DEPARTMENT OF
COMPUTER SCIENCE & ENGINEERING
Discover. Learn. Empower.

CHANDIGARH
UNIVERSITY
Discover. Learn. Empower.

UNIVERSITY INSTITUTE OF ENGINEERING

Department of Computer Science & Engineering

Subject Name: DM LAB

Subject Code: 20CSP376

Submitted to:

Faculty name

Er. Parvez Rahi

E14563

Submitted by:

Name: Vikash Yadav

UID: 21BCS8093

Section: 20BCS_DM-719

Group: B

INDEX

S.NO	PROGRAM	DATE	EVALUATION				SIGN
			LW (12)	VV (8)	FW (10)	Total (30)	
1.	Demonstration of preprocessing on .arff file using super_sleepers.arff	15/02/23					
2.	To perform the statistical analysis of data.	22/02/23					
3.	Demonstration of association rule mining using Apriori algorithm on supermarket data	01/03/23					
4.	Demonstration of FP Growth algorithm on supermarket data.	15/03/23					
5.	To perform the classification by decision tree induction using WEKA tools.	05/04/23					
6.	To perform classification using Bayesian classification algorithm using R.	12/04/23					
7.	To perform the cluster analysis by k-means method using R	19/04/23					
8.	To perform hierarchical clustering using R programming	26/04/23					
9.	Study of Regression Analysis using R programming	03/05/23					
10.	Outlier detection using R programming	10/05/23					

EXPERIMENT 1.1

Student Name: Vikash Yadav	**UID: 21BCS8093**
Branch: CSE	**Section/Group: DM_719/B**
Semester: 6th	**Date of Performance: 15/02/23**
Subject Name: Data Mining	**Subject Code: 20CSP-376**

1. Aim:

Demonstration of preprocessing on .arff file using super_sleepers.arff.

2. Code:

i)

```
library(RWeka)

setwd("C:\\Users\\CU\\Downloads")

getwd()

rating <- 1:4
animal <- c('koala', 'hedgehog', 'sloth', 'panda')
country <- c('Australia', 'Italy', 'Peru', 'China')
avg_sleep_hours <- c(21, 18, 17, 10)
```

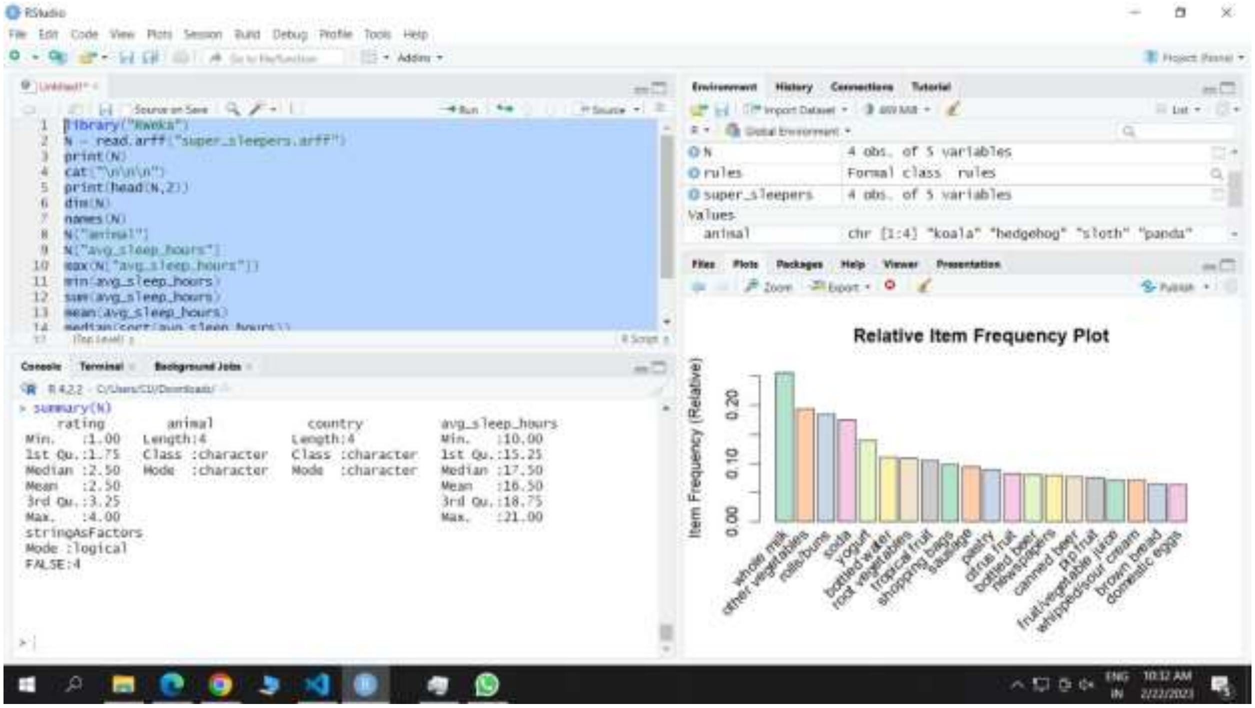
```
super_sleepers <- data.frame(rating, animal, country, avg_sleep_hours,
stringAsFactors=FALSE)
print(super_sleepers)

print(class(super_sleepers))

print(str(super_sleepers))

write.arff(super_sleepers, file="super_sleepers.arff")
```

ii)

```
library("RWeka")
N = read.arff("super_sleepers.arff")
print(N)
cat("\n\n\n")
print(head(N,2))
dim(N)
names(N)
N["animal"]
N["avg_sleep_hours"]
max(N["avg_sleep_hours"])
min(avg_sleep_hours)
sum(avg_sleep_hours)
mean(avg_sleep_hours)
median(sort(avg_sleep_hours))
sd(avg_sleep_hours)
```

summary(N)

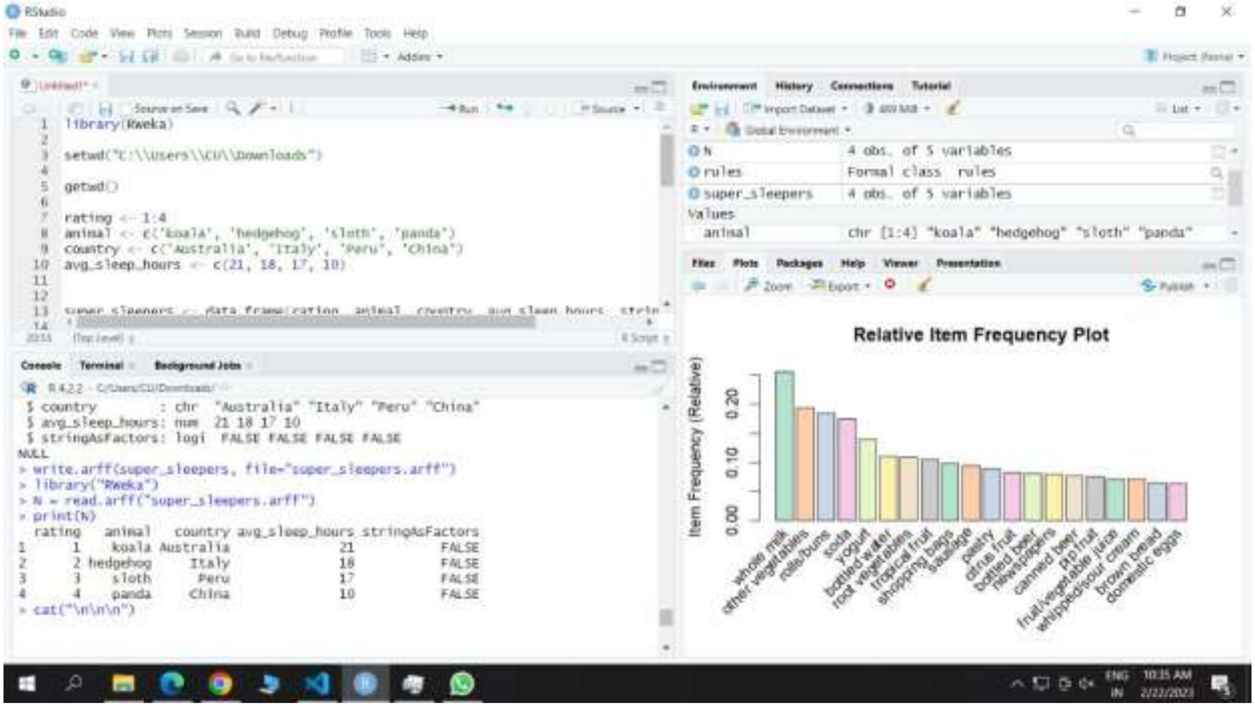
3. Output:



```
library("Rweka")
N <- read.arff("super_sleepers.arff")
print(N)
cat("\n\n\n")
print(head(N,2))
dim(N)
names(N)
N["animal"]
N["avg_sleep_hours"]
max(N["avg_sleep_hours"])
min(avg_sleep_hours)
sum(avg_sleep_hours)
mean(avg_sleep_hours)
median(sort(avg_sleep_hours))
```

```
> summary(N)
     rating        animal              country            avg_sleep_hours
 Min.   :1.00   Length:4           Length:4           Min.   :10.00
 1st Qu.:1.75   Class :character   Class :character   1st Qu.:15.25
 Median :2.50   Mode  :character   Mode  :character   Median :17.50
 Mean   :2.50                                         Mean   :16.50
 3rd Qu.:3.25                                         3rd Qu.:18.75
 Max.   :4.00                                         Max.   :21.00
 stringAsFactors
 Mode :logical
 FALSE:4
```

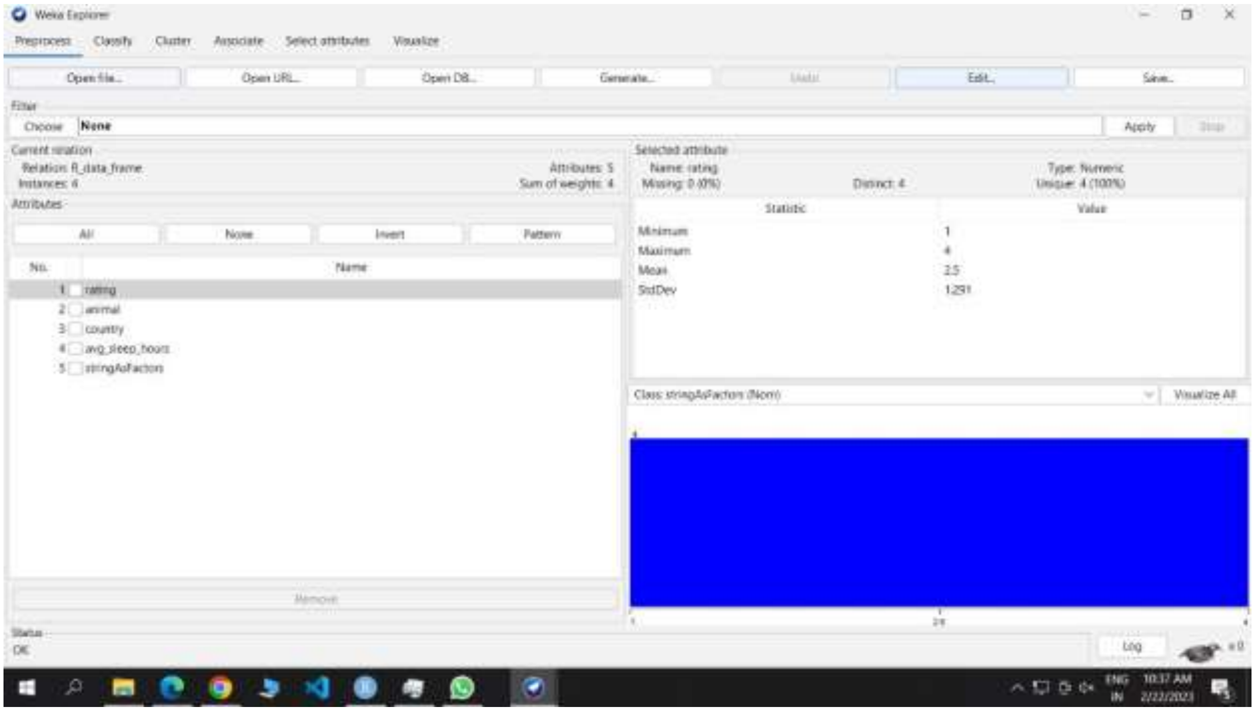
```
N                4 obs. of 5 variables
rules            Formal class  rules
super_sleepers   4 obs. of 5 variables
Values
  animal         chr [1:4] "koala" "hedgehog" "sloth" "panda"
```



```
library(Rweka)

setwd("C:\\Users\\CU\\Downloads")

getwd()

rating <- 1:4
animal <- c('koala', 'hedgehog', 'sloth', 'panda')
country <- c('Australia', 'Italy', 'Peru', 'China')
avg_sleep_hours <- c(21, 18, 17, 10)


super_sleepers <- data.frame(rating, animal, country, avg_sleep_hours, strin
```

```
 $ country        : chr  "Australia" "Italy" "Peru" "China"
 $ avg_sleep_hours: num  21 18 17 10
 $ stringAsFactors: logi  FALSE FALSE FALSE FALSE
NULL
> write.arff(super_sleepers, file="super_sleepers.arff")
> library("Rweka")
> N = read.arff("super_sleepers.arff")
> print(N)
  rating   animal   country avg_sleep_hours stringAsFactors
1      1    koala Australia              21           FALSE
2      2 hedgehog     Italy              18           FALSE
3      3    sloth      Peru              17           FALSE
4      4    panda     China              10           FALSE
> cat("\n\n\n")
```

```
N               4 obs. of 5 variables
rules           Formal class  rules
super_sleepers  4 obs. of 5 variables
Values
  animal        chr [1:4] "koala" "hedgehog" "sloth" "panda"
```



Current relation
Relation: R_data_frame
Instances: 4
Attributes: 5
Sum of weights: 4

