
library(e1071)
library(caTools)
library(class)
```

```
# Loading data
data(iris)
head(iris)
# Splitting data into train and test data
split <- sample. split(iris, SplitRatio = 0.7)
train_cl <- subset(iris, split == "TRUE")
test_cl <- subset(iris, split == "FALSE")
# Feature Scaling
train_scale <- scale(train_cl[, 1:4])

test_scale <- scale(test_cl[, 1:4])

head(train_scale)
head(test_scale)
# Fitting KNN Model to training dataset
classifier_knn <- knn(train = train_scale,
                      test = test_scale,
                      cl = train_cl$Species, k = 1)
classifier_knn
```

OUTPUT:



RESULT:

Thus, the test data are classified successfully based on the model built using K-NN classification algorithm.

EX.NO: 9

IMPLEMENTATION OF K-MEANS CLUSTERING ALGORITHM

AIM:

To group the IRIS (flower) data points into two different clusters with the help k-means clustering algorithm.

PROCEDURE:

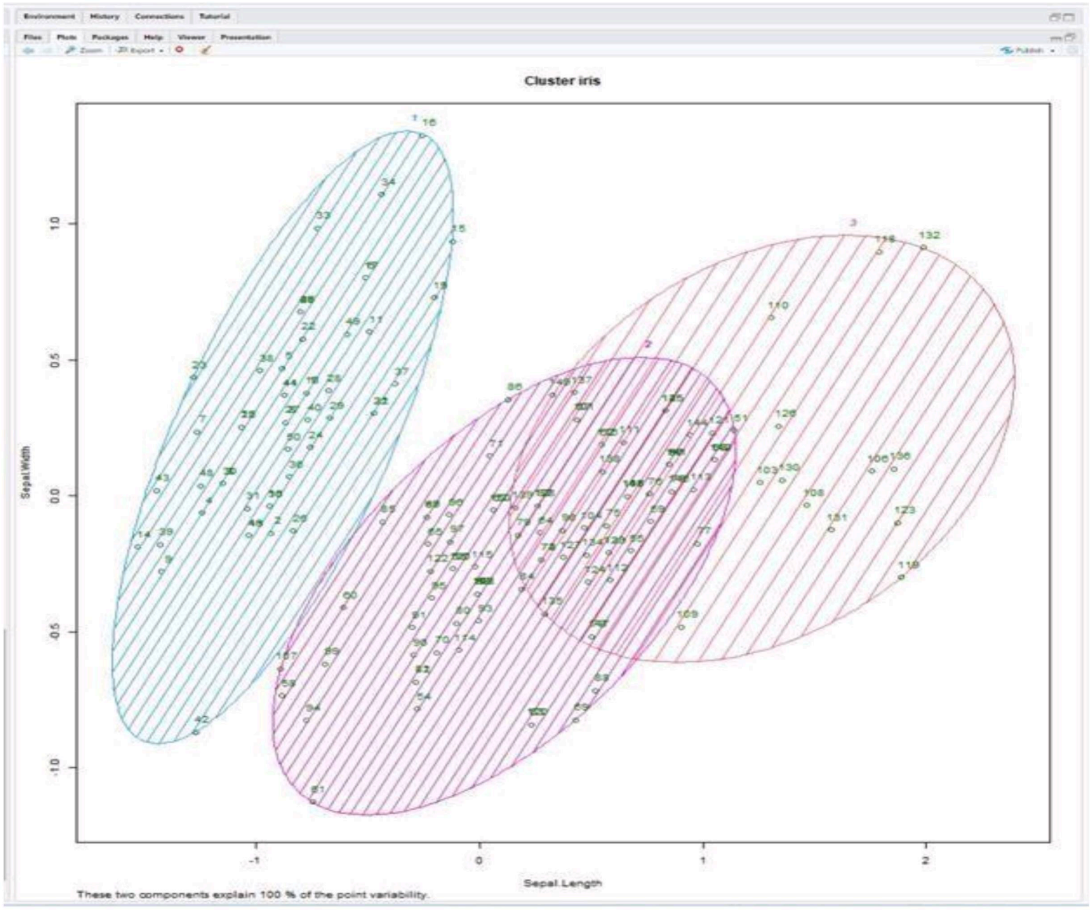
1. Install the necessary packages in R Studio such as “stats”,”dplyr”, “ggplot2”, “ggfortify”
2. Set the current working directory
3. Save the IRIS dataset in the current working directory
4. Read the dataset in the form of csv file using **read.csv ()**
5. Finally create the model (k=2) with the help of k-means clustering algorithm

SOURCE CODE:

