Estd.2012

NILGIRI COLLEGE
OF ARTS AND SCIENCE

(Affiliated to Bharathiar University)

PG DEPARTMENT OF COMPUTER SCIENCE

DATA MINING USING
R - LAB

PRACTICAL RECORD

2020—2021

NAME . . . .

REGISTER No . . . .

CLASS . . . .

SEMESTER . . . .

Estd.2012

NILGIRI COLLEGE OF ARTS AND SCIENCE

(Affiliated to Bharathiar University)

PG DEPARTMENT OF COMPUTER SCIENCE

DATA MINING USING R - LAB

PRACTICAL RECORD

NAME . . . . CLASS . . . .

REGISTER No. . . . .

Certified that this is the bonafide record of work done by the above student of M. Sc. Computer Science in the Data Mining Using R Laboratory during the year 2020- 2021.

Staff in-charge **Head of the Department** **Principal**

Submitted for the Practical Examination held on . . . .

Internal Examiner **External Examiner**

INDEX

Ex. No : 1	**APRIORI ALGORITHM**
Date :	

Aim :

Implement Apriori algorithm to extract association rule of datamining

Algorithm :

Step 1:Start the Process

Step 2:Select the cran mirror

Step 3:install the package arules.

Step 4: include the library Function of arules

Step 5:inspect rules and summary the rules

Stop 6: Stop the process

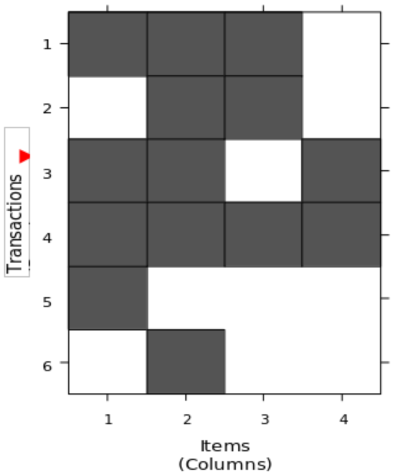
Code :

```
library(arules)
tr<-read.transactions("E:/Mic.txt",format="basket",sep=",")
inspect(tr)
image(tr)
rules<-apriori(tr,parameter=list(supp=0.5,conf=0.5))
inspect(rules)
summary(rules)
```

OUTPUT

```
 tr<-read.transactions("E:/Mic.txt",format="basket",sep=",")
 Warning message:
 In readLines(file, encoding = encoding) :
   incomplete final line found on 'E:/Mic.txt'
 >inspect(tr)
     items
 [1] {A,B,C}
 [2] {B,C}
 [3] {A,B,D}
 [4] {A,B,C,D}
 [5] {A}
 [6] {B}

 Apriori
 Parameter specification:
  confidence minval smax arem  aval originalSupport maxtime support minlen
         0.5    0.1    1 none FALSE            TRUE       5     0.5
               1 maxlen target ext
      10  rules TRUE

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

 Absolute minimum support count: 3
 set item appearances ...[0 item(s)] done [0.00s].
 set transactions ...[4 item(s), 6 transaction(s)] done [0.00s].
 sorting and recoding items ... [3 item(s)] done
 [0.00s]. creating transaction tree ... done [0.00s].
 checking subsets of size 1 2 done
```

```
[0.00s]. writing ... [7 rule(s)] done

[0.00s]. creating S4 object ... done

[0.00s].
```



```
    inspect(rules)

    lhs    rhs support  confidence coverage  lift count
[1] {}  => {C} 0.5000000 0.5000000  1.0000000 1.0  3
[2] {}  => {A} 0.6666667 0.6666667  1.0000000 1.0  4
[3] {}  => {B} 0.8333333 0.8333333  1.0000000 1.0  5
[4] {C} => {B} 0.5000000 1.0000000  0.5000000 1.2  3
[5] {B} => {C} 0.5000000 0.6000000  0.8333333 1.2  3
[6] {A} => {B} 0.5000000 0.7500000  0.6666667 0.9  3
[7] {B} => {A} 0.5000000 0.6000000  0.8333333 0.9  3

set of 7 rules

rule length distribution (lhs +
rhs):sizes
1 2
3 4
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu. Max.
  1.000  1.000  2.000  1.571  2.000  2.000


 summary of quality measures:
    support        confidence       coverage          lift
  Min.  :0.5000   Min.   :0.5000   Min.   :0.5000   Min.   :0.900
  1st Qu.:0.5000  1st Qu.:0.6000  1st Qu.:0.7500  1st Qu.:0.950
  Median :0.5000  Median :0.6667   Median :0.8333   Median :1.000
  Mean  :0.5714  Mean  :0.7071    Mean   :0.8333  Mean   :1.029
  3rd Qu.:0.5833  3rd Qu.:0.7917  3rd Qu.:1.0000  3rd Qu.:1.100
  Max. :0.8333  Max. :1.0000  Max. :1.0000  Max. :1.200
     count
  Min.   :3.00
  1st
  Qu.:3.000
  Median :3.000
  Mean :3.429
  3rd Qu.:3.500
  Max.    :5.000

 mining info:
  data ntransactions support confidence
  tr                6     0.5          0.5
```