Attributes

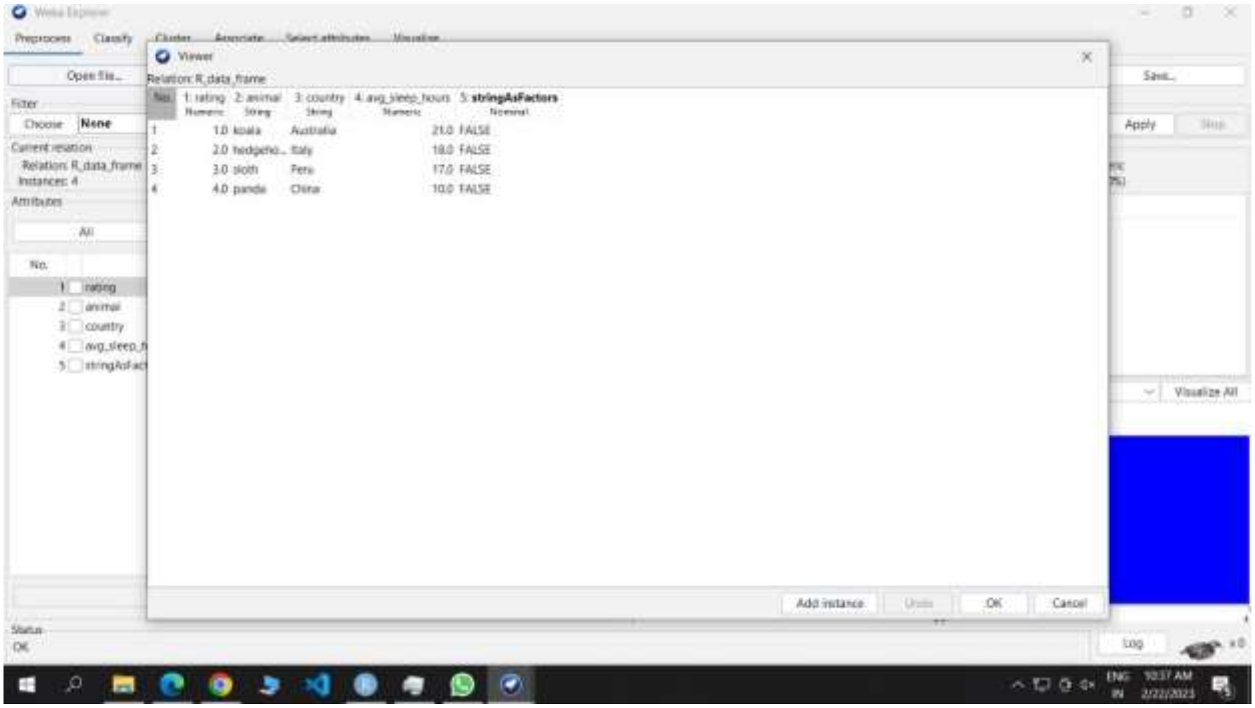
No.	Name
1	rating
2	animal
3	country
4	avg_sleep_hours
5	stringAsFactors

Selected attribute
Name: rating
Missing: 0 (0%)
Distinct: 4
Type: Numeric
Unique: 4 (100%)

Statistic	Value
Minimum	1
Maximum	4
Mean	2.5
StdDev	1.291

Class: stringAsFactors (Nom)

Status
OK



Relation: R_data_frame

No.	1: rating Numeric	2: animal String	3: country String	4: avg_sleep_hours Numeric	5: **stringAsFactors** Nominal
1	1.0	koala	Australia	21.0	FALSE
2	2.0	hedgeho...	Italy	18.0	FALSE
3	3.0	sloth	Peru	17.0	FALSE
4	4.0	panda	China	10.0	FALSE

EXPERIMENT 1.2

Student Name: Vikash Yadav **UID: 21BCS8093**
Branch: CSE **Section/Group: DM_719/B**
Semester: 6th **Date of Performance: 22/02/23**
Subject Name: Data Mining **Subject Code: 20CSP-376**

1. Aim:

To perform the statistical analysis of data.

2. Code:

i)

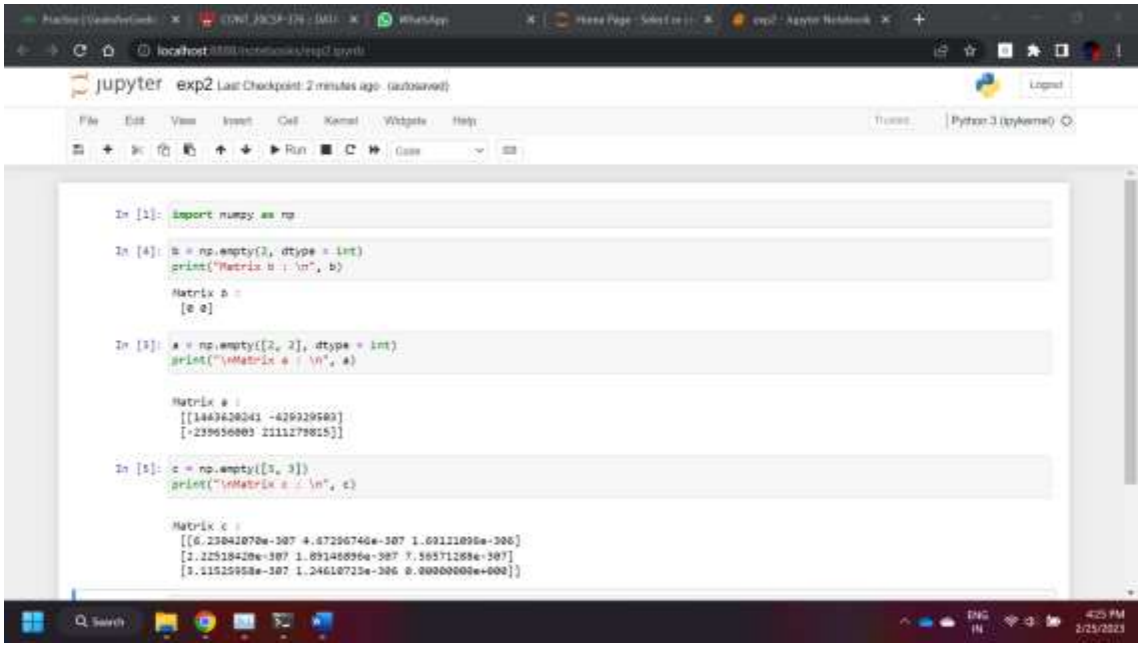
```
import numpy as np

b = np.empty(2, dtype = int)
print("Matrix b : \n", b)

a = np.empty([2, 2], dtype = int)
print("\nMatrix a : \n", a)

c = np.empty([3, 3])
print("\nMatrix c : \n", c)
```

ii)

```
import numpy as np

b = np.zeros(2, dtype = int)
print("Matrix b : \n", b)

a = np.zeros([2, 2], dtype = int)
print("\nMatrix a : \n", a)

c = np.zeros([3, 3])
print("\nMatrix c : \n", c)
```

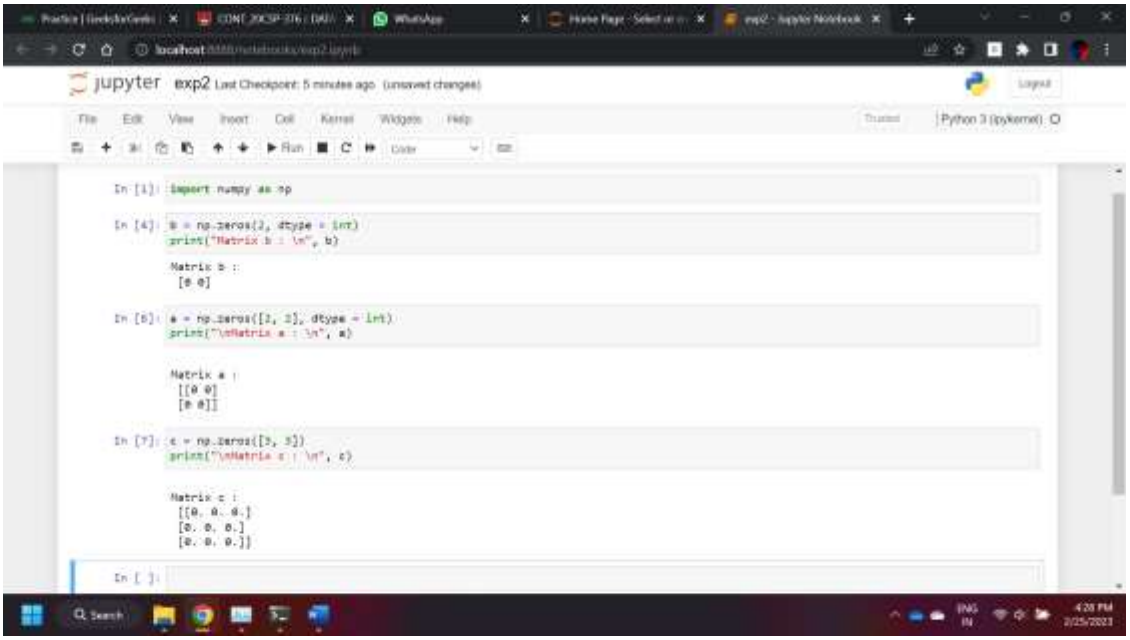
3. Output:

i)



```
In [1]: import numpy as np

In [4]: b = np.empty(2, dtype = int)
        print("Matrix b : \n", b)

        Matrix b :
         [0 0]

In [3]: a = np.empty([2, 2], dtype = int)
        print("\nMatrix a : \n", a)

        Matrix a :
         [[1443620241 -429329583]
         [-239656003 2111279815]]

In [5]: c = np.empty([3, 3])
        print("\nMatrix c : \n", c)

        Matrix c :
         [[6.23042070e-307 4.67296746e-307 1.69121096e-306]
         [2.22518420e-307 1.89146896e-307 7.56571268e-307]
         [3.11525958e-307 1.24610723e-306 0.00000000e+000]]
```

ii)



```
In [1]: import numpy as np

In [4]: b = np.zeros(2, dtype = int)
        print("Matrix b : \n", b)

        Matrix b :
         [0 0]

In [6]: a = np.zeros([2, 2], dtype = int)
        print("\nMatrix a : \n", a)

        Matrix a :
         [[0 0]
         [0 0]]

In [7]: c = np.zeros([3, 3])
        print("\nMatrix c : \n", c)

        Matrix c :
         [[0. 0. 0.]
         [0. 0. 0.]
         [0. 0. 0.]]

In [ ]:
```

Experiment-1.3

Student Name: Vikash Yadav **UID: 21BCS8093**
Branch: CSE **Section/Group: DM_719/B**
Semester: 6th **Date of Performance: 01/03/23**
Subject Name: Data Mining **Subject Code: 20CSP-376**

1. Aim:

Demonstration of association rule mining using Apriori algorithm on supermarket data.

2. Objective:

- I have implement the association rule on given data via apriori algorithm.
- Association rule mining finds interesting associations and relationships among large sets of data items.
- This rule shows how frequently a itemset occurs in a transaction.
- In this experiment I have learn to create plot and how to use different pacakges libraries.

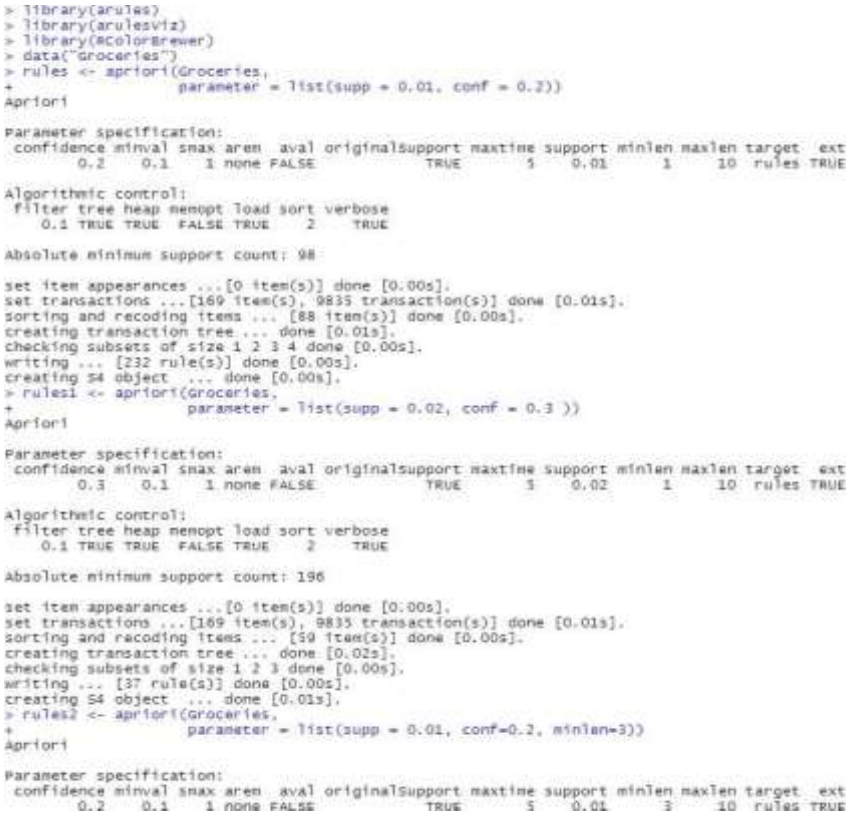
3- Script and Output:

```
    #performing association rule using apriori algo

    library(arules)
    library(arulesViz)
    library(RColorBrewer)
data("Groceries")

    rules <- apriori(Groceries, parameter = list(supp = 0.01, conf = 0.2)) rules1 <-
    apriori(Groceries, parameter = list(supp = 0.02, conf = 0.3 )) rules2 <-
    apriori(Groceries, parameter = list(supp = 0.01, conf=0.2, minlen=3))
    plot(rules) plot(rules1)
```

```
inspect(rules[1:10])

inspect(rules2[1:5])

plot(rules2)

arules::itemFrequencyPlot(Groceries, topN = 20,
                          col = brewer.pal(8, 'Pastel2'), main =
                          'Relative Item Frequency Plot', type =
                          "relative",
                        ylab = "Item Frequency (Relative)")
```