```
# Loading data

data(iris)

# Structre

str(iris)

# Installing Packages

install.packages("Cluster")

install.  packages("cluster")

# Loading package

library(ClusterR)

library(cluster)

# Removing initial label of

# Species from original dataset
```

```
iris_1 <- iris[, -5]

# Fitting K-Means clustering Model

# to training dataset

set.seed(240) # Setting seed
kmeans.re <- kmeans(iris_1, centers = 3, nstart = 20) kmeans.re

# Cluster identification for

# each observation kmeans.re$cluster

# Confusion Matrix

cm <- table(iris$Species, kmeans.re$cluster) cm

# Model Evaluation and visualization

plot (iris_1[c("Sepal.Length", "Sepal.Width")])

plot (iris_1[c("Sepal.Length", "Sepal.Width")], col = kmeans.re$cluster)

plot(iris_1[c("Sepal.Length", "Sepal.Width")],col = kmeans.re$cluster,main = "K-means with 3 clusters")

## Plotiing cluster centers kmeans.re$centers

kmeans.re$centers[, c("Sepal.Length", "Sepal.Width")]

# cex is font size, pch is symbol

points(kmeans.re$centers[, c("Sepal.Length", "Sepal.Width")], col = 1:3, pch = 8, cex = 3)

## Visualizing clusters y_kmeans <- kmeans.re$cluster

clusplot(iris_1[, c("Sepal.Length", "Sepal.Width")], y_kmeans,

        lines =  0,

        shade = TRUE,

        color = TRUE,


        labels =  2,

        plotchar = FALSE,

        span = TRUE,

        main = paste("Cluster iris"), xlab = 'Sepal.Length', ylab = 'Sepal.Width')
```

OUTPUT:



RESULT:

Thus, the IRIS data points are successfully grouped into two different clusters with the help of kmeans clustering algorithm.

EX.NO: 10

IMPLEMENTATION OF K-MEDOIDS CLUSTERING ALGORITHM

AIM:

To group the customers of a bank based on their credit card details into three different clusters with the help k-medoids clustering algorithm.

PROCEDURE:

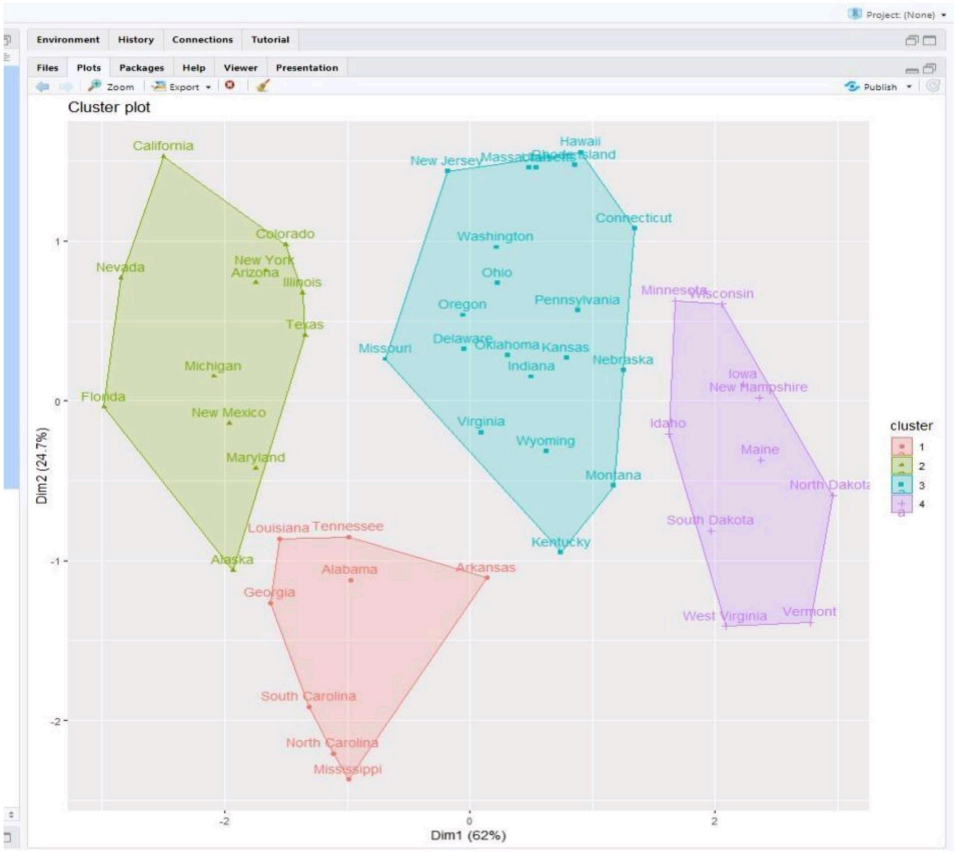
1. Install the necessary packages in R Studio such as “cluster” and “factoextra”
2. Set the current working directory
3. Save the Credit Card details dataset in the current working directory
4. Read the dataset in the form of csv file using **read.csv** ()
5. Finally create the model (k=3) with the help of k-medoids clustering algorithm

SOURCE CODE:

```
install.packages("factoextra")
library(factoextra)


#load data
df <- USArrests


#remove rows with missing
values df <- na.omit(df)


#scale each variable to have a mean of 0 and sd of
```

```
1 df <- scale(df)



#view first six rows of dataset

head(df)

fviz_nbclust(df, pam, method = "wss")

#calculate gap statistic based on number of clusters

gap_stat <- clusGap(df, FUN = pam,K.max = 10, #max clusters to consider B = 50)



 #total bootstrapped iterations #plot number of clusters vs. gap statistic

fviz_gap_stat(gap_stat)

#make this example reproducible

set.seed(1)



#perform k-medoids clustering with k = 4

clusters kmed <- pam(df, k = 4)

