

7. PREPROCESSING OF DATA USING WEKA

Open file... Open URL... Open DB... Generate... Undo Edit... Save...

se: **None** Apply Stop

relation: weather
Attributes: 5
Sum of weights: 14

Selected attribute
Name: outlook
Missing: 0 (0%)
Distinct: 3
Type: Nominal
Unique: 0 (0%)

No.	Label	Count	Weight
1	sunny	5	5
2	overcast	4	4
3	rainy	5	5

Class: play (Nom) Visualize

Remove

8. K-MEANS CLUSTERING BY WEKA

SimpleKMeans -init 0 -max-candidates 100 -periodic-pruning 10000 -min-density 2.0 -t1 -1.25 -t2 -1.0 -N 3 -A "weka.core.EuclideanDistance -R first-last" -I 500 -num-slots 1 -S 10

node

training set

plied test set Set...

centage split % 66

ses to clusters evaluation

om) class

e clusters for visualization

Ignore attributes

Start Stop

st (right-click for options)

5 - SimpleKMeans

4 - SimpleKMeans

Clusterer output

Attribute	Full Data (150.0)	0 (61.0)	1 (50.0)	2 (39.0)
sepal.length	5.8433	5.8885	5.006	6.8462
sepal.width	3.054	2.7377	3.418	3.0821
petal.length	3.7587	4.3967	1.464	5.7026
petal.width	1.1987	1.418	0.244	2.0795

Time taken to build model (full training data) : 0 seconds

=== Model and evaluation on training set ===

Clustered Instances

```

0      61 ( 41%)
1      50 ( 33%)
2      39 ( 26%)

Class attribute: class
Classes to Clusters:

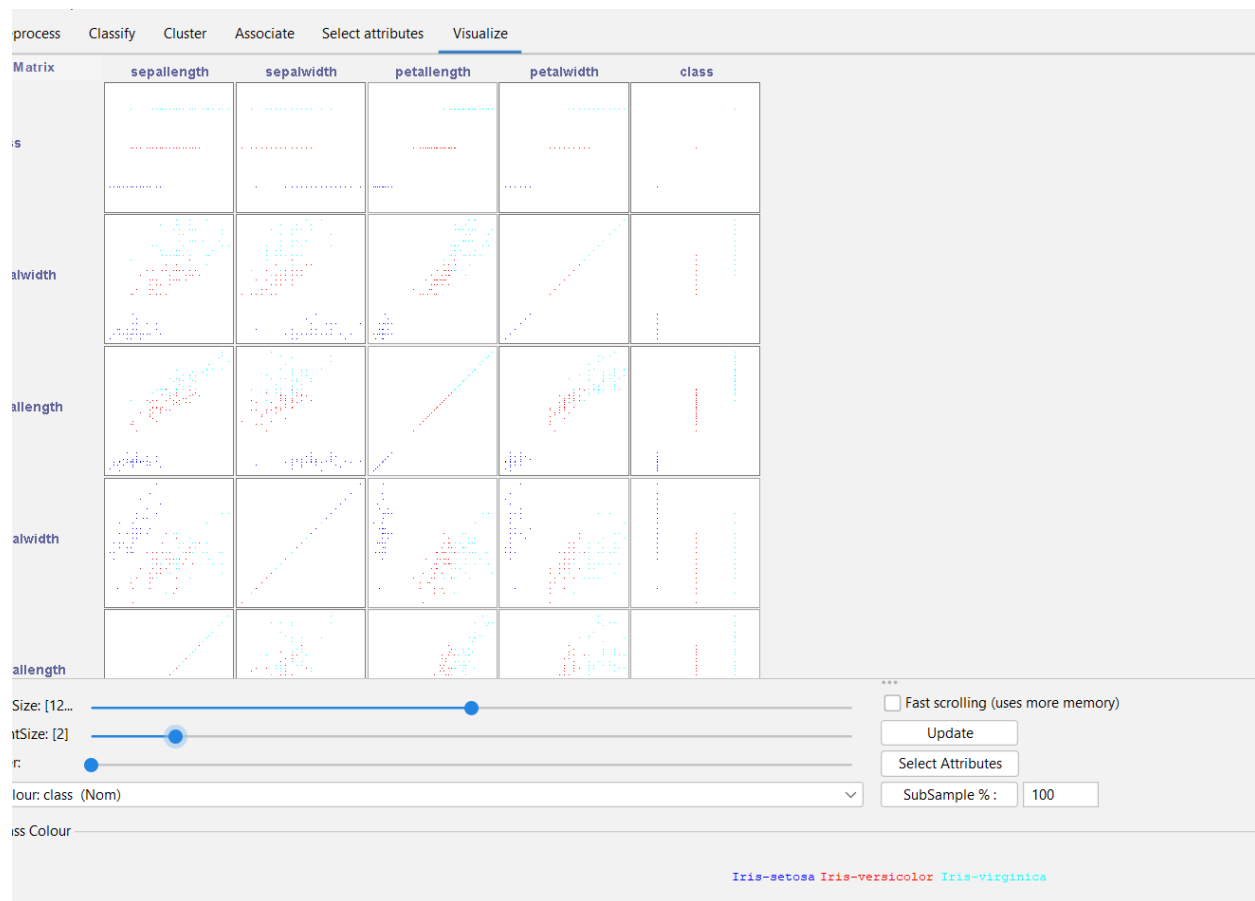
  0  1  2 <-- assigned to cluster
  0 50  0 | Iris-setosa
47  0  3 | Iris-versicolor
14  0 36 | Iris-virginica

Cluster 0 <-- Iris-versicolor
Cluster 1 <-- Iris-setosa
Cluster 2 <-- Iris-virginica

Incorrectly clustered instances :      17.0      11.3333 %

```

OUTPUT INFORMATION OF SIMPLE K MEANS CLUSTERING –CLUSTERER OUTPUT



K MEANS CLUSTERING IN WEKAA USING IRIS INFORMATION – PLOTTING ..