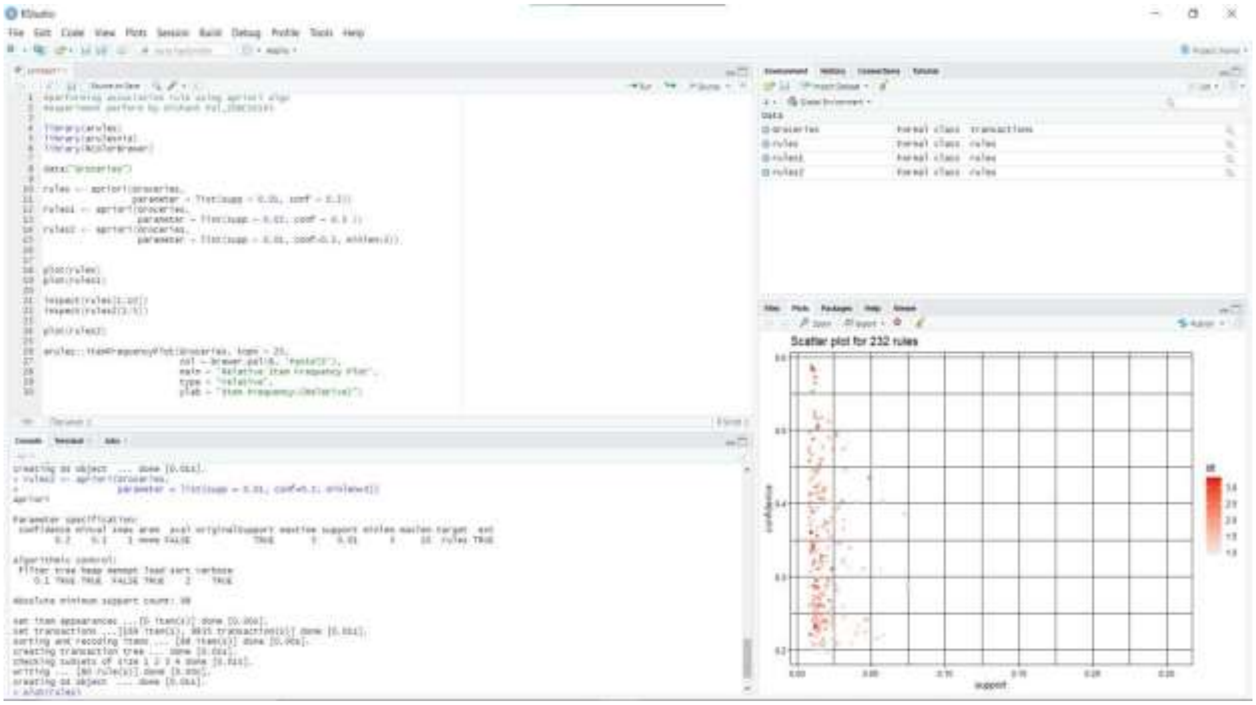
4. Output-

- **Output on R console-:**



```
> library(arules)
> library(arulesViz)
> library(RColorBrewer)
> data("Groceries")
> rules <- apriori(Groceries,
+                  parameter = list(supp = 0.01, conf = 0.2))
Apriori

Parameter specification:
 confidence minval smax arem  aval originalSupport maxtime support minlen maxlen target  ext
        0.2    0.1    1 none FALSE            TRUE       5    0.01      1     10  rules TRUE

Algorithmic control:
 filter tree heap memopt load sort verbose
    0.1 TRUE TRUE  FALSE TRUE    2    TRUE

Absolute minimum support count: 98

set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[169 item(s), 9835 transaction(s)] done [0.01s].
sorting and recoding items ... [88 item(s)] done [0.00s].
creating transaction tree ... done [0.01s].
checking subsets of size 1 2 3 4 done [0.00s].
writing ... [232 rule(s)] done [0.00s].
creating S4 object  ... done [0.00s].
> rules1 <- apriori(Groceries,
+                   parameter = list(supp = 0.02, conf = 0.3 ))
Apriori

Parameter specification:
 confidence minval smax arem  aval originalSupport maxtime support minlen maxlen target  ext
        0.3    0.1    1 none FALSE            TRUE       5    0.02      1     10  rules TRUE

Algorithmic control:
 filter tree heap memopt load sort verbose
    0.1 TRUE TRUE  FALSE TRUE    2    TRUE

Absolute minimum support count: 196

set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[169 item(s), 9835 transaction(s)] done [0.01s].
sorting and recoding items ... [59 item(s)] done [0.00s].
creating transaction tree ... done [0.02s].
checking subsets of size 1 2 3 done [0.00s].
writing ... [37 rule(s)] done [0.00s].
creating S4 object  ... done [0.01s].
> rules2 <- apriori(Groceries,
+                   parameter = list(supp = 0.01, conf=0.2, minlen=3))
Apriori

Parameter specification:
 confidence minval smax arem  aval originalSupport maxtime support minlen maxlen target  ext
        0.2    0.1    1 none FALSE            TRUE       5    0.01      3     10  rules TRUE
```

- **Scatter Plot Output-**

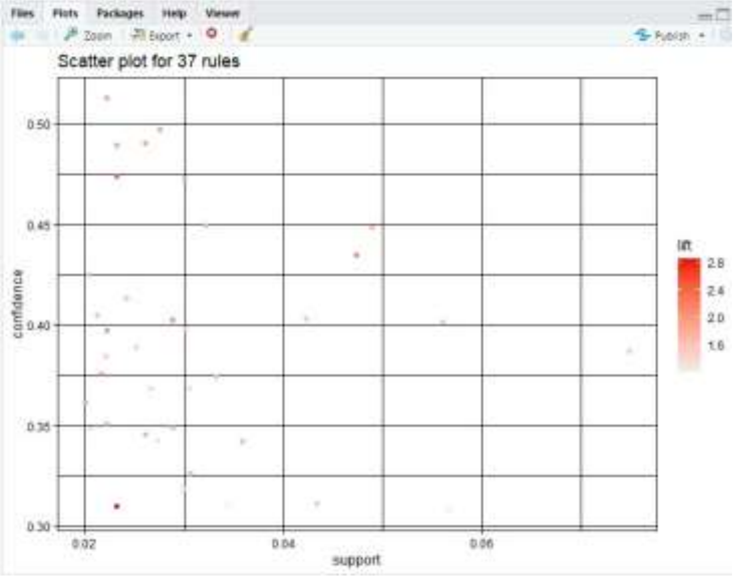

RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Environment History Connections Tutorial
Files Plots Packages Help Viewer
Zoom Export Publish
Scatter plot for 232 rules
confidence
support
lift

```
#performing association rule using apriori algo
#experiment perform by [illegible]

library(arules)
library(arulesViz)
library(RColorBrewer)

data("Groceries")

rules <- apriori(Groceries,
                 parameter = list(supp = 0.01, conf = 0.2))
rules1 <- apriori(Groceries,
                  parameter = list(supp = 0.01, conf = 0.3 ))
rules2 <- apriori(Groceries,
                  parameter = list(supp = 0.01, conf=0.3, minlen=3))


plot(rules)
plot(rules1)

inspect(rules[1:10])
inspect(rules2[1:5])

plot(rules2)

arules::itemFrequencyPlot(Groceries, topN = 20,
                          col = brewer.pal(8, 'Pastel2'),
                          main = 'Relative Item Frequency Plot',
                          type = "relative",
                          ylab = "Item Frequency (Relative)")
```

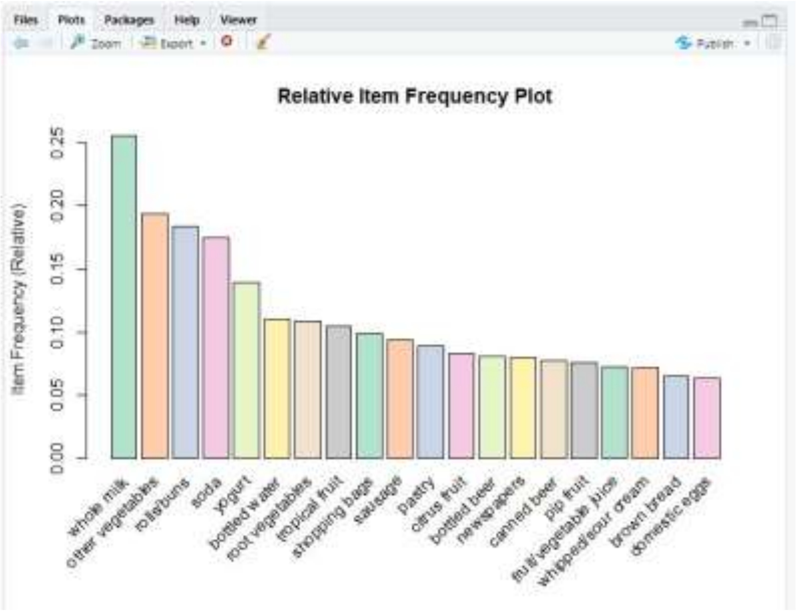
```
creating S4 object  ... done [0.01s].
> rules2 <- apriori(Groceries,
+                   parameter = list(supp = 0.01, conf=0.2, minlen=3))
Apriori

Parameter specification:
 confidence minval smax arem  aval originalSupport maxtime support minlen maxlen target  ext
        0.2    0.1    1 none FALSE            TRUE       5    0.01      3     10  rules TRUE

Algorithmic control:
 filter tree heap memopt load sort verbose
    0.1 TRUE TRUE  FALSE TRUE    2    TRUE

Absolute minimum support count: 98

set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[169 item(s), 9835 transaction(s)] done [0.01s].
sorting and recoding items ... [88 item(s)] done [0.00s].
creating transaction tree ... done [0.00s].
checking subsets of size 1 2 3 4 done [0.01s].
writing ... [60 rule(s)] done [0.00s].
creating S4 object  ... done [0.00s].
> plot(rules2)
```

```
Data
groceries    Formal class  transactions
rules        Formal class  rules
rules1       Formal class  rules
rules2       Formal class  rules
```

- Output of Confidence v/s Support-



- Output of Relative item Frequency Plot



Learning Outcomes-

1. Learned how to use of arules, arulesViz and RcolorBewer libraries in data mining.
2. Learned how to create scatter plots on given data.
3. Learned how to implement association rule using Apriori algorithm.

Experiment 1.4

Student Name: Vikash Yadav **UID: 21BCS8093**
Branch: BE-CSE **Section/Group:** DM_719/B
Semester: 6th **Subject Code:** 20CSP-376

Aim:

Demonstration of FP Growth algorithm on supermarket data.

Objective:

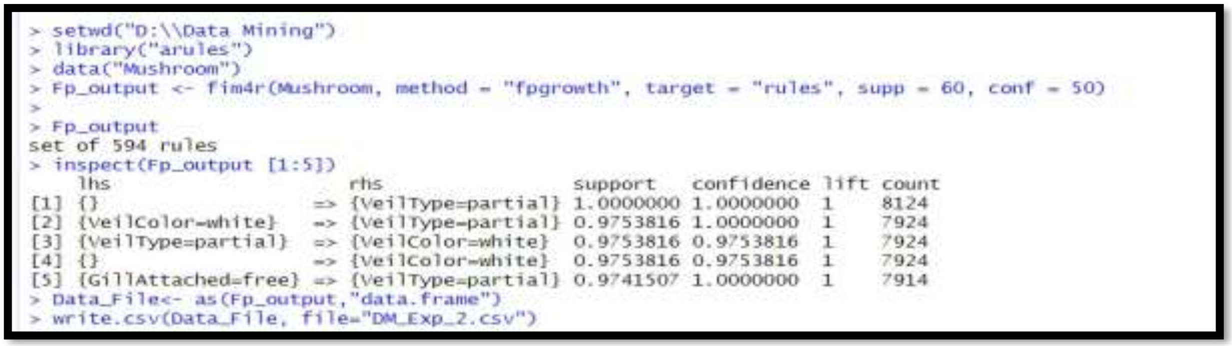
Association rule mining finds interesting associations and relationships among large sets of data items. This rule shows how frequently a itemset occurs in a transaction. Given a set of transactions, we can find rules that will predict the occurrence of an item based on the occurrences of other items in the transaction.

Code and Output:

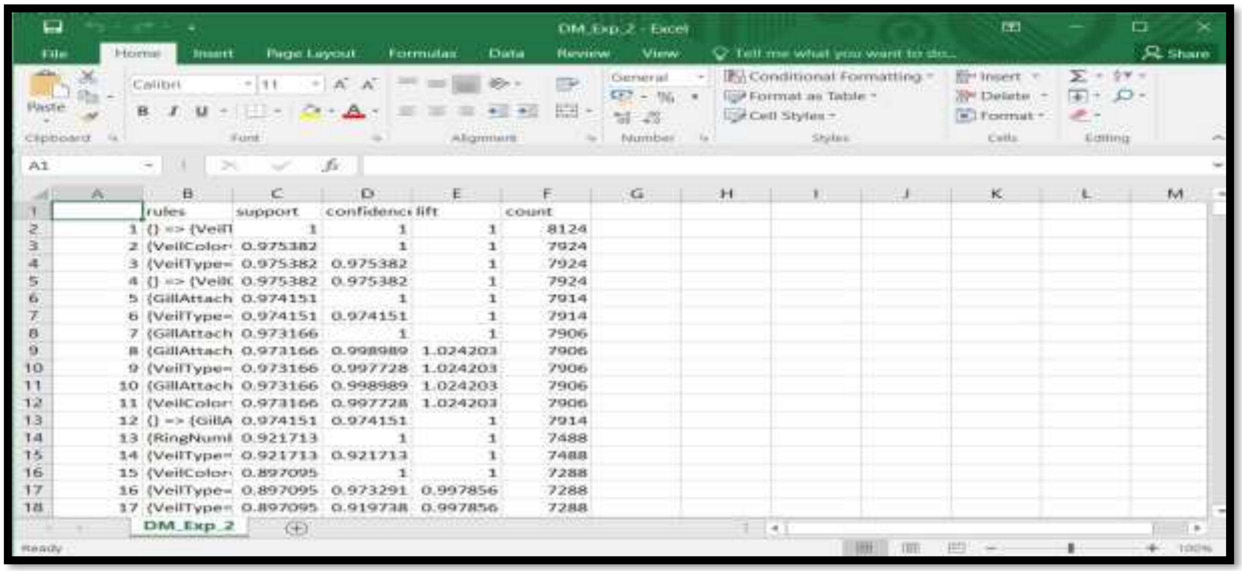
Creating Records
```
setwd("D:\\
Data Mining")
library("arules")
data("Mushroom")
Fp_output <- fim4r(Mushroom, method = "fpgrowth", target = "rules", supp = 60, conf = 50)
```

Applying Operation
```
Fp_output
inspect(Fp_output [1:5])
Data_File<- as(Fp_output,"data.frame") write.csv(Data_File,
file="DM_Exp_4.csv")
```

OUTPUT:



```
> setwd("D:\\Data Mining")
> library("arules")
> data("Mushroom")
> Fp_output <- fim4r(Mushroom, method = "fpgrowth", target = "rules", supp = 60, conf = 50)
>
> Fp_output
set of 594 rules
> inspect(Fp_output [1:5])
    lhs                        rhs                  support   confidence lift count
[1] {}                      => {VeilType=partial} 1.0000000 1.0000000  1    8124
[2] {VeilColor=white}       => {VeilType=partial} 0.9753816 1.0000000  1    7924
[3] {VeilType=partial}      => {VeilColor=white}  0.9753816 0.9753816  1    7924
[4] {}                      => {VeilColor=white}  0.9753816 0.9753816  1    7924
[5] {GillAttached=free}     => {VeilType=partial} 0.9741507 1.0000000  1    7914
> Data_File<- as(Fp_output,"data.frame")
> write.csv(Data_File, file="DM_Exp_2.csv")
```



	A	B	C	D	E	F
1		rules	support	confidence	lift	count
2	1	{} => {VeilT	1	1	1	8124
3	2	{VeilColor	0.975382	1	1	7924
4	3	{VeilType=	0.975382	0.975382	1	7924
5	4	{} => {VeilC	0.975382	0.975382	1	7924
6	5	{GillAttach	0.974151	1	1	7914
7	6	{VeilType=	0.974151	0.974151	1	7914
8	7	{GillAttach	0.973166	1	1	7906
9	8	{GillAttach	0.973166	0.998989	1.024203	7906
10	9	{VeilType=	0.973166	0.997728	1.024203	7906
11	10	{GillAttach	0.973166	0.998989	1.024203	7906
12	11	{VeilColor	0.973166	0.997728	1.024203	7906
13	12	{} => {GillA	0.974151	0.974151	1	7914
14	13	{RingNum	0.921713	1	1	7488
15	14	{VeilType=	0.921713	0.921713	1	7488
16	15	{VeilColor	0.897095	1	1	7288
17	16	{VeilType=	0.897095	0.973291	0.997856	7288
18	17	{VeilType=	0.897095	0.919738	0.997856	7288

Observations & Conclusion:

The "fim4r" function is used to mine frequent itemsets and generate association rules using the "fpgrowth" method with a minimum support of 60% and minimum confidence of 50%. The output of the function is stored in the "Fp_output" variable, which is then inspected using the "inspect" function to display the first five association rules.