#view results kmed

#plot results of final k-medoids model fviz_cluster(kmed, data = df)
```

OUTPUT:



RESULT:

Thus, the three different clusters are created with the helpof k-medoids clustering algorithm.

EX.NO: 11

IMPLEMENTATION OF DATA VISUALIZATION IN R- BAR PLOT AND HISTOGRAM

AIM:

To implement the Data Visualization (Bar plot and Histogram) using R Studio.

PROCEDURE:

1. Install the necessary packages in R Studio such as “vcd”
2. Set the current working directory
3. Save the dataset in the form of csv file in the current working directory
4. Read the csv file using **read.csv ()**
5. Finally create the bar plot and histogram for the dataset.

SOURCE CODE:

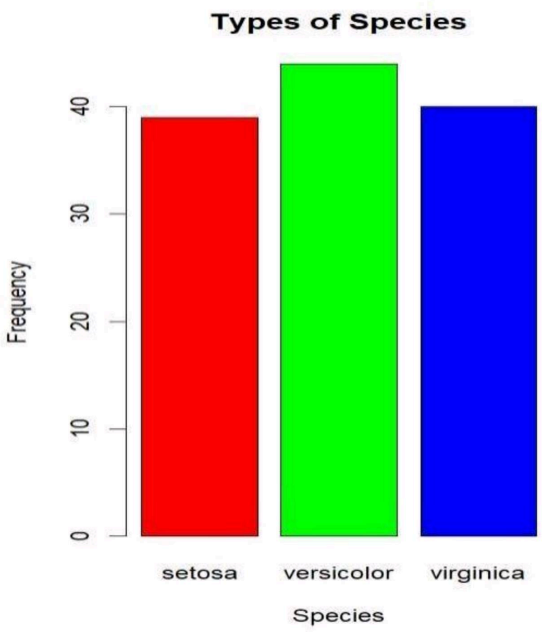
(i) Bar Plot:

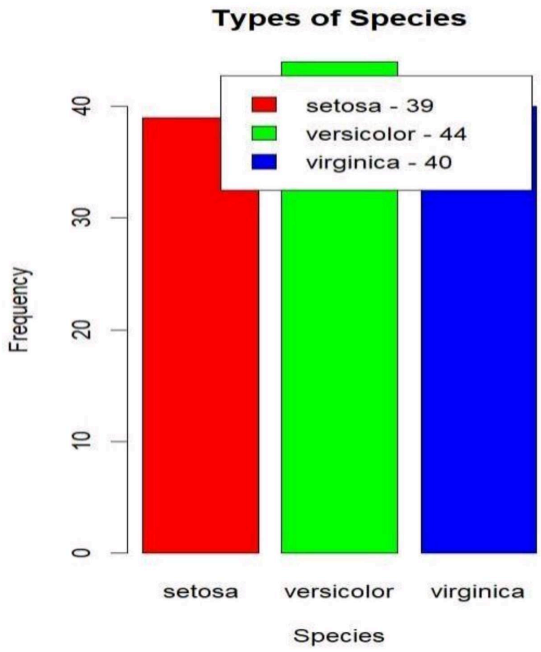
```
library(vcd)
setwd("E:\\SRMIST\\DA using R\\knn")
data=read.csv("iris.csv")
View(data) plot=table(data$Species)
barplot(plot,col=rainbow(length(plot)),xlab="Species",ylab="Frequency",main="Types of Species")
a=paste(rownames(plot),"-",plot)