RESULT :

Thus the program is successfully executed and output is verified.

Ex. No : 2	**K-MEANS CLUSTERING**
Date :	

Aim :

Implement k means Clustering.

Algorithm :

Step 1:Start the Process

Step 2:Install the iris dataset

Step3:display the items in the iris dataset

Step4:Display the plot diagram of iris dataset

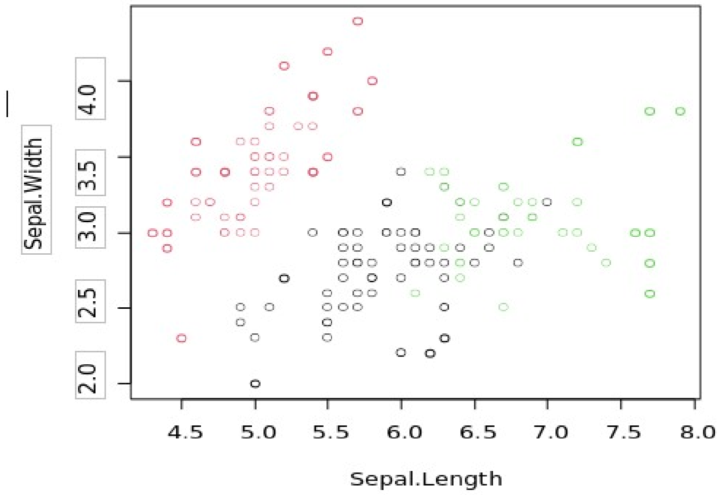
Step5:Stop the process.

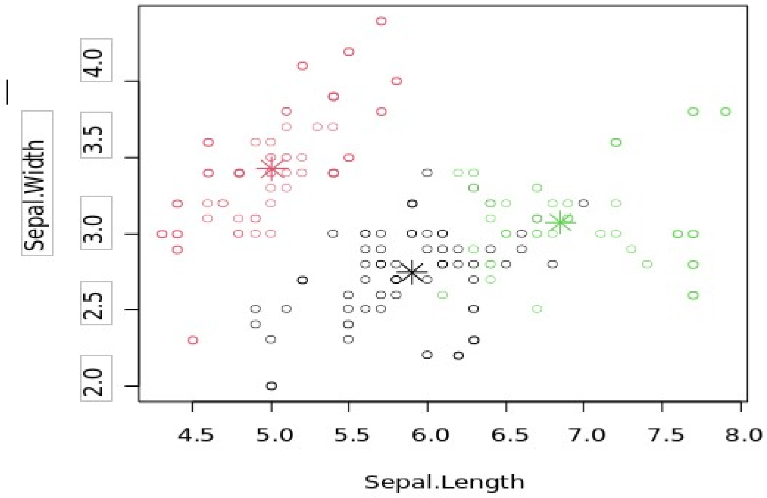
Code :

```
 > iris2 <- iris
> iris2$Species <- NULL
> (kmeans.result <- kmeans(iris2, 3))
> table(iris$Species, kmeans.result$cluster)
> plot(iris2[c("Sepal.Length", "Sepal.Width")], col = kmeans.result$cluster)
> # plot cluster centers
points(kmeans.result$centers[,c("Sepal.Length", "Sepal.Width")], col = 1:3,pch = 8, cex=2)
```