Learning outcomes (What I have learnt):

1. Association rule mining: Students can learn how to use different methods, such as Apriori or FP-Growth, to mine frequent itemsets and generate association rules.
2. Minimum support and confidence: The code uses the minimum support and minimum confidence parameters to filter out weak rules and ensure that only meaningful rules

Experiment-2.1

Student Name: Vikash Yadav **UID: 21BCS8093**
Branch: BE-CSE **Section/Group: 719/B**
Semester: 6th **Date of Performance: 05/04/2023**
Subject Name: Data Mining Lab **Subject Code: 20CSP-376**

1. Aim:

To perform the classification by decision tree induction using WEKA tools.

2. Objective:

- The objective is to identify the most important predictor variables for a given outcome.
- To create a visual representation of the decision-making process for a particular problem.
- To classify or predict outcomes based on a set of input variables.
- To determine the optimal decision path based on the expected value of outcomes.

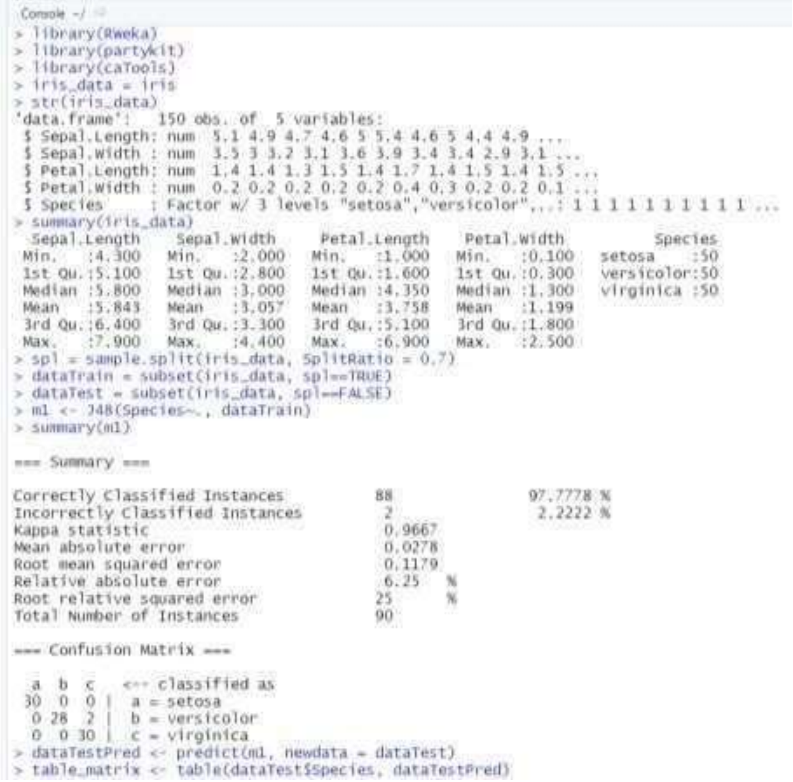
3. Code and Output:

- **PROGRAM**

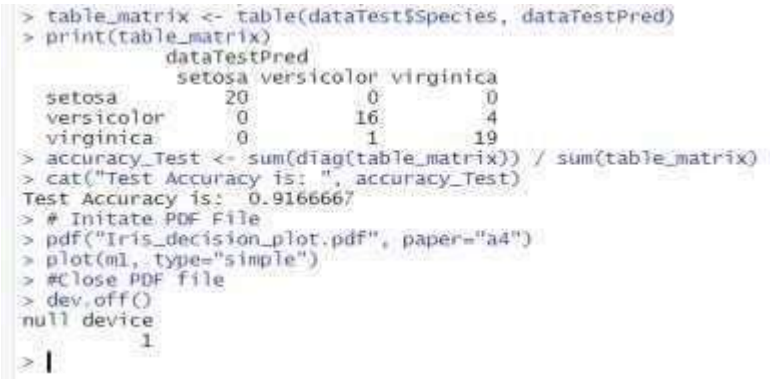
```
library(RWeka) library(partykit)
library(caTools)
 iris_data =
iris


str(iris_data) summary(iris_data) spl =


sample.split(iris_data, SplitRatio = 0.7)



dataTrain = subset(iris_data, spl==TRUE)
dataTest = subset(iris_data, spl==FALSE)


m1 <- J48(Species~., dataTrain) summary(m1)
dataTestPred <- predict(m1, newdata = dataTest)
table_matrix <- table(dataTest$Species,
dataTestPred)
```

```
print(table_matrix) accuracy_Test <-
 sum(diag(table_matrix)) /
 sum(table_matrix)
  cat("Test Accuracy is: ",
 accuracy_Test)
 # Initate PDF File
 pdf("Iris_decision_plot.pdf", paper="a4")
 plot(m1,
 type="simple")
 #Close PDF file
 dev.off() •
```

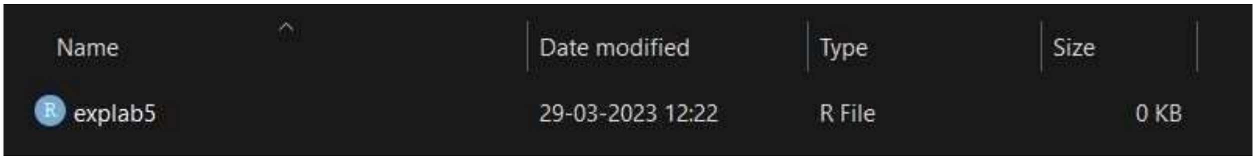
OUTPUT



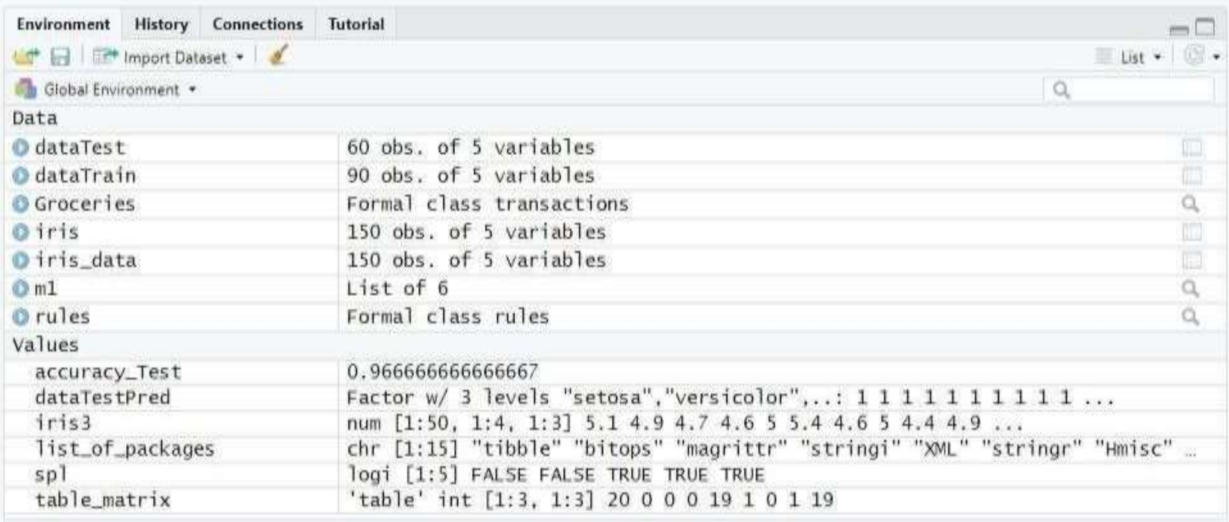
```
> library(RWeka)
> library(partykit)
> library(caTools)
> iris_data = iris
> str(iris_data)
'data.frame':   150 obs. of  5 variables:
 $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
 $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
 $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
 $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
 $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
> summary(iris_data)
  Sepal.Length    Sepal.Width     Petal.Length    Petal.Width          Species  
 Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100   setosa    :50  
 1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300   versicolor:50  
 Median :5.800   Median :3.000   Median :4.350   Median :1.300   virginica :50  
 Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199                  
 3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800                  
 Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500                  
> spl = sample.split(iris_data, SplitRatio = 0.7)
> dataTrain = subset(iris_data, spl==TRUE)
> dataTest = subset(iris_data, spl==FALSE)
> m1 <- J48(Species~., dataTrain)
> summary(m1)

=== Summary ===

Correctly Classified Instances          88               97.7778 %
Incorrectly Classified Instances         2                2.2222 %
Kappa statistic                          0.9667
Mean absolute error                      0.0278
Root mean squared error                  0.1179
Relative absolute error                  6.25   %
Root relative squared error             25      %
Total Number of Instances               90     

=== Confusion Matrix ===

  a  b  c   <-- classified as
 30  0  0 |  a = setosa
  0 28  2 |  b = versicolor
  0  0 30 |  c = virginica
> dataTestPred <- predict(m1, newdata = dataTest)
> table_matrix <- table(dataTest$Species, dataTestPred)
```



```
> table_matrix <- table(dataTest$Species, dataTestPred)
> print(table_matrix)
            dataTestPred
             setosa versicolor virginica
  setosa         20          0         0
  versicolor      0         16         4
  virginica       0          1        19
> accuracy_Test <- sum(diag(table_matrix)) / sum(table_matrix)
> cat("Test Accuracy is: ", accuracy_Test)
Test Accuracy is:  0.9166667
> # Initate PDF File
> pdf("Iris_decision_plot.pdf", paper="a4")
> plot(m1, type="simple")
> #Close PDF file
> dev.off()
null device
          1
> 
```

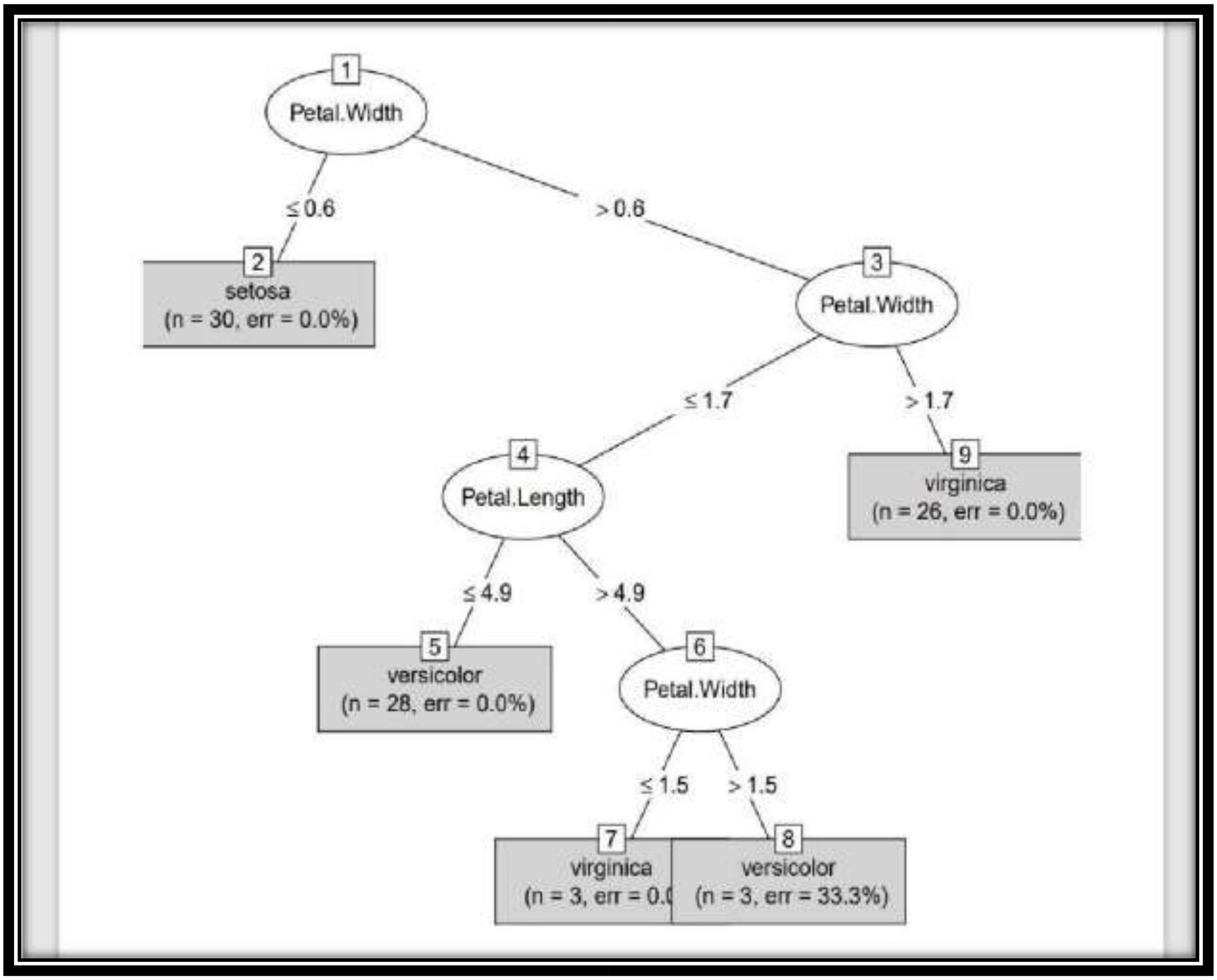


Name	Date modified	Type	Size
explab5	29-03-2023 12:22	R File	0 KB



Data	
dataTest	60 obs. of 5 variables
dataTrain	90 obs. of 5 variables
Groceries	Formal class transactions
iris	150 obs. of 5 variables
iris_data	150 obs. of 5 variables
m1	List of 6
rules	Formal class rules
Values	
accuracy_Test	0.966666666666667
dataTestPred	Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
iris3	num [1:50, 1:4, 1:3] 5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
list_of_packages	chr [1:15] "tibble" "bitops" "magrittr" "stringi" "XML" "stringr" "Hmisc" ...
spl	logi [1:5] FALSE FALSE TRUE TRUE TRUE
table_matrix	'table' int [1:3, 1:3] 20 0 0 0 19 1 0 1 19

Final Output and Decision Tree



Experiment-2.2

Student Name: Vikash Yadav **UID: 21BCS8093**
Branch: BE-CSE **Section/Group: 719/B**
Semester: 6th **Date of Performance: 12/04/2023**
Subject Name: Data Mining Lab **Subject Code: 20CSP-376**

Aim: To perform classification using Bayesian classification algorithm using R.