barplot(plot,col=rainbow(length(plot)),xlab="Species",ylab="Frequency",main="Types of
Species",legend.text = a)
```

(ii) Histogram:

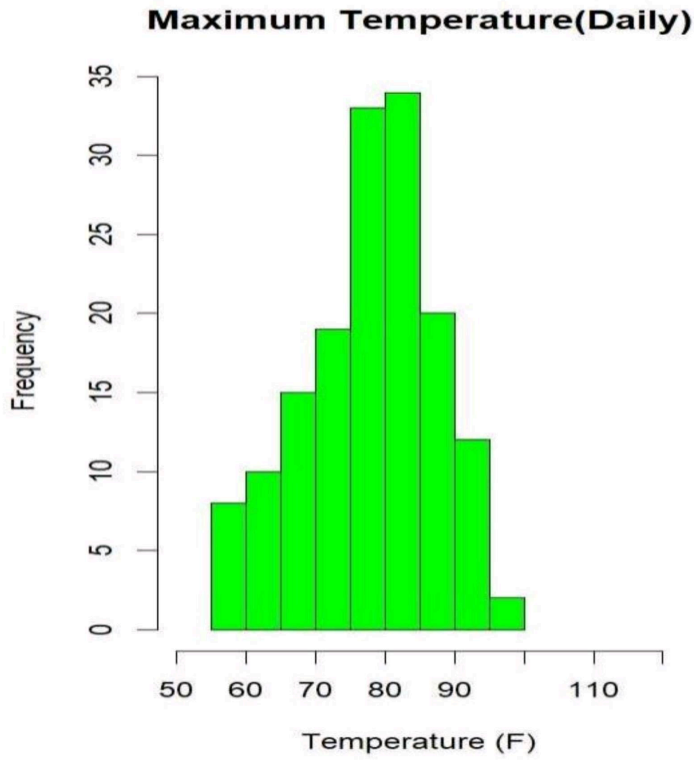
```
library(vcd)
setwd("E:\\SRMIST\\DA using R\\knn")
data=read.csv("airquality.csv") View(data)
hist(airquality$Temp,main="Maximum
Temperature(Daily)",xlab="Temperature(F)",xlim=c(50,120),col="green")
```

(i) Bar Plot:





(ii) Histogram:



RESULT:

Thus, the Bar Plot and Histogram have been successfully created in R Studio.

EX.NO: 12

IMPLEMENTATION OF DATA VISUALIZATION IN R- BOX PLOT AND SCATTER PLOT

AIM:

To implement the Data Visualization (Box plot and Scatter Plot) using R Studio.

PROCEDURE:

1. Install the necessary packages in R Studio such as “vcd”
2. Set the current working directory
3. Save the dataset in the form of csv file in the current working directory
4. Read the csv file using **read.csv ()**
5. Finally create the bar plot and histogram for the dataset.

SOURCE CODE:

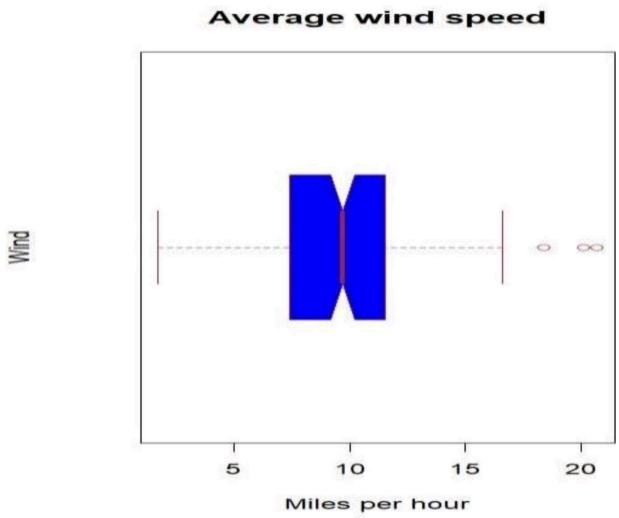
(i) Box Plot:

```
library(vcd)
library(ggplot2)
setwd("E:\\SRMIST\\DA using R\\knn")
data=read.csv("airquality.csv")
boxplot(airquality$Wind, main = "Average wind speed",xlab = "Miles per hour", ylab =
"Wind", col = "orange", border = "brown",  horizontal = TRUE, notch = TRUE)
```

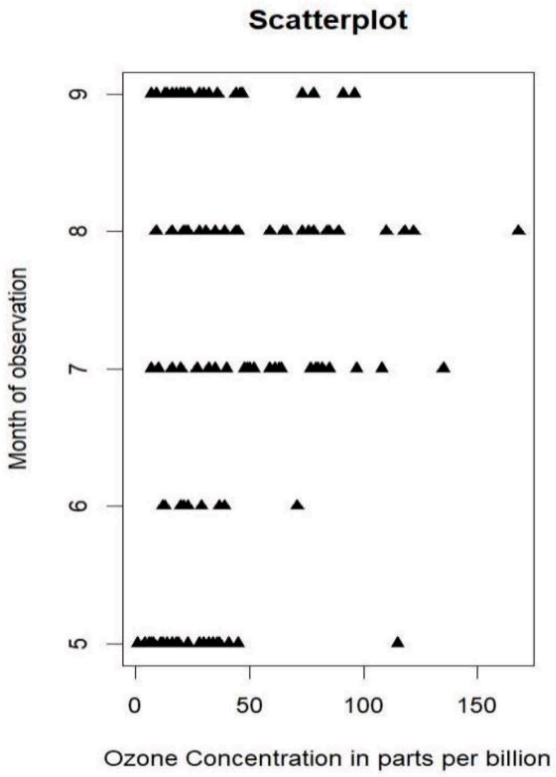
(ii) Scatter Plot:

```
library(vcd)
library(ggplot2)
setwd("E:\\SRMIST\\DA using R\\knn") data=read.csv("airquality.csv") plot(airquality$Ozone,
airquality$Month,main ="Scatterplot",xlab ="Ozone Concentration in parts per billion", ylab =" Month
of observation ", pch = 17)
```

(i) Box Plot:



(ii) Scatter Plot:



RESULT:

Thus, the Box Plot and Scatter Plot have been successfully created in R Studio.

EX.NO: 13

IMPLEMENTATION OF VARIOUS CHARTS

AIM:

To implement different types of charts in R, including bar charts, pie charts, and line charts, using built-in R functions.

PROCEDURE:

1. Install and Load Required Packages – Ensure R's graphical libraries (like ggplot2) are installed if needed.
2. Create Sample Data – Define datasets for visualization.
3. Generate Different Charts – Use functions like barplot(), pie(), and plot() to create bar, pie, and line charts.
4. Display Output – Execute the script and visualize the charts.

SOURCE CODE:

```
# Bar Chart

data <- c(20, 30, 50, 40, 60)

names <- c("A", "B", "C", "D", "E")

barplot(data, names.arg=names, col="blue", main="Bar Chart Example", xlab="Categories", ylab="Values")


# Pie Chart

values <- c(25, 35, 40)

labels <- c("Apple", "Banana", "Cherry")

pie(values, labels=labels, col=rainbow(length(values)), main="Pie Chart Example")


# Line Chart
```

```
x <- 1:10

y <- x^2

plot(x, y, type="o", col="red", main="Line Chart Example", xlab="X-axis", ylab="Y-axis")
```

OUTPUT:

1. Bar Chart – Displays a bar graph with labeled categories and corresponding values.
2. Pie Chart – Shows a pie chart with colored segments for different items.
3. Line Chart – Plots a line graph representing the relation between x and y values.

Result:

The implementation of different charts in R was successfully executed, and the required visualizations were generated.

EX.NO: 14

IMPLEMENTATION OF PREDICTIVE MODEL IN R

AIM:

To implement a predictive model in R using linear regression to forecast outcomes based on given data.

PROCEDURE:

1. Load the Dataset – Use built-in or custom datasets for predictive modeling.
2. Split Data into Training and Testing Sets – Divide the dataset for model training and evaluation.
3. Build and Train the Model – Use the lm() function for linear regression or other machine learning models.
4. Make Predictions and Evaluate Performance – Use predict() to forecast values and assess model accuracy.

SOURCE CODE:

```
# Load dataset
data <- data.frame(
  Experience = c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10),
  Salary = c(30000, 35000, 40000, 45000, 50000, 55000, 60000, 65000, 70000, 75000)
)

# Split data into training and testing sets
train_data <- data[1:8, ]  # First 8 rows for training
test_data <- data[9:10, ]  # Last 2 rows for testing

# Build Linear Regression Model
model <- lm(Salary ~ Experience, data = train_data)
```

```
# Predict salary for test data
predictions <- predict(model, newdata = test_data)


# Display results
print("Predicted Salaries:")
print(predictions)


# Plot the regression line
plot(data$Experience, data$Salary, col="blue", pch=16, main="Experience vs Salary")
abline(model, col="red", lwd=2)
```

OUTPUT:

1. Predicted Salary Values – Prints the predicted salaries for the test data.
2. Regression Line Plot – Displays a scatter plot of experience vs. salary with a fitted regression line.

Result:

The predictive model was successfully implemented in R using linear regression, and future salary predictions based on experience were generated.