OUTPUT

```
K-means clustering with 3 clusters of sizes 62, 50, 38

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

Clustering vector:
  [1] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 [38] 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [75] 1 1 1 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3 1 3 3 3 3 1 3 3 3 3
[112] 3 3 1 1 3 3 3 3 1 3 1 3 1 3 3 1 1 3 3 3 3 3 1 3 3 3 3 1 3 3 3 1 3 3 3 1 3
[149] 3 1

Within cluster sum of squares by cluster:
[1] 39.82097 15.15100 23.87947
 (between_SS / total_SS =  88.4 %)

Available components:
[1] "cluster"    "centers"     "totss"       "withinss"   "tot.withinss"
[6] "betweenss"     "size"      "iter"        "ifault"



             1  2  3
  setosa      0 50  0
  versicolor 48  0  2
  virginica  14  0 36
```





RESULT :

Thus the program is successfully executed and output is verified.

Ex. No : 3	HIERARCHAL CLUSTERING
Date :	

Aim :

Implement any one Hierarchal Clustering.

Algorithm :

Step1: Start the Process

Step2: Install the iris datasets

Step3: Specify the Species in the program

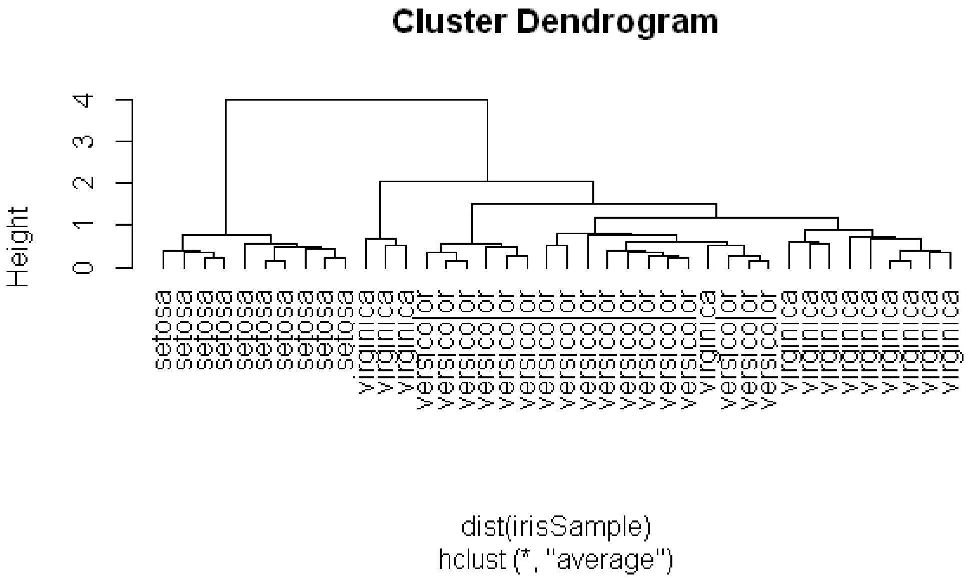
Step4: verify the output

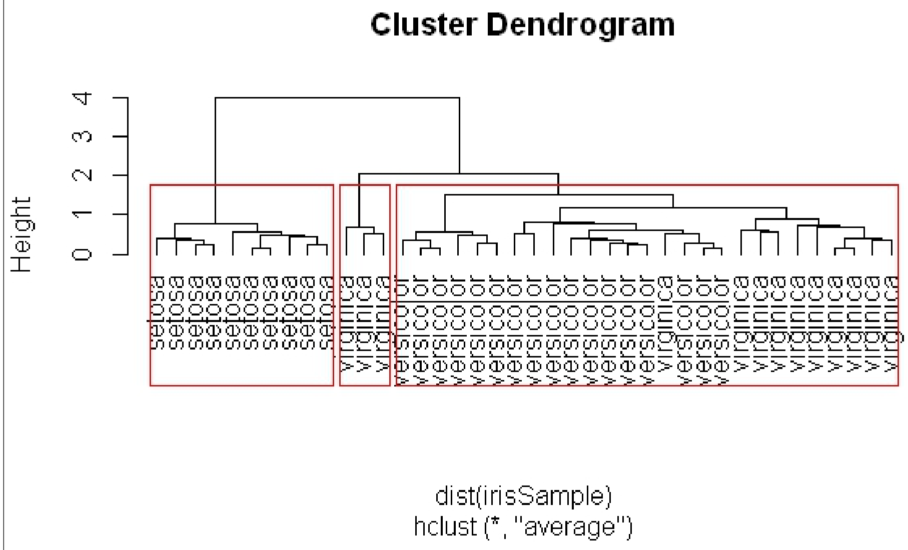
Step5: Stop the Process

Code :

```
 >idx <- sample(1:dim(iris)[1], 40)
 > irisSample <- iris[idx,]