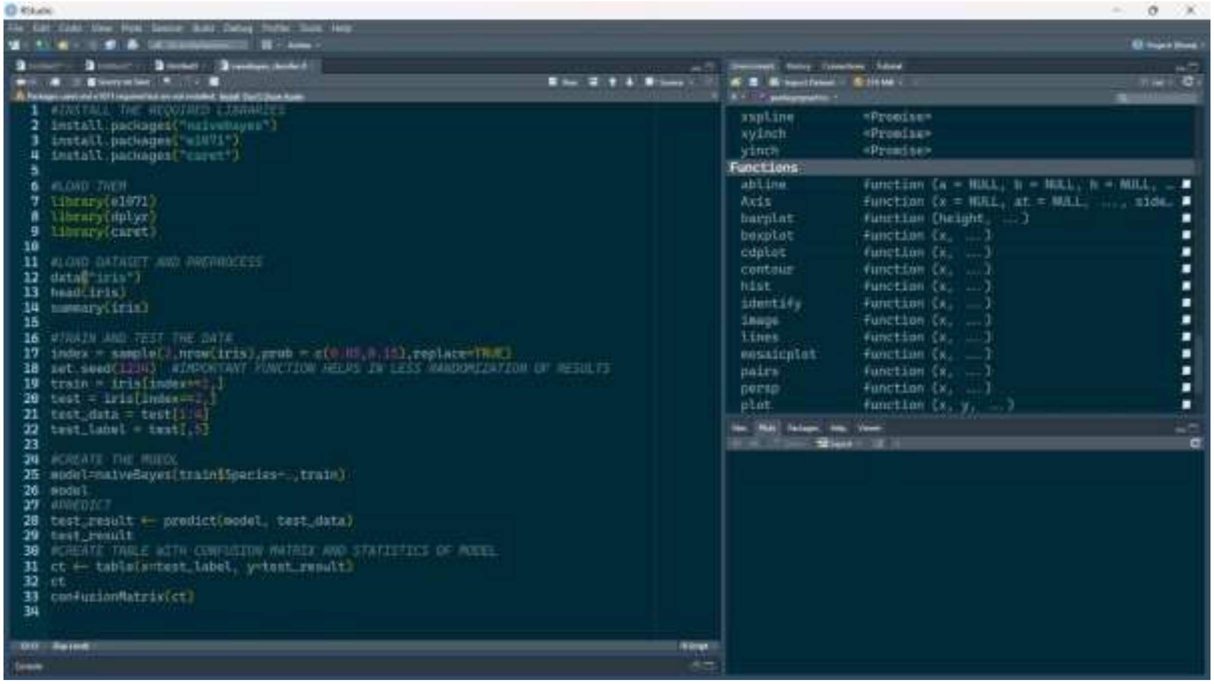
Objective: Naive Bayes is a Supervised Non-linear classification algorithm in R Programming. Naive Bayes classifiers are a family of simple probabilistic classifiers based on applying Baye's theorem with strong(Naive) independence assumptions between the features or variables. The Naive Bayes algorithm is called "Naive" because it makes the assumption that the occurrence of a certain feature is independent of the occurrence of other features.

Code:

```
#INSTALL THE REQUIRED LIBRARIES

install.packages("naivebayes") install.packages("e1071")

install.packages("caret")



#LOAD THEM

library(e1071) library(dplyr)

library(caret) #LOAD DATASET

AND PREPROCESS

data("iris") head(iris)

summary(iris)
```

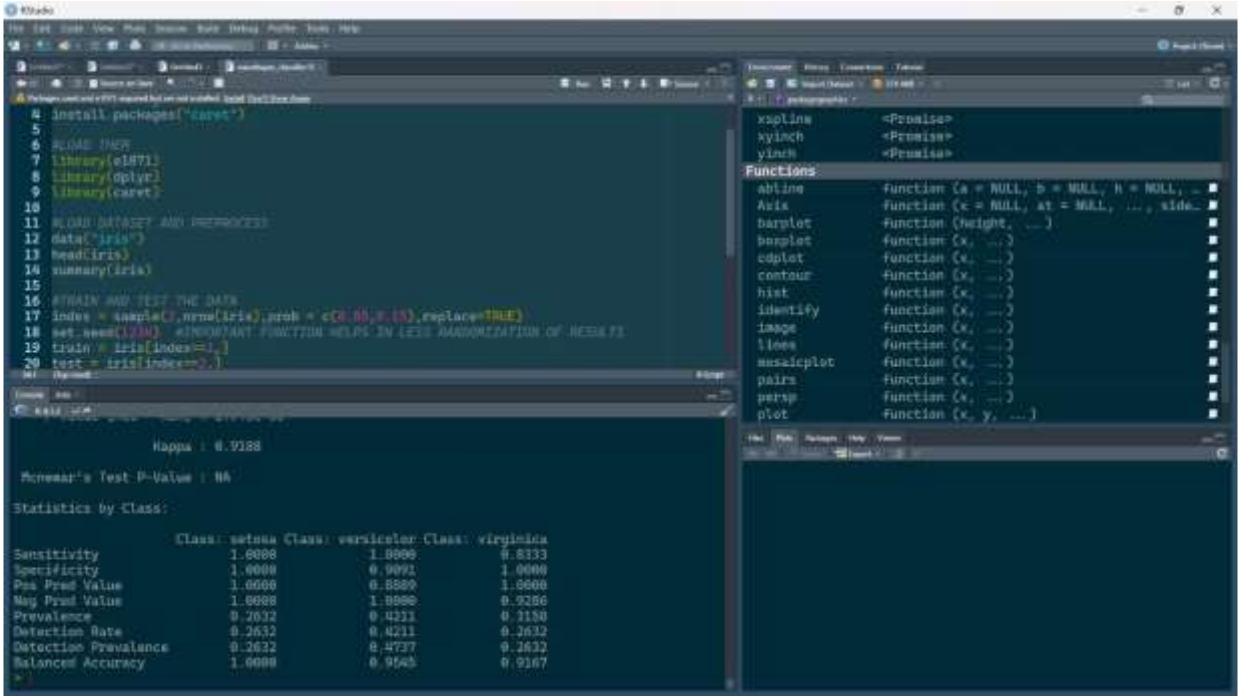
```
#TRAIN AND TEST THE DATA

index = sample(2,nrow(iris),prob = c(0.85,0.15),replace=TRUE)

set.seed(1234) #IMPORTANT FUNCTION HELPS IN LESS RANDOMIZATION OF RESULTS

train = iris[index==1,] test =
iris[index==2,]  test_data  =
test[1:4]      test_label      =
test[,5]


#CREATE THE MOEDL

model=naiveBayes(train$Species~.,train) model

#PREDICT

test_result <- predict(model, test_data) test_result



#CREATE TABLE WITH CONFUSION MATRIX AND STATISTICS OF MODEL

ct <- table(x=test_label, y=test_result) ct

confusionMatrix(ct)
```

OUTPUT:



```
#INSTALL THE REQUIRED LIBRARIES
install.packages("naiveBayes")
install.packages("e1071")
install.packages("caret")

#LOAD THEM
library(e1071)
library(dplyr)
library(caret)

#LOAD DATASET AND PREPROCESS
data("iris")
head(iris)
summary(iris)

#TRAIN AND TEST THE DATA
index = sample(2,nrow(iris),prob = c(0.85,0.15),replace=TRUE)
set.seed(1234)  #IMPORTANT FUNCTION HELPS IN LESS RANDOMIZATION OF RESULTS
train = iris[index==1,]
test = iris[index==2,]
test_data = test[1:4]
test_label = test[,5]

#CREATE THE MODEL
model=naiveBayes(train$Species~.,train)
model
#PREDICT
test_result <- predict(model, test_data)
test_result
#CREATE TABLE WITH CONFUSION MATRIX AND STATISTICS OF MODEL
ct <- table(x=test_label, y=test_result)
ct
confusionMatrix(ct)
```

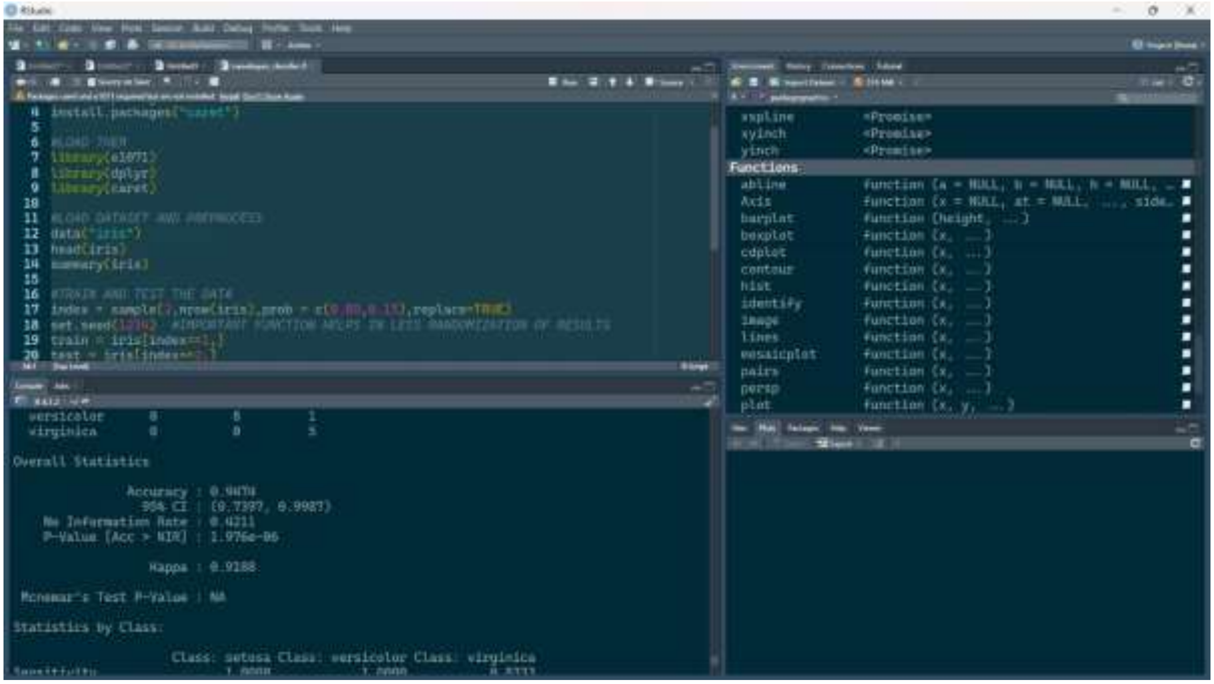
```
xspline      <Promise>
xyinch       <Promise>
yinch        <Promise>
Functions
abline       function (a = NULL, b = NULL, h = NULL, …
Axis         function (x = NULL, at = NULL, ..., side…
barplot      function (height, ...)
boxplot      function (x, ...)
cdplot       function (x, ...)
contour      function (x, ...)
hist         function (x, ...)
identify     function (x, ...)
image        function (x, ...)
lines        function (x, ...)
mosaicplot   function (x, ...)
pairs        function (x, ...)
persp        function (x, ...)
plot         function (x, y, ...)
```



```
install.packages("caret")

#LOAD THEM
library(e1071)
library(dplyr)
library(caret)

#LOAD DATASET AND PREPROCESS
data("iris")
head(iris)
summary(iris)

#TRAIN AND TEST THE DATA
index = sample(2,nrow(iris),prob = c(0.85,0.15),replace=TRUE)
set.seed(1234)  #IMPORTANT FUNCTION HELPS IN LESS RANDOMIZATION OF RESULTS
train = iris[index==1,]
test = iris[index==2,]
```

```
                  Kappa : 0.9188

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: setosa Class: versicolor Class: virginica
Sensitivity                 1.0000            1.0000           0.8333
Specificity                 1.0000            0.9091           1.0000
Pos Pred Value              1.0000            0.8889           1.0000
Neg Pred Value              1.0000            1.0000           0.9286
Prevalence                  0.2632            0.4211           0.3158
Detection Rate              0.2632            0.4211           0.2632
Detection Prevalence        0.2632            0.4737           0.2632
Balanced Accuracy           1.0000            0.9545           0.9167
>
```

```
xspline      <Promise>
xyinch       <Promise>
yinch        <Promise>
Functions
abline       function (a = NULL, b = NULL, h = NULL, …
Axis         function (x = NULL, at = NULL, ..., side…
barplot      function (height, ...)
boxplot      function (x, ...)
cdplot       function (x, ...)
contour      function (x, ...)
hist         function (x, ...)
identify     function (x, ...)
image        function (x, ...)
lines        function (x, ...)
mosaicplot   function (x, ...)
pairs        function (x, ...)
persp        function (x, ...)
plot         function (x, y, ...)
```



```
install.packages("caret")

#LOAD THEM
library(e1071)
library(dplyr)
library(caret)

#LOAD DATASET AND PREPROCESS
data("iris")
head(iris)
summary(iris)

#TRAIN AND TEST THE DATA
index = sample(2,nrow(iris),prob = c(0.80,0.15),replace=TRUE)
set.seed(1234)  #IMPORTANT FUNCTION HELPS IN LESS RANDOMIZATION OF RESULTS
train = iris[index==1,]
test = iris[index==2,]
```

```
  versicolor      0          5         1
  virginica       0          0         5

Overall Statistics

               Accuracy : 0.9474
                 95% CI : (0.7397, 0.9987)
    No Information Rate : 0.4211
    P-Value [Acc > NIR] : 1.976e-06

                  Kappa : 0.9188

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: setosa Class: versicolor Class: virginica
Sensitivity                 1.0000            1.0000           0.8333
```

```
  xspline          <Promise>
  xyinch           <Promise>
  yinch            <Promise>
Functions
  abline           function (a = NULL, b = NULL, h = NULL, …
  Axis             function (x = NULL, at = NULL, ..., side…
  barplot          function (height, ...)
  boxplot          function (x, ...)
  cdplot           function (x, ...)
  contour          function (x, ...)
  hist             function (x, ...)
  identify         function (x, ...)
  image            function (x, ...)
  lines            function (x, ...)
  mosaicplot       function (x, ...)
  pairs            function (x, ...)
  persp            function (x, ...)
  plot             function (x, y, ...)
```

Experiment-2.3

Student Name: Vikash Yadav **UID: 21BCS8093**
Branch: BE-CSE **Section/Group: 719/B**
Semester: 6th **Date of Performance: 19/04/2023**
Subject Name: Data Mining Lab **Subject Code: 20CSP-376**

Aim: To perform the cluster analysis by k-means method using R.