9. DATA ANALYSIS BY EXPECTATION MAXIMISATION ALGORITHM USING WEKA

Preprocess **Classify** **Cluster** Associate Select attributes Visualize

Clusterer

Choose **EM** -I 100 -N -1 -X 10 -max -1 -II-cv 1.0E-6 -II-iter 1.0E-6 -M 1.0E-6 -K 10 -num-slots 1 -S 100

Cluster mode

☐ Use training set

☐ Supplied test set

☐ Percentage split %

☒ **Classes to clusters evaluation**

(Nom) class

☒ Store clusters for visualization

Ignore attributes

Result list (right-click for options)

10:38:24 - EM

Clusterer output

	0	1	2	3	4
mean	0.1240	2.1327	1.3757	0.3020	1.1650
std. dev.	0.0557	0.2359	0.2196	0.1212	0.1351

Time taken to build model (full training data) : 0.4 seconds

=== Model and evaluation on training set ===

Clustered Instances

Cluster	Count	Percentage
0	28	(19%)
1	35	(23%)
2	42	(28%)
3	22	(15%)
4	23	(15%)

Log likelihood: -1.60803

Class attribute: class

Classes to Clusters:

```

0 1 2 3 4 <-- assigned to cluster
28 0 0 22 0 | Iris-setosa
0 0 27 0 23 | Iris-versicolor
0 35 15 0 0 | Iris-virginica

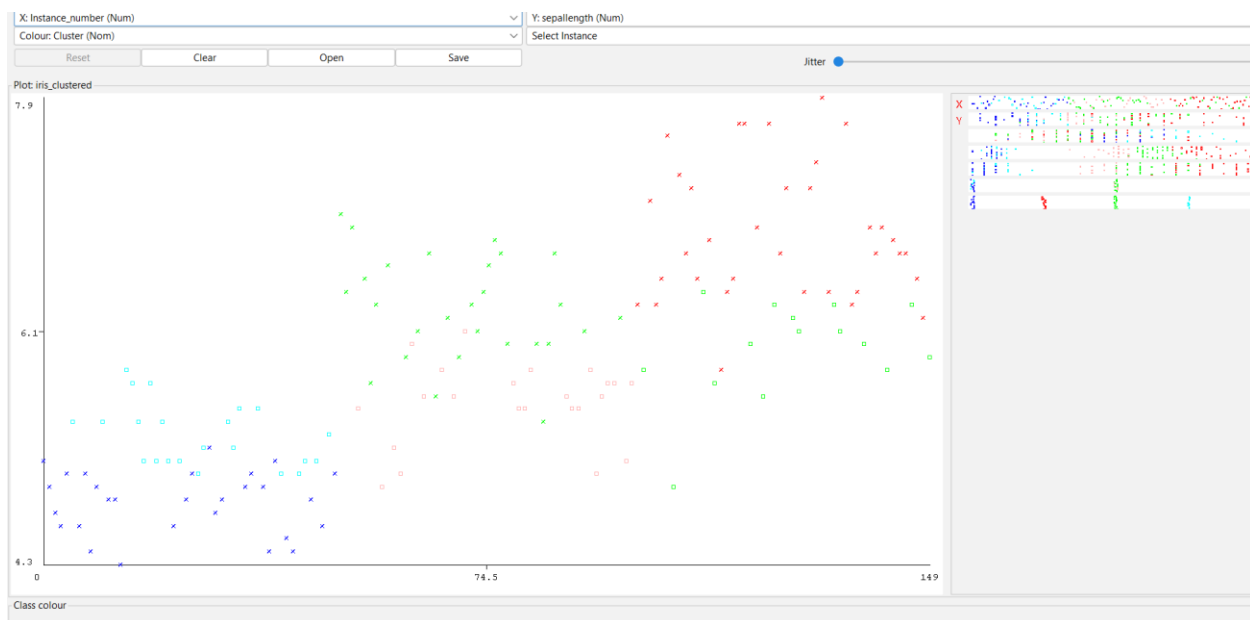
```

Cluster 0 <-- Iris-setosa
Cluster 1 <-- Iris-virginica
Cluster 2 <-- Iris-versicolor
Cluster 3 <-- No class
Cluster 4 <-- No class

Incorrectly clustered instances : 60.0 40 %

Status

DATA ANALYSIS BY EXPECTATION MAXIMISATION ALGORITHM – OUTPUT INNFO OF IRIS



DATA ANALYSIS BY EXPECTATION MAXIMISATION ALGORITHM—PLOT DIAGRAM OF IRIS INFO

10.DATA ANALYSIS BY HIERARCHICAL CLUSTERING. IN WEKA

reprocess Classify Cluster Associate Select attributes Visualize

Clusterer: HierarchicalClusterer -N 2 -L SINGLE -P -A "weka.core.EuclideanDistance -R first-last"

Clusterer mode:
☐ Use training set
☐ Supplied test set
☐ Percentage split
☒ Classes to clusters evaluation
(Nom) play
☒ Store clusters for visualization