 > irisSample$Species <- NULL
> hc <- hclust(dist(irisSample), method="ave")
> plot(hc, hang = -1, labels=iris$Species[idx])
> rect.hclust(hc, k=3)
> groups <- cutree(hc, k=3)
```

OUTPUT





RESULT :

Thus the program is successfully executed and output is verified.

Ex. No : 4	CLASSIFICATION ALGORITHM
Date :	

Aim :

Implement Classification Algorithm

Algorithm :

Step1:Start the Process

Step2:Install the iris datasets in the program

Step3:set the plot in Decision tree based classification

Step4:Verify the output

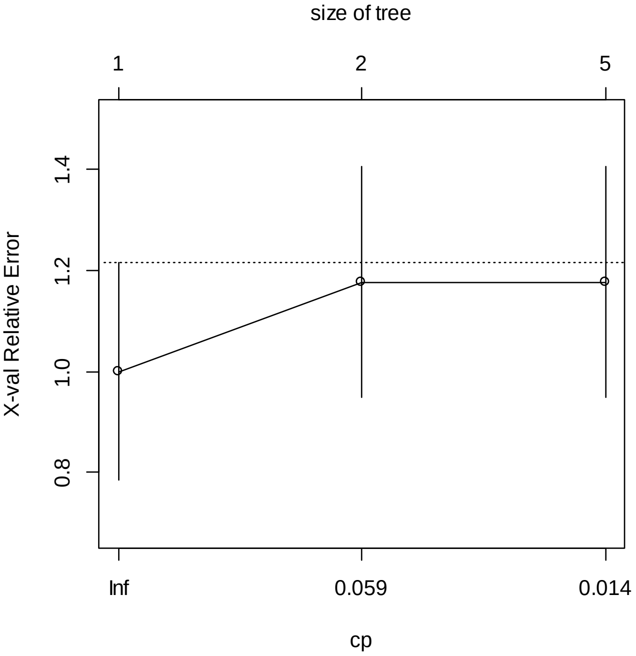
Step5:Stop the Process

Code :

```
>library(rpart)
> fit<-rpart(Kyphosis~Age+Number+Start,data=kyphosis,method="class")
> printcp(fit)
> plotcp(fit)
> summary(fit)
> plot(fit,uniform=TRUE,main="classification tree for kyphosis")
>text(fit,uniform=TRUE,all=TRUE,cex=.8)
```

OUTPUT

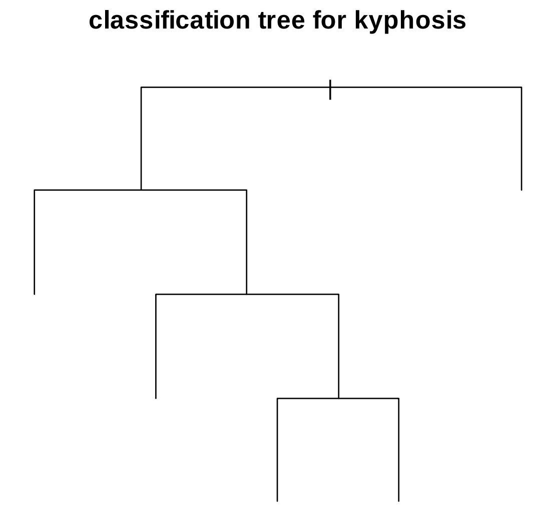
```
Classification tree:
rpart(formula = Kyphosis ~ Age + Number + Start, data = kyphosis,
    method = "class")

Variables actually used in tree construction:
[1] Age   Start

Root node error: 17/81 = 0.20988

n= 81

        CP nsplit rel error xerror    xstd
1 0.176471      0   1.00000 1.0000 0.21559
2 0.019608      1   0.82353 1.1765 0.22829
3 0.010000      4   0.76471 1.1765 0.22829
```