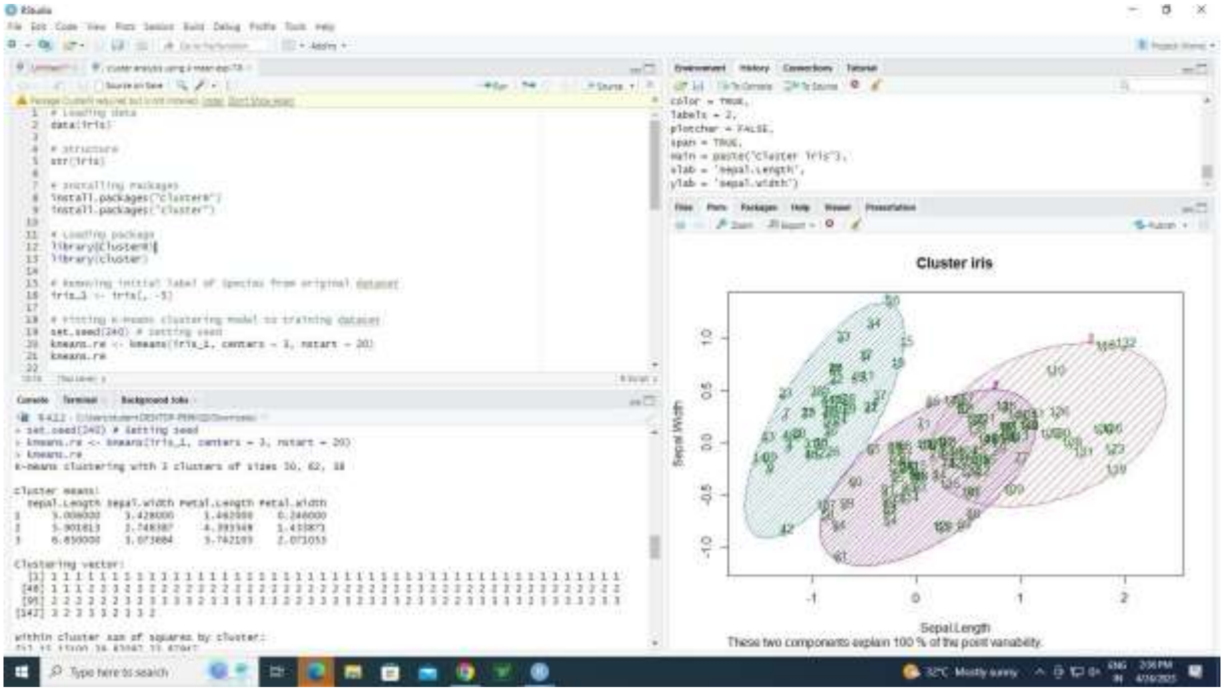
Objective: K Means Clustering in R Programming is an Unsupervised Non- linear algorithm that cluster data based on similarity or similar groups. It seeks to partition the observations into a pre-specified number of clusters. Segmentation of data takes place to assign each training example to a segment called a cluster. In the unsupervised algorithm, high reliance on raw data is given with large expenditure on manual review for review of relevance is given. It is used in a variety of fields like Banking, healthcare, retail, Media, etc.

Code:

```
# Loading data

data(iris) #

Structure

str(iris)



# Installing Packages

install.packages("ClusterR")

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 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')
```

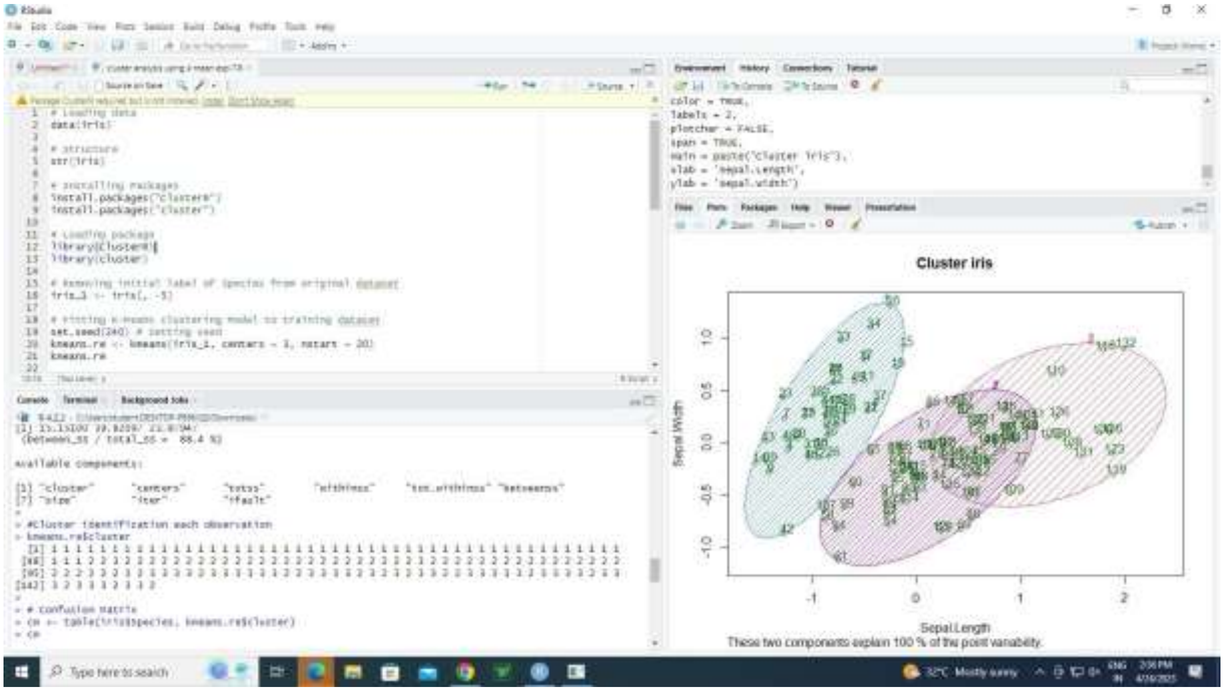
OUTPT:


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Addins
Cluster analysis using k mean exp-7.R
Source on Save
Run
Source
Console Terminal Background Jobs
Environment History Connections Tutorial
Files Plots Packages Help Viewer Presentation
Zoom Export
Publish
Cluster iris
Sepal.Width
Sepal.Length
-1.0
-0.5
0.0
0.5
1.0
-1
0
1
2
These two components explain 100 % of the point variability.
Type here to search
32°C Mostly sunny

```
# Loading data
data(iris)

# Structure
str(iris)

# Installing Packages
install.packages("ClusterR")
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
```

```
color = TRUE,
labels = 2,
plotchar = FALSE,
span = TRUE,
main = paste("Cluster iris"),
xlab = 'Sepal.Length',
ylab = 'Sepal.Width')
```

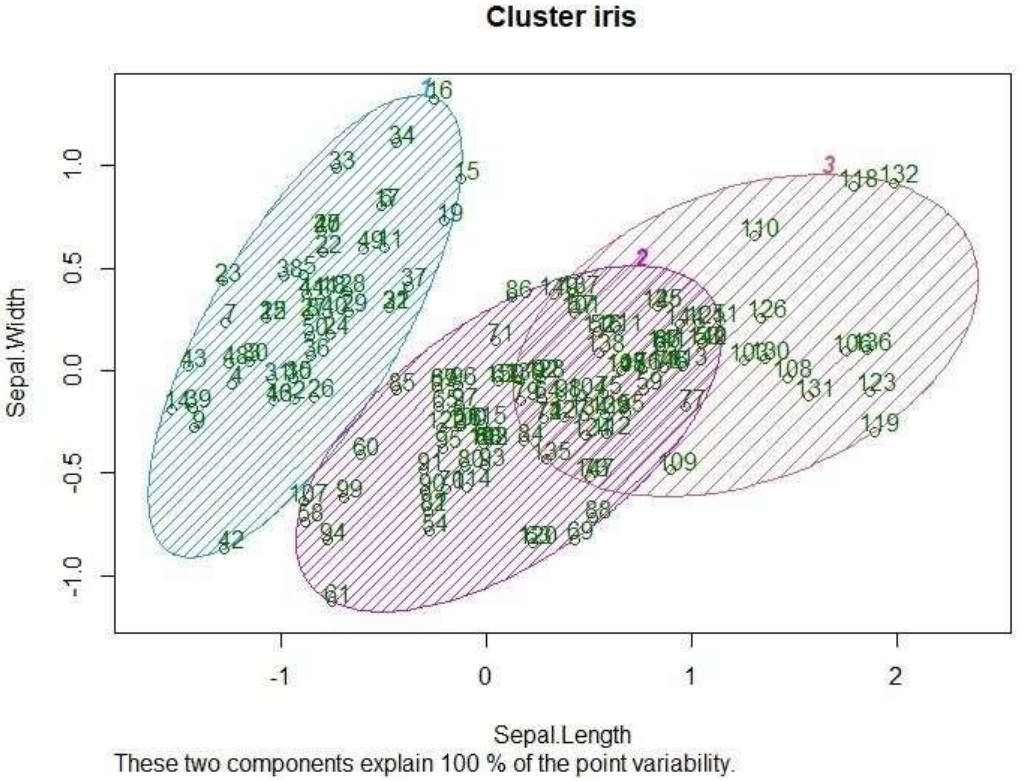
```
> set.seed(240) # Setting seed
> kmeans.re <- kmeans(iris_1, centers = 3, nstart = 20)
> kmeans.re
K-means clustering with 3 clusters of sizes 50, 62, 38

Cluster means:
  Sepal.Length Sepal.Width Petal.Length Petal.Width
1     5.006000    3.428000     1.462000    0.246000
2     5.901613    2.748387     4.393548    1.433871
3     6.850000    3.073684     5.742105    2.071053

Clustering vector:
  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [48] 1 1 1 2 2 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 [95] 2 2 2 2 2 2 3 2 3 3 3 3 2 3 3 3 3 3 3 2 2 3 3 3 3 2 3 2 3 2 3 3 2 2 3 3 3 3 3 2 3 3 3 3 2 3 3
[142] 3 2 3 3 3 2 3 3 2

Within cluster sum of squares by cluster:
[1] 15.15100 39.82097 23.87947
```

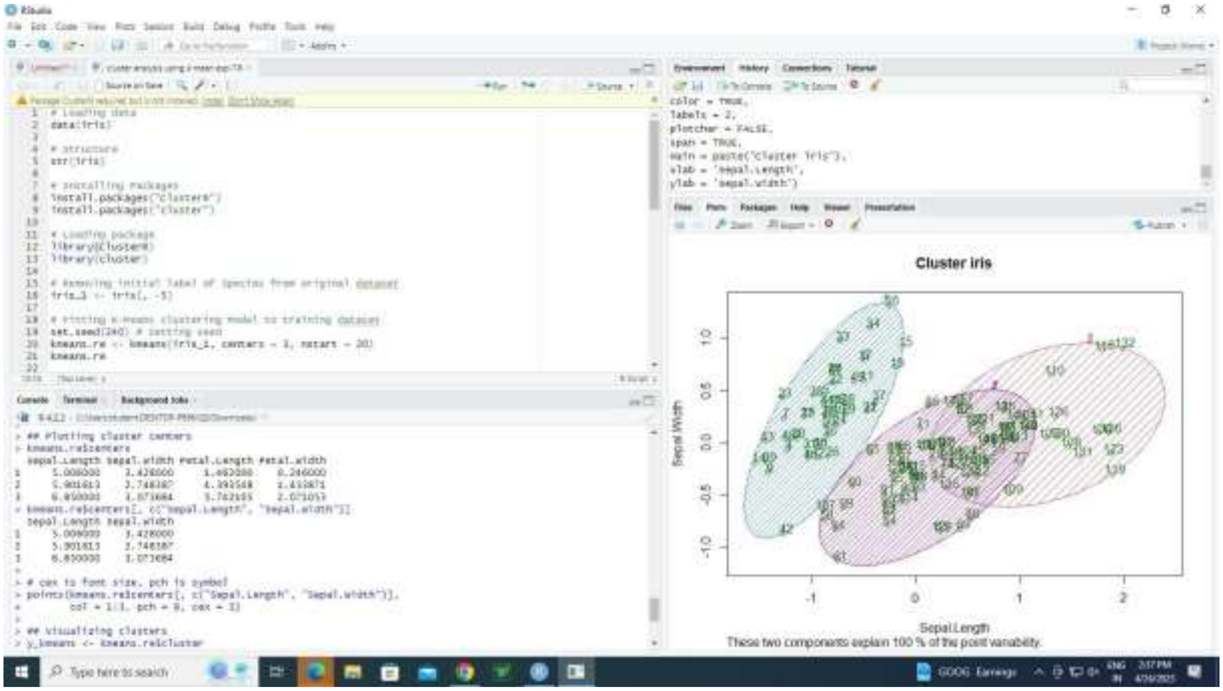

RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Addins
Cluster analysis using k mean exp-7.R
Source on Save
Run
Source
Console Terminal Background Jobs
Environment History Connections Tutorial
Files Plots Packages Help Viewer Presentation
Zoom Export
Publish
Cluster iris
Sepal.Width
Sepal.Length
-1.0
-0.5
0.0
0.5
1.0
-1
0
1
2
These two components explain 100 % of the point variability.
Type here to search
32°C Mostly sunny

```
# Loading data
data(iris)

# Structure
str(iris)

# Installing Packages
install.packages("ClusterR")
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
```

```
color = TRUE,
labels = 2,
plotchar = FALSE,
span = TRUE,
main = paste("Cluster iris"),
xlab = 'Sepal.Length',
ylab = 'Sepal.Width')
```

```
[1] 15.15100 39.82097 23.87947
 (between_SS / total_SS =  88.4 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss" "betweenss"
[7] "size"         "iter"         "ifault"
>
> #Cluster identification each observation
> kmeans.re$cluster
  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [48] 1 1 1 2 2 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 [95] 2 2 2 2 2 2 3 2 3 3 3 3 2 3 3 3 3 3 3 2 2 3 3 3 3 2 3 2 3 2 3 3 2 2 3 3 3 3 3 2 3 3 3 3 2 3 3
[142] 3 2 3 3 3 2 3 3 2
>
> # Confusion Matrix
> cm <- table(iris$Species, kmeans.re$cluster)
> cm
```





```
# Loading data
data(iris)

# Structure
str(iris)

# Installing Packages
install.packages("ClusterR")
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
```

```
> ## Plotting cluster centers
> kmeans.re$centers
  Sepal.Length Sepal.Width Petal.Length Petal.Width
1     5.006000    3.428000     1.462000    0.246000
2     5.901613    2.748387     4.393548    1.433871
3     6.850000    3.073684     5.742105    2.071053
> kmeans.re$centers[, c("Sepal.Length", "Sepal.Width")]
  Sepal.Length Sepal.Width
1     5.006000    3.428000
2     5.901613    2.748387
3     6.850000    3.073684
> 
> # 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
```

```
color = TRUE,
labels = 2,
plotchar = FALSE,
span = TRUE,
main = paste("Cluster iris"),
xlab = 'Sepal.Length',
ylab = 'Sepal.Width')
```

Experiment-3.1

Student Name: Vikash Yadav **UID: 21BCS8093**
Branch: BE-CSE **Section/Group: 719/B**
Semester: 6th **Date of Performance: 26/04/2023**
Subject Name: Data Mining Lab **Subject Code: 20CSP-376**

Aim/Overview of the practical:

- To perform hierarchical clustering using R programming.