Ignore attributes: Start Stop

Result list (right-click for options): 0:49:18 - HierarchicalClusterer

Clusterer output:

```
Ignored: play
Test mode: Classes to clusters evaluation on training data

=== Clustering model (full training set) ===

Cluster 0
(((1.0:0.69805,1.0:0.69805):0.12055,1.0:0.81901):0.18604,(1.0:0.3674,1.0:0.3674):0.63765):0,(1.0:0.52484,(1.0:0.33333,1.0:0.33333):0.19151):0.480

Cluster 1
(0.0:0.68769,0.0:0.68769)

Time taken to build model (full training data) : 0 seconds

=== Model and evaluation on training set ===

Clustered Instances
0 12 ( 86%)
1 2 ( 14%)

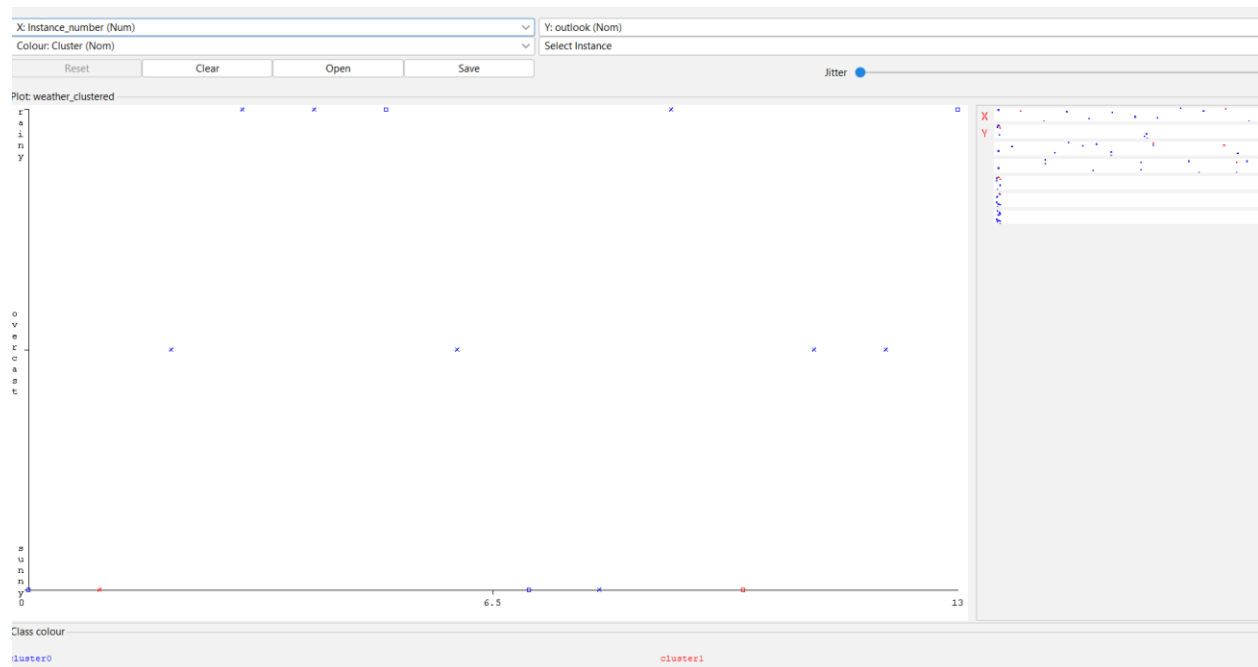
Class attribute: play
Classes to Clusters:

0 1 <-- assigned to cluster
8 1 | yes
4 1 | no

Cluster 0 <-- yes
Cluster 1 <-- no

Incorrectly clustered instances : 5.0 35.7143 %
```

OUTPUT INFORMATION



VISUALISING THE PLOT GRAPH OF CLUSTER ASSIGNMENTS

OUTPUT INFORMATION USING WEATHER NOMINAL DATA

15. EVALUATING THE ACCURACY OF THE CLASSIFIERS

Preprocess Classify Cluster Associate Select attributes Visualize

Classifier

Choose J48 -C 0.25 -M 2

Test options

☐ Use training set

☐ Supplied test set Set...

☒ Cross-validation Folds 10

☐ Percentage split % 66

More options...

(Nom) class

Start Stop

Result list (right-click for options)

11:26:10 - trees.J48

Classifier output

```
| petalwidth > 1.7: Iris-versicolor (3.0/1.0)
| petalwidth > 1.7: Iris-virginica (46.0/1.0)

Number of Leaves : 5
Size of the tree : 9

Time taken to build model: 0.01 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances 144 96 %
Incorrectly Classified Instances 6 4 %
Kappa statistic 0.94
Mean absolute error 0.035
Root mean squared error 0.1586
Relative absolute error 7.8705 %
Root relative squared error 33.6353 %
Total Number of Instances 150

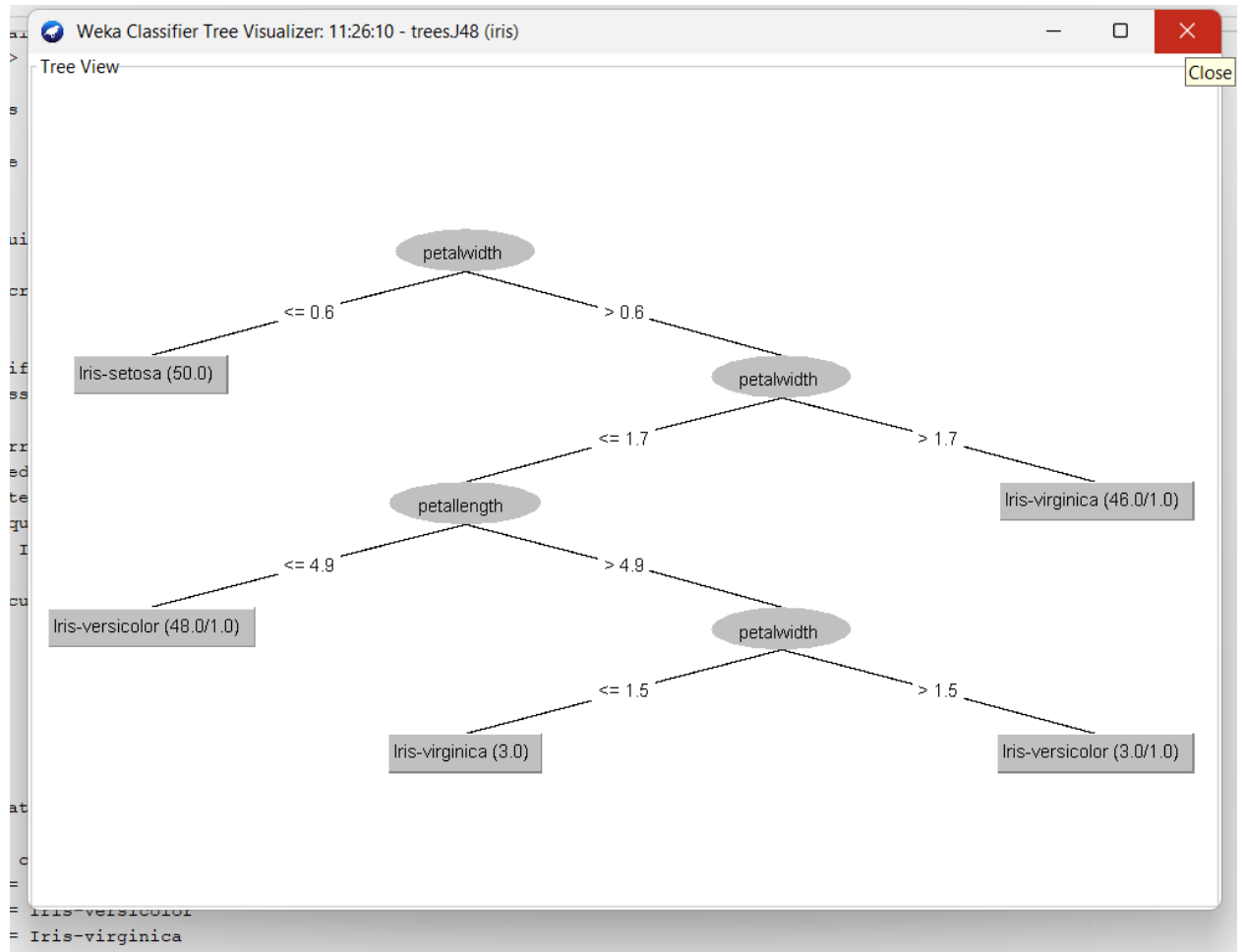
=== Detailed Accuracy By Class ===

      TP Rate  FP Rate  Precision  Recall  F-Measure  MCC  ROC Area  PRC Area  Class
      0.980    0.000    1.000    0.980    0.990    0.985    0.990    0.987    Iris-setosa
      0.940    0.030    0.940    0.940    0.940    0.910    0.952    0.880    Iris-versicolor
      0.960    0.030    0.941    0.960    0.950    0.925    0.961    0.905    Iris-virginica
Weighted Avg. 0.960    0.020    0.960    0.960    0.960    0.940    0.968    0.924

=== Confusion Matrix ===

 a b c <-- classified as
49 1 0 | a = Iris-setosa
 0 47 3 | b = Iris-versicolor
 0 2 48 | c = Iris-virginica
```