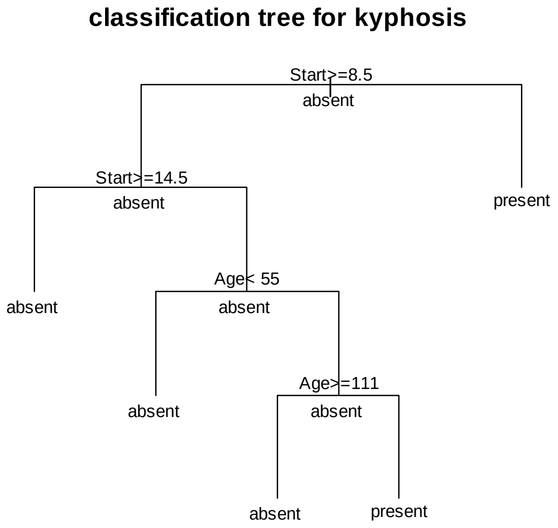


```
Call:
rpart(formula = Kyphosis ~ Age + Number + Start, data = kyphosis,
    method = "class")
  n= 81

          CP nsplit rel error   xerror      xstd
1 0.17647059      0 1.0000000 1.000000 0.2155872
2 0.01960784      1 0.8235294 1.176471 0.2282908
3 0.01000000      4 0.7647059 1.176471 0.2282908

Variable importance
   Start    Age Number
      64     24     12
```

```
Node number 1: 81 observations,    complexity param=0.1764706
  predicted class=absent   expected loss=0.2098765  P(node) =1
    class counts:    64    17
   probabilities: 0.790 0.210
  left son=2 (62 obs) right son=3 (19 obs)
  Primary splits:
      Start  < 8.5  to the right, improve=6.762330, (0 missing)
      Number < 5.5  to the left,  improve=2.866795, (0 missing)
      Age    < 39.5 to the left,  improve=2.250212, (0 missing)
  Surrogate splits:
      Number < 6.5  to the left,  agree=0.802, adj=0.158, (0 split)

Node number 2: 62 observations,    complexity param=0.01960784
  predicted class=absent   expected loss=0.09677419  P(node) =0.7654321
    class counts:    56     6
   probabilities: 0.903 0.097
  left son=4 (29 obs) right son=5 (33 obs)
  Primary splits:
      Start  < 14.5 to the right, improve=1.0205280, (0 missing)
      Age    < 55   to the left,  improve=0.6848635, (0 missing)
      Number < 4.5  to the left,  improve=0.2975332, (0 missing)
  Surrogate splits:
      Number < 3.5  to the left,  agree=0.645, adj=0.241, (0 split)
      Age    < 16   to the left,  agree=0.597, adj=0.138, (0 split)

Node number 3: 19 observations
  predicted class=present  expected loss=0.4210526  P(node) =0.2345679
    class counts:     8    11
   probabilities: 0.421 0.579

Node number 4: 29 observations
  predicted class=absent   expected loss=0  P(node) =0.3580247
    class counts:    29     0
   probabilities: 1.000 0.000

Node number 5: 33 observations,    complexity param=0.01960784
  predicted class=absent   expected loss=0.1818182  P(node) =0.4074074
```

```
    class counts:    27     6
   probabilities: 0.818 0.182
  left son=10 (12 obs) right son=11 (21 obs)
  Primary splits:
      Age    < 55   to the left,  improve=1.2467530, (0 missing)
      Start  < 12.5 to the right, improve=0.2887701, (0 missing)
      Number < 3.5  to the right, improve=0.1753247, (0 missing)
  Surrogate splits:
      Start  < 9.5  to the left,  agree=0.758, adj=0.333, (0 split)
      Number < 5.5  to the right, agree=0.697, adj=0.167, (0 split)

Node number 10: 12 observations
  predicted class=absent   expected loss=0  P(node) =0.1481481
    class counts:    12     0
   probabilities: 1.000 0.000

Node number 11: 21 observations,    complexity param=0.01960784
  predicted class=absent   expected loss=0.2857143  P(node) =0.2592593
    class counts:    15     6
   probabilities: 0.714 0.286
  left son=22 (14 obs) right son=23 (7 obs)
  Primary splits:
      Age    < 111  to the right, improve=1.71428600, (0 missing)
      Start  < 12.5 to the right, improve=0.79365080, (0 missing)
      Number < 3.5  to the right, improve=0.07142857, (0 missing)

Node number 22: 14 observations
  predicted class=absent   expected loss=0.1428571  P(node) =0.1728395
    class counts:    12     2
   probabilities: 0.857 0.143


Node number 23: 7 observations
  predicted class=present  expected loss=0.4285714  P(node) =0.08641975
    class counts:     3     4
   probabilities: 0.429 0.571
```





RESULT :

Thus the program is successfully executed and output is verified.