Apparatus/Simulator used:

- R- Studio
- R-language
- Datasets, cluster, factoextra

Theory:

Hierarchical cluster analysis (also known as hierarchical clustering) is a clustering technique where clusters have a hierarchy or a predetermined order. Hierarchical clustering can be represented by a tree-like structure called as **Dendrogram**. There are two types of hierarchical clustering.

Agglomerative hierarchical clustering: This is a bottom-up approach where each data point starts in its own cluster and as one moves up the hierarchy, similar pairs of clusters are merged.

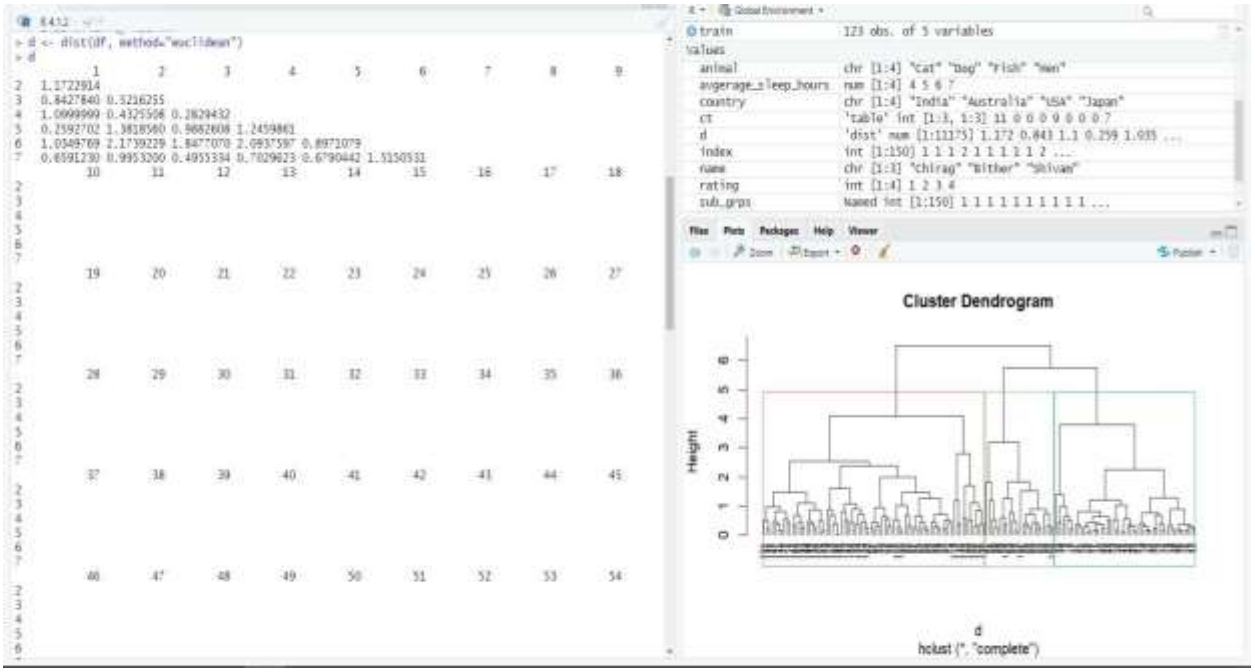
Divisive hierarchical clustering: This is a top-down approach where all data points start in one cluster and as one moves down the hierarchy, clusters are split recursively.

Code :

```
# Load required packages

library(datasets) # contains iris dataset
library(cluster) # clustering algorithms
library(factoextra) # visualization
library(purrr) # to use map_dbl()
function

# Load and preprocess the dataset

df <- iris[, 1:4]
```

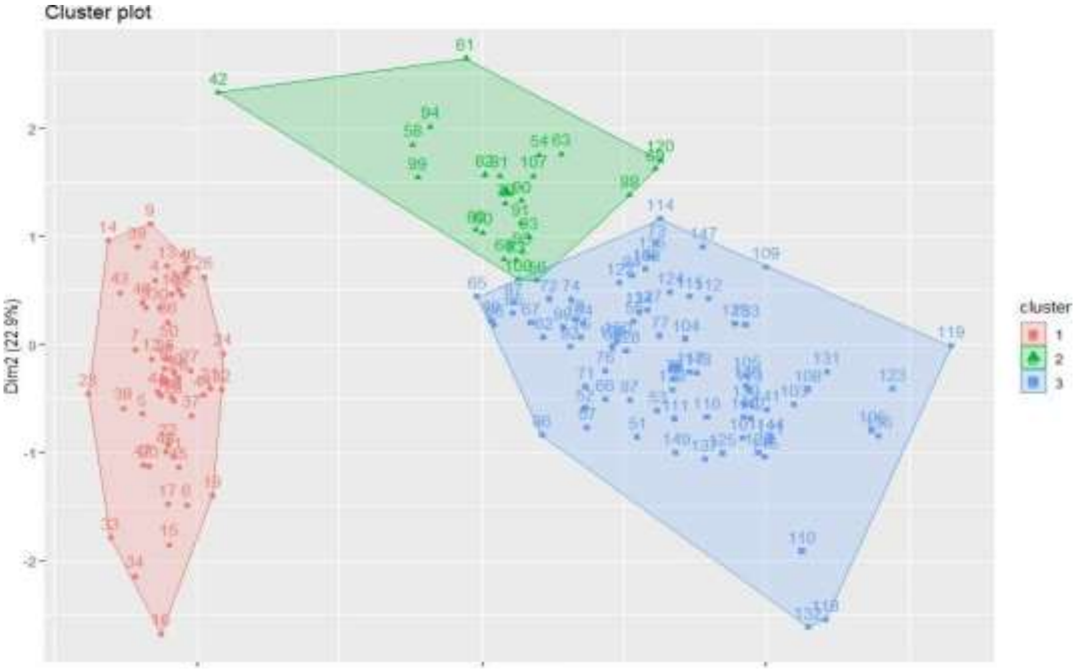
```
df <- na.omit(df) df <- scale(df) #
Dissimilarity matrix d <- dist(df,
method="euclidean") d hc1 <- hclust(d,
method = "complete") plot(hc1, cex =
0.6, hang=-1) sub_grps <- cutree(hc1,
k=3) fviz_cluster(list(data = df, cluster =
sub_grps))

plot(hc1, cex = 0.6, hang=-1)
rect.hclust(hc1, k = 3, border=2:4)
```

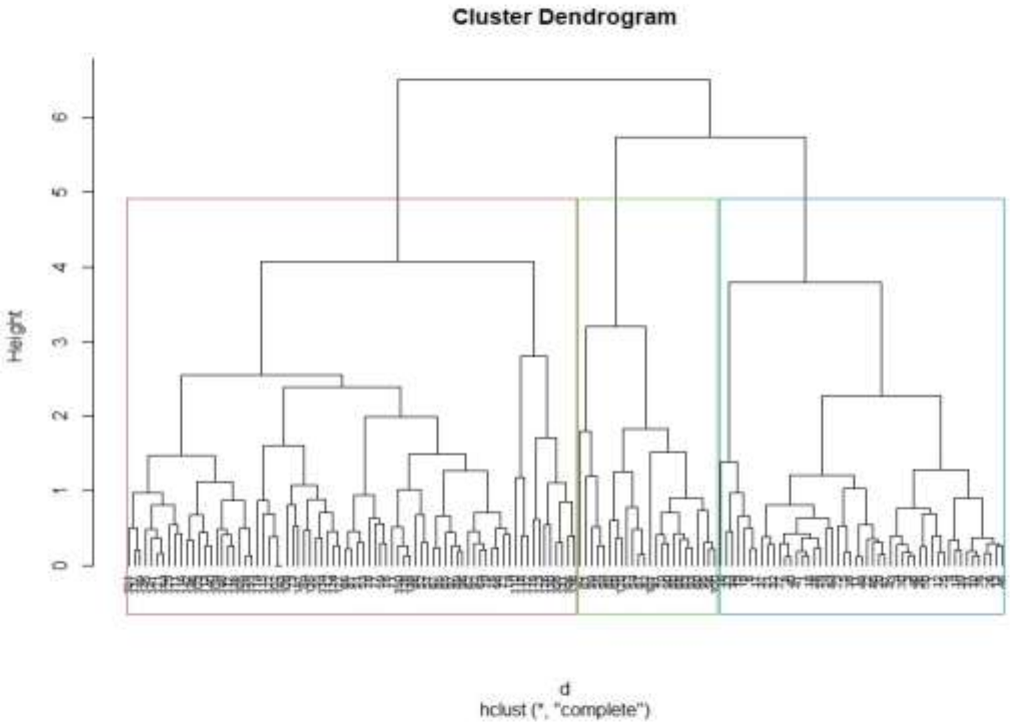
OUTPUT:



```
> d <- dist(df, method="euclidean")
> d
          1         2         3         4         5         6         7         8         9
2 1.1722914
3 0.8427840 0.5216255
4 1.0999999 0.4325508 0.2829432
5 0.2592702 1.3818560 0.9882808 1.2459861
6 1.0349769 2.1739229 1.8477070 2.0937597 0.8971079
7 0.6591230 0.9953200 0.4955334 0.7029823 0.6790442 1.5150531
         10        11        12        13        14        15        16        17        18
2
3
4
5
6
7
         19        20        21        22        23        24        25        26        27
2
3
4
5
6
7
         28        29        30        31        32        33        34        35        36
2
3
4
5
6
7
         37        38        39        40        41        42        43        44        45
2
3
4
5
6
7
         46        47        48        49        50        51        52        53        54
2
3
4
5
6
```

```
train                123 obs. of 5 variables
Values
  animal               chr [1:4] "Cat" "Dog" "Fish" "Hen"
  average_sleep_hours  num [1:4] 4 5 6 7
  country              chr [1:4] "India" "Australia" "USA" "Japan"
  ct                   'table' int [1:3, 1:3] 11 0 0 0 9 0 0 0 7
  d                    'dist' num [1:11175] 1.172 0.843 1.1 0.259 1.035 ...
  index                int [1:150] 1 1 1 2 1 1 1 1 1 2 ...
  name                 chr [1:3] "Chirag" "Bither" "Shivam"
  rating               int [1:4] 1 2 3 4
  sub_grps             Named int [1:150] 1 1 1 1 1 1 1 1 1 1 ...
```


Cluster plot
Dim2 (22.9%)
2
1
0
-1
-2
cluster
1
2
3
61
42
94
58
99
54
63
107
120
89
114
147
109
119
131
123
108
106
110
132
118
14
9
39
43
34
28
38
33
24
16
15
19
65
86
51
140


Cluster Dendrogram
Height
0
1
2
3
4
5
6
d
hclust (*, "complete")

Experiment-3.2

Student Name: Vikash Yadav **UID: 21BCS8093**
Branch: BE-CSE **Section/Group: 719/B**
Semester: 6th **Date of Performance: 03/05/2023**
Subject Name: Data Mining Lab **Subject Code: 20CSP-376**

1. Aim:

Study of Regression Analysis using R programming.

2. Objective:

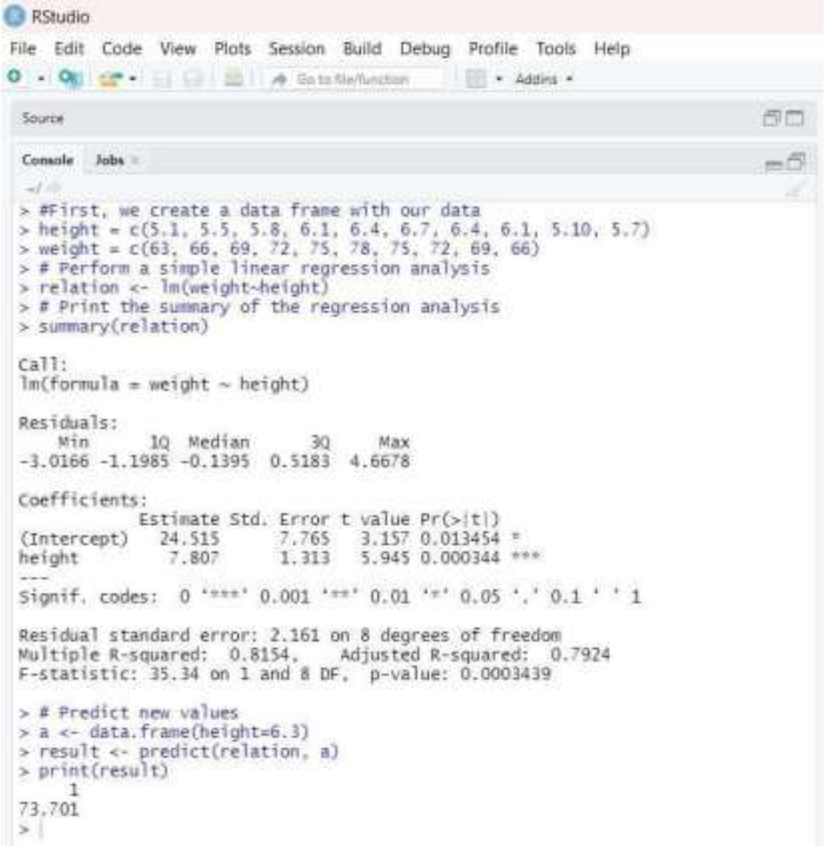
- Regression Analysis is used to develop a predictive model for estimating the value of a dependent variable based on one or more independent variables.
- It is used to determine the strength and direction of the relationship between two variables.
- It is used to identify which independent variables have a significant impact on the dependent variable. • It is used to test hypotheses about the relationship between variables.