OUTPUT INFORMATION OF IRIS IN ACCURACY FINDING.



TREE DIAGRAM OF ACCURACY WITH IRIS INFO IN WEKA



ERROR ACCURACY RATE OF IRIS IN WEKA.

1. CREATE THE ARFF FILE FOR THE DIABETES DATABASE AND PERFORM THE RULE BASED CLASSIFICATION.

Preprocess **Classify** Cluster Associate Select attributes Visualize

Classifier

Choose **J48 -C 0.25 -M 2**

Test options

☐ Use training set

☐ Supplied test set

☒ Cross-validation Folds

☐ Percentage split %

(Nom) class

Result list (right-click for options)

09:43:25 - functions.Logistic

09:47:10 - trees.J48

Classifier output

Number of Leaves : 20

Size of the tree : 39

Time taken to build model: 0.04 seconds

=== Stratified cross-validation ===

=== Summary ===

Correctly Classified Instances	567	73.8281 %
Incorrectly Classified Instances	201	26.1719 %
Kappa statistic	0.4164	
Mean absolute error	0.3158	
Root mean squared error	0.4463	
Relative absolute error	69.4841 %	
Root relative squared error	93.6293 %	
Total Number of Instances	768	

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
	0.814	0.403	0.790	0.814	0.802	0.417	0.751	0.811	tested_r
	0.597	0.186	0.632	0.597	0.614	0.417	0.751	0.572	tested_f
Weighted Avg.	0.738	0.327	0.735	0.738	0.736	0.417	0.751	0.727	