Ex. No : 5	DECISION TREE
Date :	

Aim :

Implement Decision Tree

Algorithm :

Step1: Start the Process

Step2:install the iris datasets

Step3:include here train and test data

Step4:install the package party

Step5:Verify the output

Step6:Stop the process

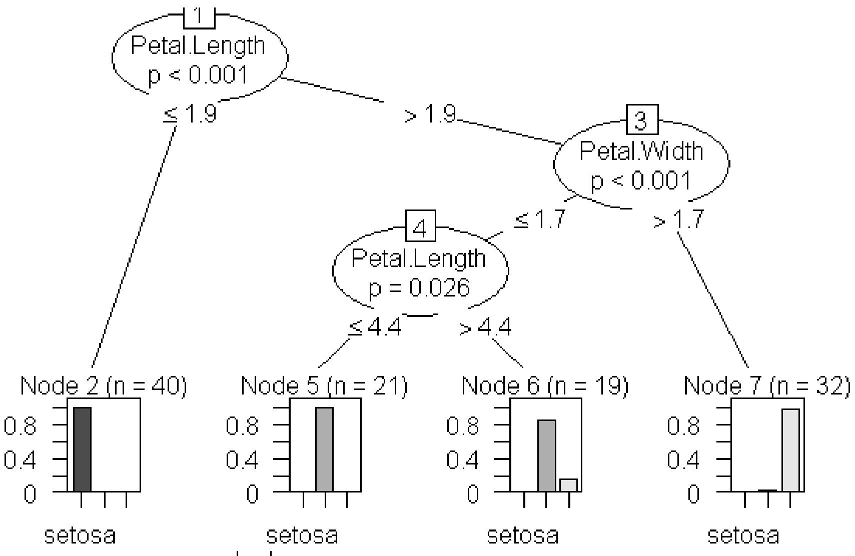
Code :

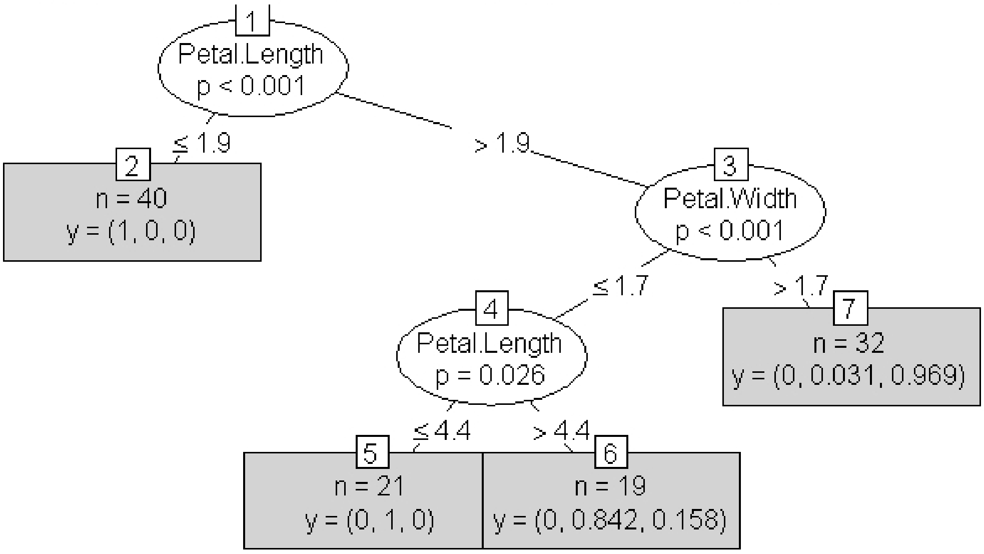
```
> str(iris)
> set.seed(1234)
> ind <- sample(2, nrow(iris), replace=TRUE, prob=c(0.7, 0.3))
> trainData <- iris[ind==1,]
> testData <- iris[ind==2,]
> library(party)
> myFormula <- Species ~ Sepal.Length + Sepal.Width + Petal.Length + Petal.Width
> iris_ctree <- ctree(myFormula, data=trainData)
> table(predict(iris_ctree), trainData$Species)
> print(iris_ctree)
> plot(iris_ctree)
> plot(iris_ctree, type="simple")
```