3. Code and Output:

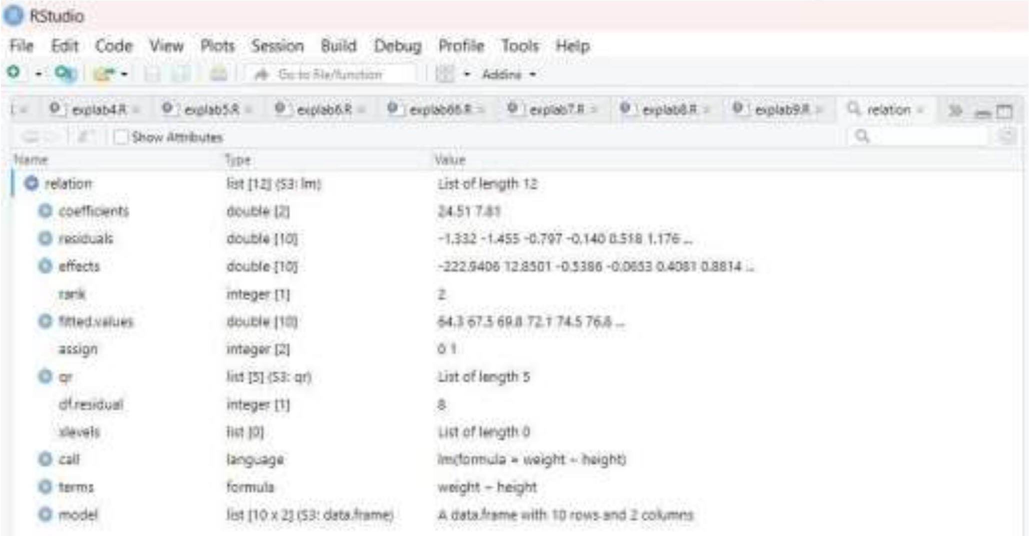
- PROGRAM

```
#First, we create a data frame with our data
height = c(5.1, 5.5, 5.8, 6.1, 6.4, 6.7, 6.4, 6.1, 5.10, 5.7)
weight = c(63, 66, 69, 72, 75, 78, 75, 72, 69, 66)
# Perform a simple linear regression analysis relation <-
lm(weight~height)

# Print the summary of the regression analysis
summary(relation)

# Predict new values a <-
data.frame(height=6.3) result <-
predict(relation, a)
print(result)
```

- OUTPUT


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
Source
Console Jobs
~/

```
> #First, we create a data frame with our data
> height = c(5.1, 5.5, 5.8, 6.1, 6.4, 6.7, 6.4, 6.1, 5.10, 5.7)
> weight = c(63, 66, 69, 72, 75, 78, 75, 72, 69, 66)
> # Perform a simple linear regression analysis
> relation <- lm(weight~height)
> # Print the summary of the regression analysis
> summary(relation)

Call:
lm(formula = weight ~ height)

Residuals:
    Min      1Q  Median      3Q     Max 
-3.0166 -1.1985 -0.1395  0.5183  4.6678 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)   24.515      7.765   3.157 0.013454 *  
height         7.807      1.313   5.945 0.000344 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.161 on 8 degrees of freedom
Multiple R-squared:  0.8154,	Adjusted R-squared:  0.7924 
F-statistic: 35.34 on 1 and 8 DF,  p-value: 0.0003439

> # Predict new values
> a <- data.frame(height=6.3)
> result <- predict(relation, a)
> print(result)
     1 
73.701 
> 
```



Name	Type	Value
relation	list [12] (S3: lm)	List of length 12
coefficients	double [2]	24.51 7.81
residuals	double [10]	-1.332 -1.455 -0.797 -0.140 0.518 1.176 ...
effects	double [10]	-222.9406 12.8501 -0.5386 -0.0653 0.4081 0.8814 ...
rank	integer [1]	2
fitted.values	double [10]	64.3 67.3 69.8 72.1 74.5 76.8 ...
assign	integer [2]	0 1
qr	list [5] (S3: qr)	List of length 5
df.residual	integer [1]	8
xlevels	list [0]	List of length 0
call	language	lm(formula = weight ~ height)
terms	formula	weight ~ height
model	list [10 x 2] (S3: data.frame)	A data.frame with 10 rows and 2 columns

Experiment-3.3

Student Name: Vikash Yadav **UID: 21BCS8093**
Branch: BE-CSE **Section/Group: 719/B**
Semester: 6th **Date of Performance: 10/05/2023**
Subject Name: Data Mining Lab **Subject Code: 20CSP-376**

1. Aim:

Outlier detection using R programming.

2. Objective:

- Outlier detection is used to identify anomalies in data that do not conform to expected patterns or behaviors.
- It is used to improve accuracy and reliability of statistical analyses by detecting and removing outliers.
- It is used to identify potential errors or fraudulent activities in a dataset.
- It is used to optimize machine learning models by removing outliers that may skew results.

3. Code and Output:

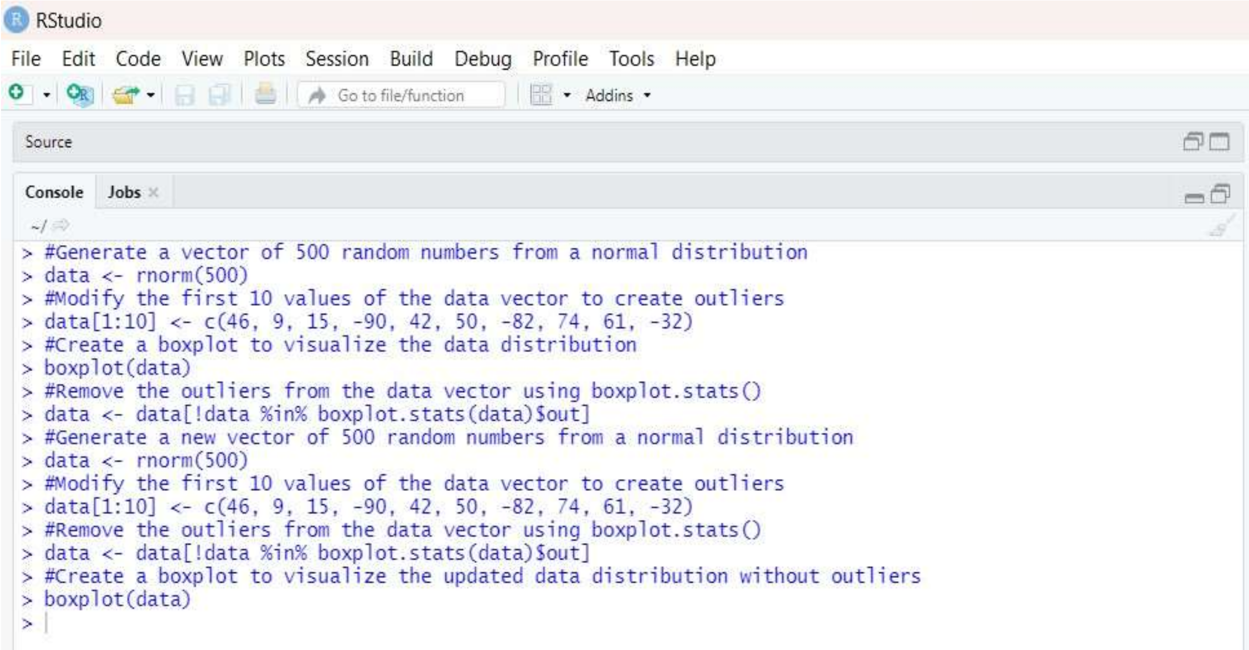
- **PROGRAM**

```
#Generate a vector of 500 random numbers from a normal distribution data <- rnorm(500)

#Modify the first 10 values of the data vector to create outliers data[1:10] <- c(46, 9, 15, -90, 42, 50, -
82, 74, 61, -32)

#Create a boxplot to visualize the data distribution boxplot(data)

#Remove the outliers from the data vector using boxplot.stats() data <-
data[!data %in% boxplot.stats(data)$out]

#Generate a new vector of 500 random numbers from a normal distribution data <- rnorm(500)


#Modify the first 10 values of the data vector to create outliers data[1:10] <- c(46,
9, 15, -90, 42, 50, -82, 74, 61, -32)
```

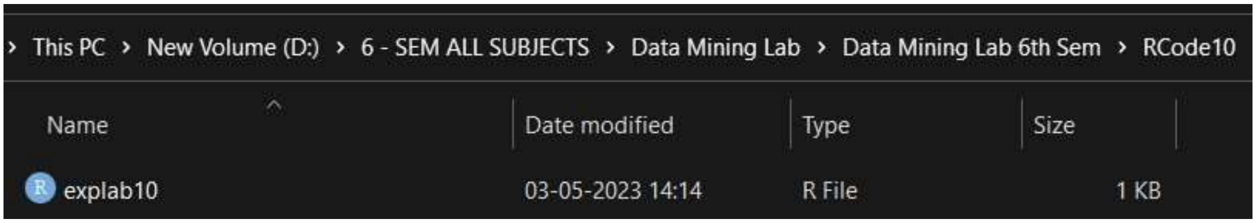
#Remove the outliers from the data vector using boxplot.stats() data <- data[!data %in% boxplot.stats(data)$out]

#Create a boxplot to visualize the updated data distribution without outliers boxplot(data)

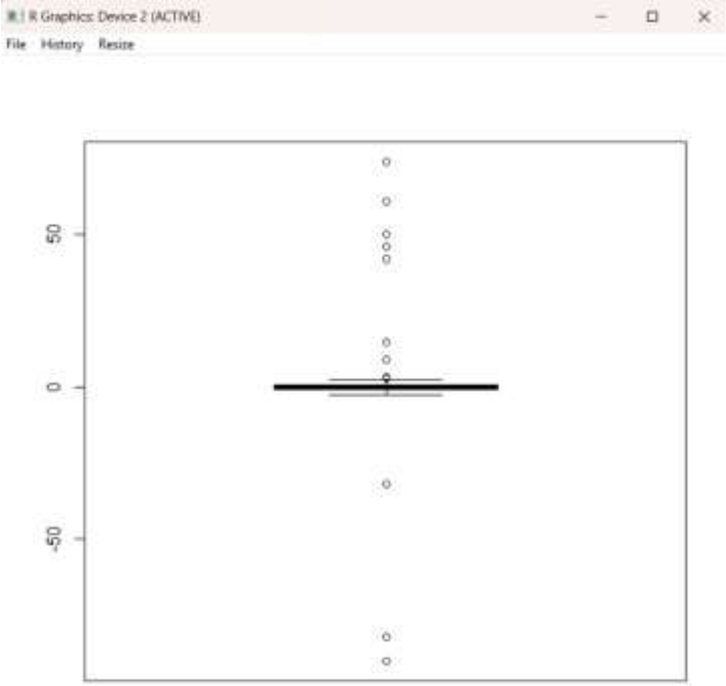
- **OUTPUT**

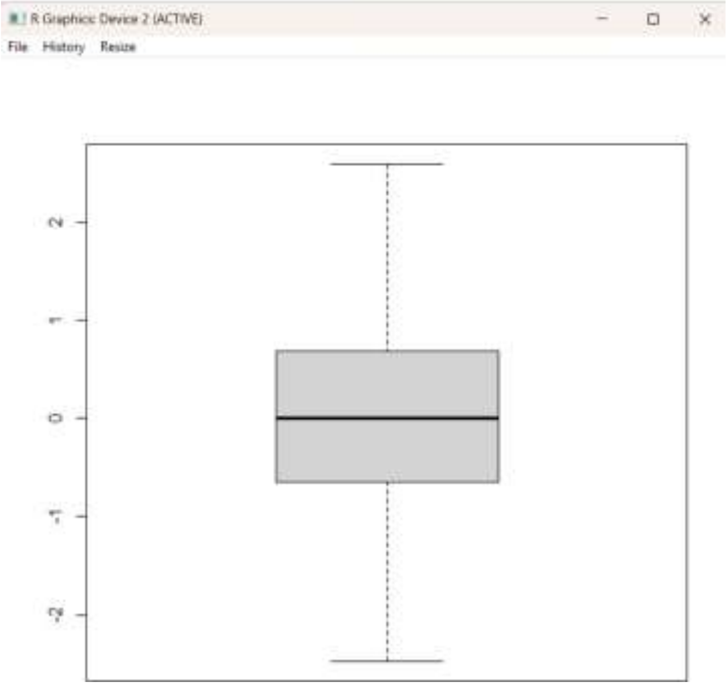


```
> #Generate a vector of 500 random numbers from a normal distribution
> data <- rnorm(500)
> #Modify the first 10 values of the data vector to create outliers
> data[1:10] <- c(46, 9, 15, -90, 42, 50, -82, 74, 61, -32)
> #Create a boxplot to visualize the data distribution
> boxplot(data)
> #Remove the outliers from the data vector using boxplot.stats()
> data <- data[!data %in% boxplot.stats(data)$out]
> #Generate a new vector of 500 random numbers from a normal distribution
> data <- rnorm(500)
> #Modify the first 10 values of the data vector to create outliers
> data[1:10] <- c(46, 9, 15, -90, 42, 50, -82, 74, 61, -32)
> #Remove the outliers from the data vector using boxplot.stats()
> data <- data[!data %in% boxplot.stats(data)$out]
> #Create a boxplot to visualize the updated data distribution without outliers
> boxplot(data)
> |
```


› This PC › New Volume (D:) › 6 - SEM ALL SUBJECTS › Data Mining Lab › Data Mining Lab 6th Sem › RCode10

Name	Date modified	Type	Size
explab10	03-05-2023 14:14	R File	1 KB


R Graphics: Device 2 (ACTIVE)
File
History
Resize
50
0
-50


R Graphics: Device 2 (ACTIVE)
File
History
Resize
2
1
0
-1
-2