=== Confusion Matrix ===

```

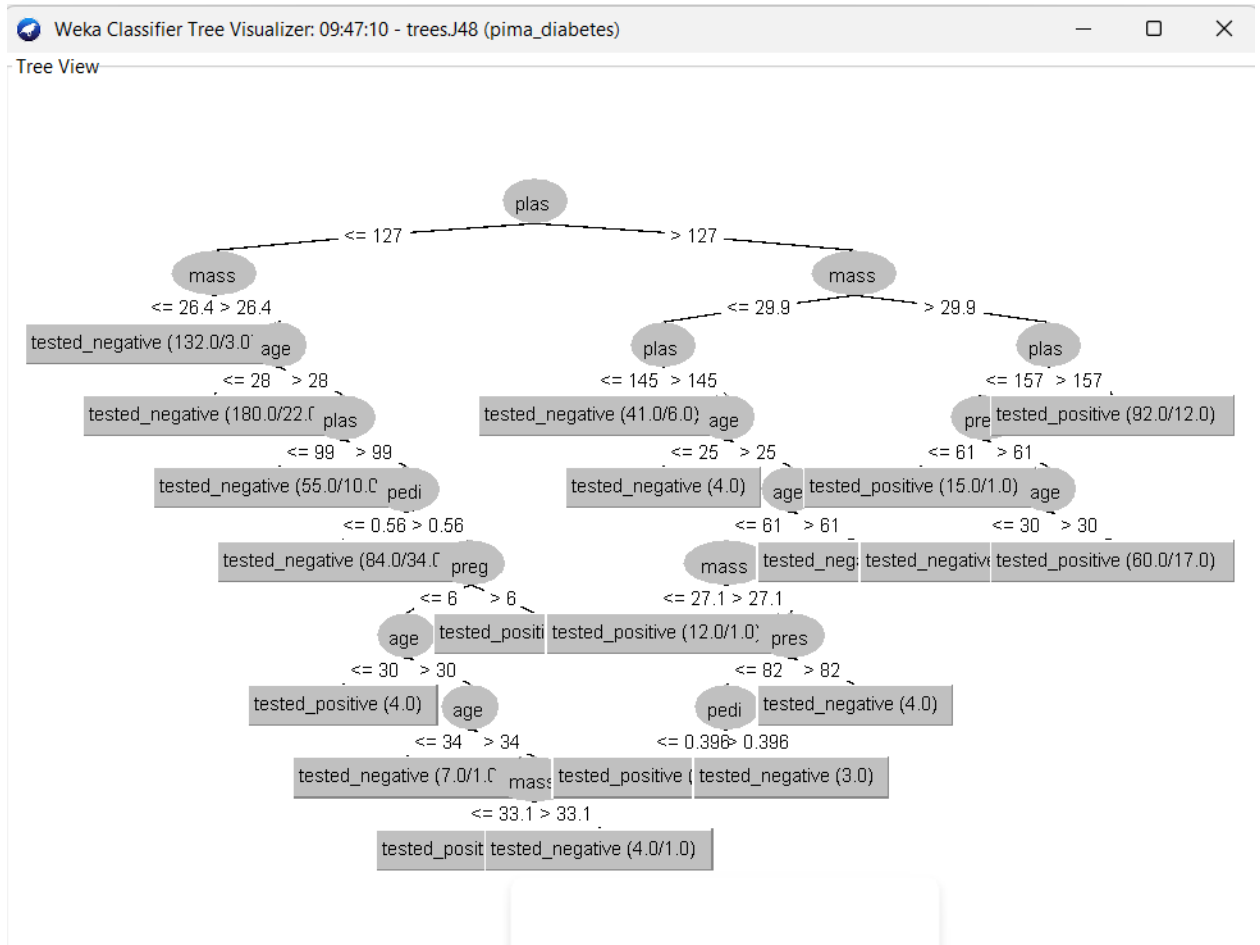
a   b   <-- classified as
407  93 |   a = tested_negative
108 160 |   b = tested_positive

```

Status
OK

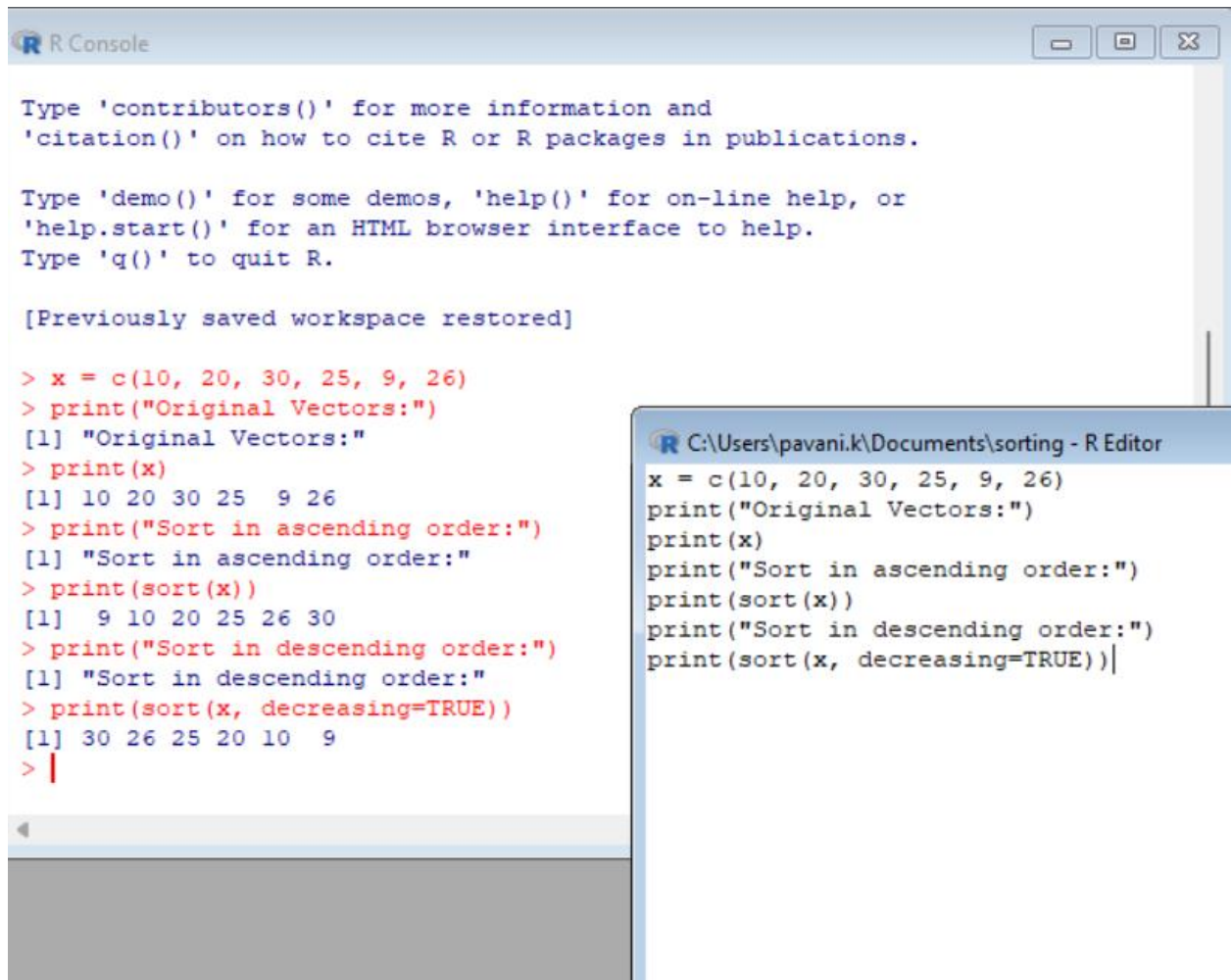


Output info of diabetes by rule based classification.