OUTPUT

```
'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 ...
            setosa versicolor virginica
  setosa        40          0        0
  versicolor     0         37        3
  virginica      0          1       31
```

```
Conditional inference tree with 4 terminal

nodes Response:  Species
 Inputs: Sepal.Length, Sepal.Width, Petal.Length, Petal.Width
 Number of observations:  112


 1) Petal.Length <= 1.9; criterion = 1, statistic =
   104.643 2)*  weights = 40

 1) Petal.Length > 1.9
   3) Petal.Width <= 1.7; criterion = 1, statistic = 48.939
     4) Petal.Length <= 4.4; criterion = 0.974, statistic =
       7.397 5)*  weights = 21
     4) Petal.Length >
       4.4 6)*  weights
       = 19

   3) Petal.Width >
     1.7 7)*  weights
     = 32
```





RESULT :

Thus the program is successfully executed and output is verified.

Ex. No : 6	LINEAR REGRESSION
Date :	

Aim :

Implement Linear Regression

Algorithm :

Step1: Start the Process

Step2:read the health dataset

Step3:campare lungcapacity and age

Step4:Make a code for abline

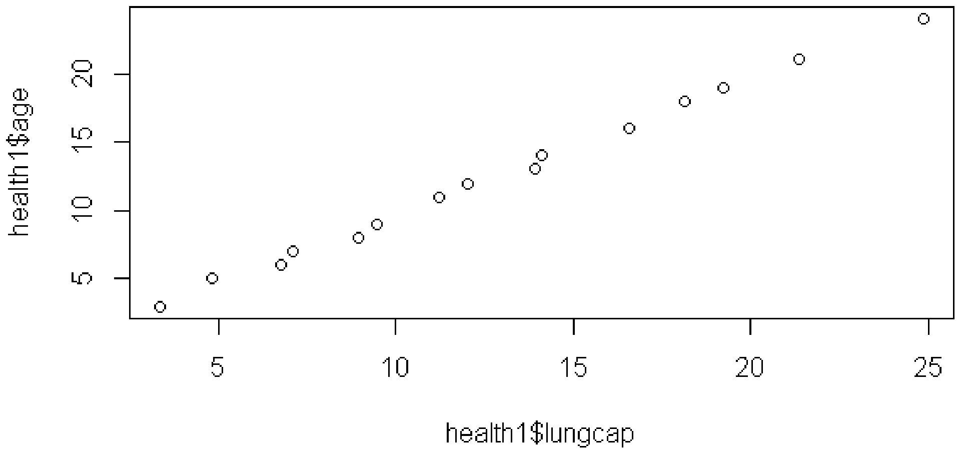
Step5:Verify the output

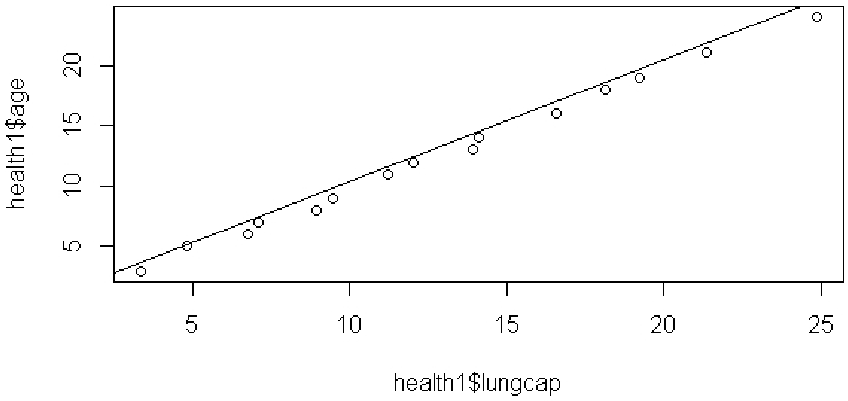
Step6:Stop the process

Code :

```
> health1<-read.csv("E:/health.csv")

> lm(health1$lungcap~health1$age)

> plot(health1$lungcap, health1$age)

> abline(lm(health1$lungcap~health1$age))
```

OUTPUT





```
Call:

lm(formula = health1$lungcap ~ health1$age)

Coefficients:

(Intercept) health1$age

0.2783

1.0087
```

RESULT :

Thus the program is successfully executed and output is verified.