Tree view of diabetes on rule based classification.

*. R PROGRAM FOR SORTING VECTOR



The image shows two windows from the R environment. The top window is the 'R Console', which displays help text about 'contributors()', 'citation()', 'demo()', 'help()', 'help.start()', and 'q()', followed by a message '[Previously saved workspace restored]'. Below this, it shows the execution of R code for sorting a vector. The bottom window is the 'R Editor', titled 'C:\Users\pavani.k\Documents\sorting - R Editor', which contains the same R code as the console.

```
R Console

Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

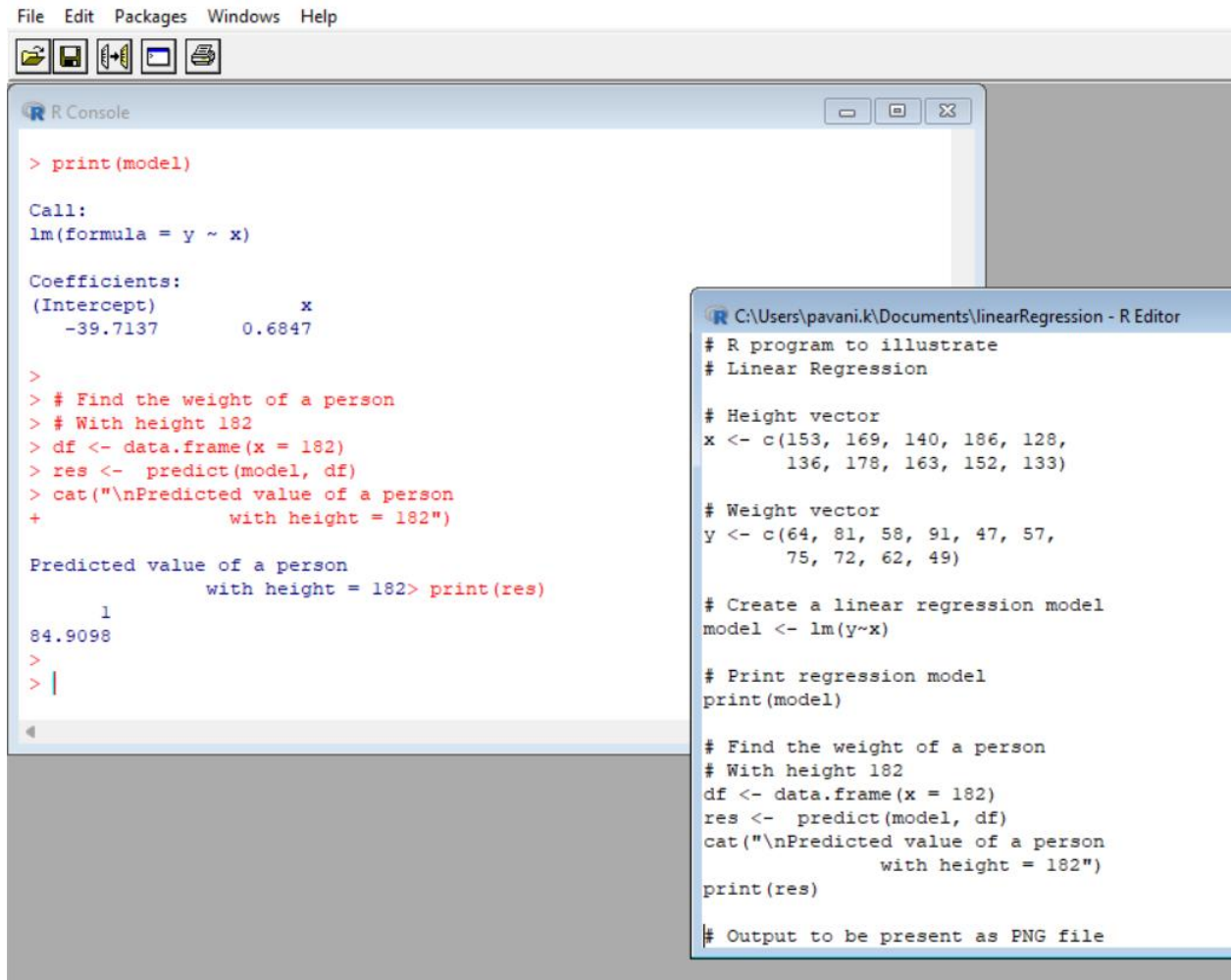
Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

[Previously saved workspace restored]

> x = c(10, 20, 30, 25, 9, 26)
> print("Original Vectors:")
[1] "Original Vectors:"
> print(x)
[1] 10 20 30 25 9 26
> print("Sort in ascending order:")
[1] "Sort in ascending order:"
> print(sort(x))
[1] 9 10 20 25 26 30
> print("Sort in descending order:")
[1] "Sort in descending order:"
> print(sort(x, decreasing=TRUE))
[1] 30 26 25 20 10 9
> |

R C:\Users\pavani.k\Documents\sorting - R Editor
x = c(10, 20, 30, 25, 9, 26)
print("Original Vectors:")
print(x)
print("Sort in ascending order:")
print(sort(x))
print("Sort in descending order:")
print(sort(x, decreasing=TRUE))|
```

2.R PROGRAM FOR LINEAR REGRESSION.



The screenshot displays the R Studio environment. The top menu bar includes 'File', 'Edit', 'Packages', 'Windows', and 'Help'. Below the menu is a toolbar with icons for file operations. The main workspace is divided into two panes. The left pane, titled 'R Console', shows the execution of an R script. The right pane, titled 'C:\Users\pavani.k\Documents\linearRegression - R Editor', shows the source code of the script.

R Console Output:

```
> print(model)

Call:
lm(formula = y ~ x)

Coefficients:
(Intercept)          x
   -39.7137       0.6847

>
> # Find the weight of a person
> # With height 182
> df <- data.frame(x = 182)
> res <- predict(model, df)
> cat("\nPredicted value of a person
+       with height = 182")

Predicted value of a person
with height = 182> print(res)

1
84.9098
>
> |
```

R Editor Code:

```
# R program to illustrate
# Linear Regression

# Height vector
x <- c(153, 169, 140, 186, 128,
      136, 178, 163, 152, 133)

# Weight vector
y <- c(64, 81, 58, 91, 47, 57,
      75, 72, 62, 49)

# Create a linear regression model
model <- lm(y~x)

# Print regression model
print(model)

# Find the weight of a person
# With height 182
df <- data.frame(x = 182)
res <- predict(model, df)
cat("\nPredicted value of a person
      with height = 182")
print(res)

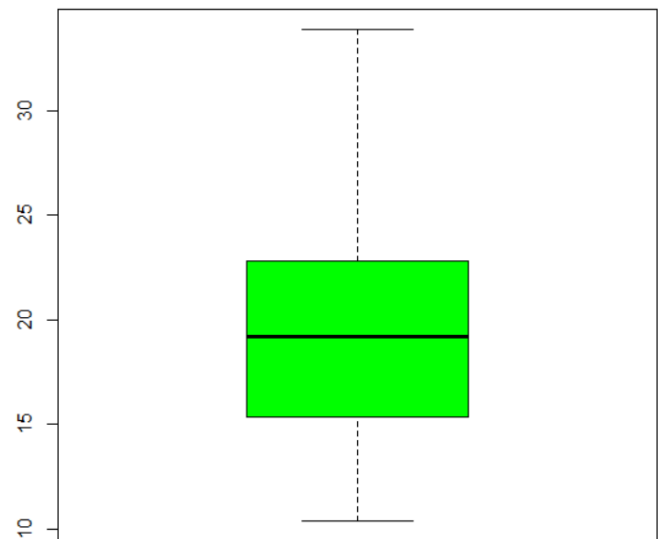
# Output to be present as PNG file
```

3.PLOTTING GRAPH

```

R C:\Users\pavani.k\Documents\plotting graph - R Editor
boxplot(mtcars$mpg, col="green")

```



4.

A) CENTRAL TENDENCY—MEAN:

```

R Console
R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help,
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

[Previously saved workspace restored]

> # Defining vector
> x <- c(1, 5, 8, 10)
>
> # Print Harmonic Mean
> print(1 / mean(1 / x))
[1] 0.82719

C:\Users\pavani.k\Documents\central tendency - R Editor
###mean::
# Defining vector
x <- c(1, 5, 8, 10)

# Print Harmonic Mean
print(1 / mean(1 / x))

####median::
# Defining vector
x <- c(3, 7, 5, 13, 20, 23, 39,
      23, 40, 23, 14, 12, 56, 23)

# Print Median
median(x)

```

B) CENTRAL TENDENCY –MEDIAN:

help.start()' for an HTML browser interface to help.
type 'q()' to quit R.

[Previously saved workspace restored]

```
# Defining vector
x <- c(1, 5, 8, 10)

# Print Harmonic Mean
print(1 / mean(1 / x))
[1] 2.807018

# Defining vector
x <- c(3, 7, 5, 13, 20, 23, 39,
      23, 40, 23, 14, 12, 56, 23)

# Print Median
median(x)
[1] 21.5
|
```

C:\Users\pavani.k\Documents\central tendency - R Editor

```
###mean::
# Defining vector
x <- c(1, 5, 8, 10)

# Print Harmonic Mean
print(1 / mean(1 / x))

####median::
# Defining vector
x <- c(3, 7, 5, 13, 20, 23, 39,
      23, 40, 23, 14, 12, 56, 23)

# Print Median
median(x)
|

###mode::
# Defining vector
x <- c(3, 7, 5, 13, 20, 23, 39,
      23, 40, 23, 14, 12, 56,
      23, 29, 56, 37, 45, 1, 25, 8)

# Generate frequency table
y <- table(x)

# Print frequency table
print(y)
```

C) CENTRAL TENDENCY—MODE:

```
> # Generate frequency table
> y <- table(x)
>
> # Print frequency table
> print(y)
x
 1  3  5  7  8 12 13 14 20 23 25 29 37 39 40 45 56
1  1  1  1  1  1  1  1  1  4  1  1  1  1  1  1  2
> |
```

```
x <- c(3, 7, 5, 13, 20, 23, 39,
      23, 40, 23, 14, 12, 56,
      23, 29, 56, 37, 45, 1, 2)

# Print Median
median(x)

###mode::
# Defining vector
x <- c(3, 7, 5, 13, 20, 23, 39,
      23, 40, 23, 14, 12, 56,
      23, 29, 56, 37, 45, 1, 2)

# Generate frequency table
y <- table(x)

# Print frequency table
print(y)
```

5.NORMALISATION AND ANALYSIS:

```
#apply Min-Max normalization to first four columns in iris data:  
iris_norm <- as.data.frame(lapply(iris[1:4], min_max_norm))
```

```
#view first six rows of normalized iris dataset
```

```
head(iris_norm)
```

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
0.22222222	0.6250000	0.06779661	0.04166667
0.16666667	0.4166667	0.06779661	0.04166667
0.11111111	0.5000000	0.05084746	0.04166667
0.08333333	0.4583333	0.08474576	0.04166667
0.19444444	0.6666667	0.06779661	0.04166667
0.30555556	0.7916667	0.11864407	0.12500000

```
R C:\Users\pavani.k\Documents\normalisation - R Editor
```

```
#define Min-Max normalization function
```

```
min_max_norm <- function(x) {  
  (x - min(x)) / (max(x) - min(x))  
}
```

```
#apply Min-Max normalization to first four columns in iris data:
```

```
iris_norm <- as.data.frame(lapply(iris[1:4], min_max_norm))
```

```
#view first six rows of normalized iris dataset
```

```
head(iris_norm)
```