Ex. No : 7	DATA VISUALIZATION
Date :	

Aim :

Implement Data Visualization

Algorithm :

Step1: Start the Process

Step2:read the sample dataset

Step3:Display the Histogram Diagram

Step4:Display the barplot and boxplot Diagram

Step5:Verify the output

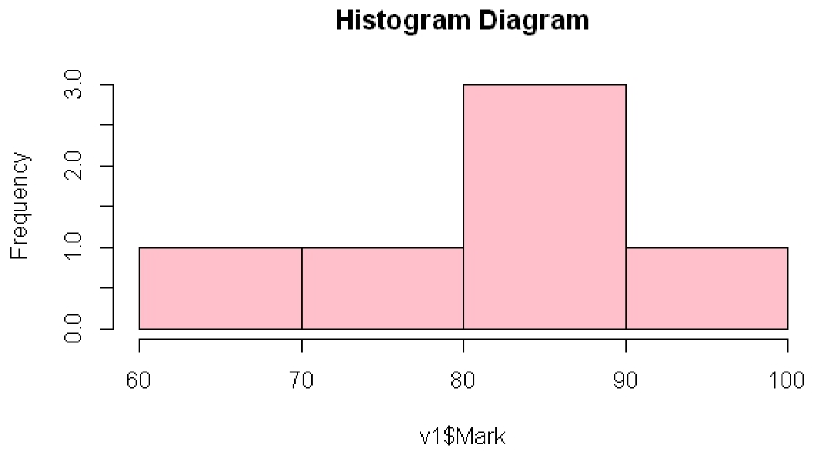
Step6:Stop the process

Code :

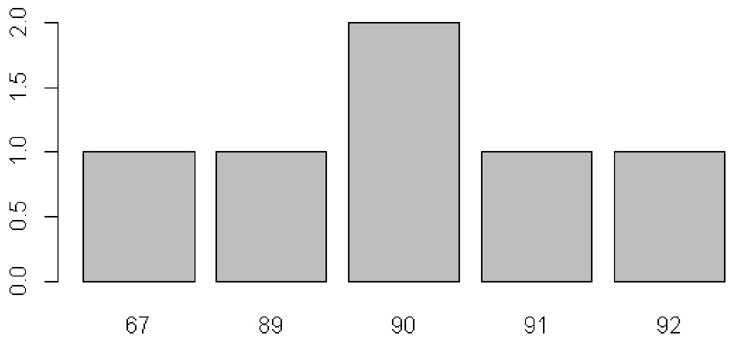
```
> v1<-read.csv("E:/sample dataset.csv")

> hist(v1$Mark,col="pink",main="Histogram Diagram")

> v2<-table(v1$Percent)

> barplot(v2)

> boxplot(v1$Mark)

> summary(v1$Mark)
```

OUTPUT

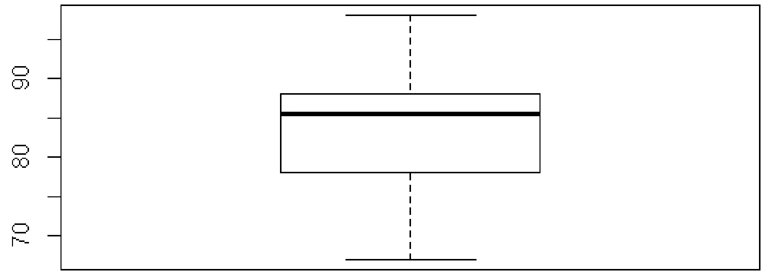
```
Min.     1st Qu.  Median   Mean     3rd Qu.   Max.
67.00     79.75    85.50    83.67     87.50    98.00
```



Barplot



Boxplot



RESULT :

Thus the program is successfully executed and output is verified.