6. REGRESSION:

```
R Console
> # Generate vector with pass and fail values of 40 students
> result <- c(0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1,
+ 1, 0, 0, 0, 1, 1, 0, 0, 1, 0,
+ 0, 0, 1, 0, 0, 1, 1, 0, 1, 1,
+ 1, 1, 1, 0, 1, 1, 1, 1, 0, 1)
>
> # Data Frame
> df <- as.data.frame(cbind(IQ, result))
>
> # Print data frame
> print(df)
      IQ result
1 26.88303     0
2 26.99592     0
3 27.27634     0
4 27.39254     1
5 27.40339     0
6 27.49747     0
7 27.66223     0
8 27.93250     0
9 28.18333     0
10 28.19356     1
11 28.23130     1

C:\Users\pavani.k\Documents\regression - R Editor
# Generate random IQ values with mean = 30 and sd =2
IQ <- rnorm(40, 30, 2)

# Sorting IQ level in ascending order
IQ <- sort(IQ)

# Generate vector with pass and fail values of 40 students
result <- c(0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1,
1, 0, 0, 0, 1, 1, 0, 0, 1, 0,
0, 0, 1, 0, 0, 1, 1, 0, 1, 1,
1, 1, 1, 0, 1, 1, 1, 1, 0, 1)

# Data Frame
df <- as.data.frame(cbind(IQ, result))

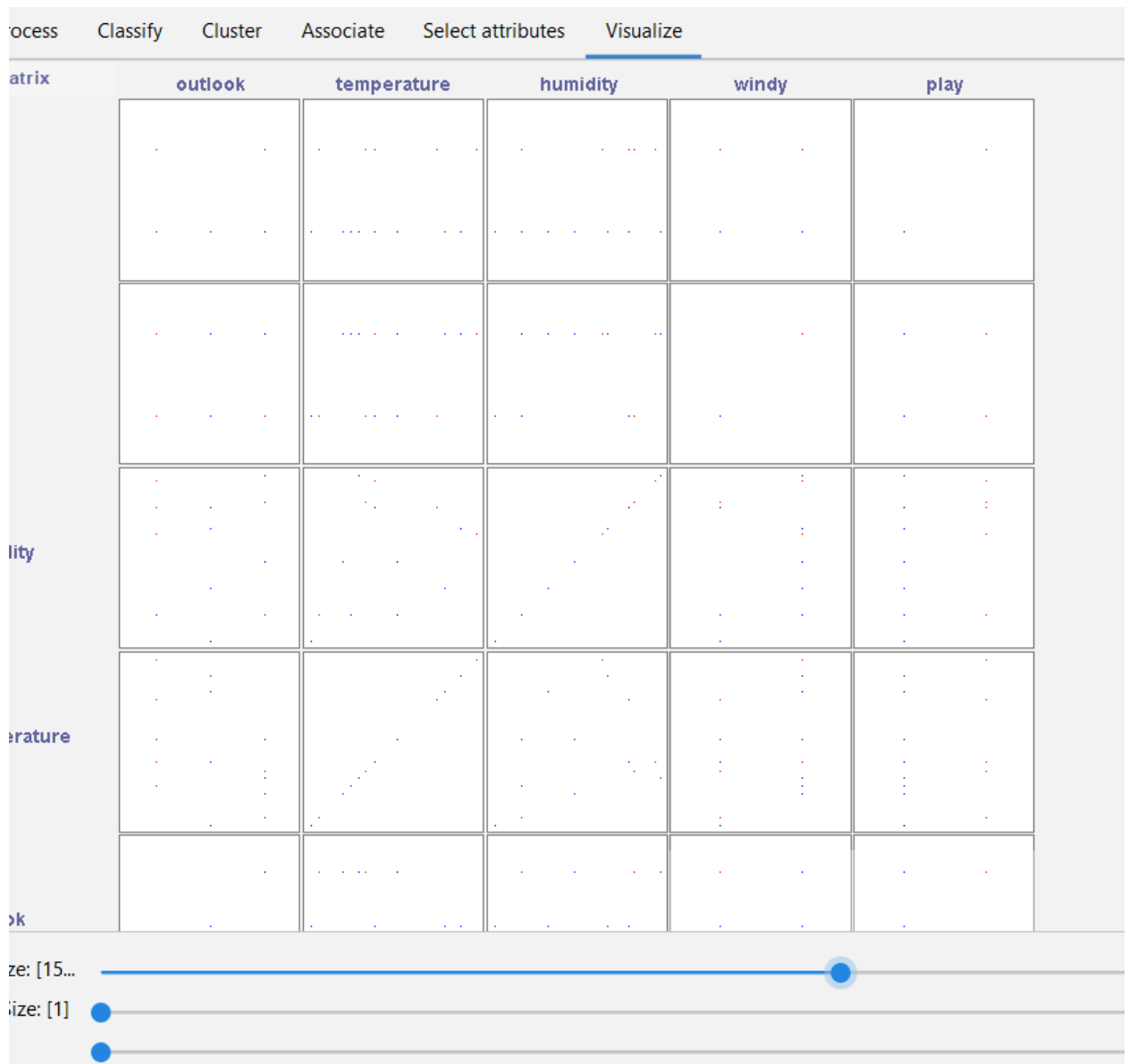
# Print data frame
print(df)

# output to be present as PNG file
png(file="LogisticRegressionGFG.png")

# Plotting IQ on x-axis and result on y-axis
plot(IQ, result, xlab = "IQ Level",
ylab = "Probability of Passing")

# Create a logistic model
```

12.FP GROWTH



FP GROWTH VISUALISATION

th -P 2 -I 1 -N 10 -T 0 -C 0.9 -D 0.05 -U 1.0 -M 0.1

```
Associator output
temperature
humidity
windy
play
=== Associator model (full training set) ===

Apriori
=====

Minimum support: 0.15 (2 instances)
Minimum metric <confidence>: 0.9
Number of cycles performed: 17

Generated sets of large itemsets:

Size of set of large itemsets L(1): 12

Size of set of large itemsets L(2): 47

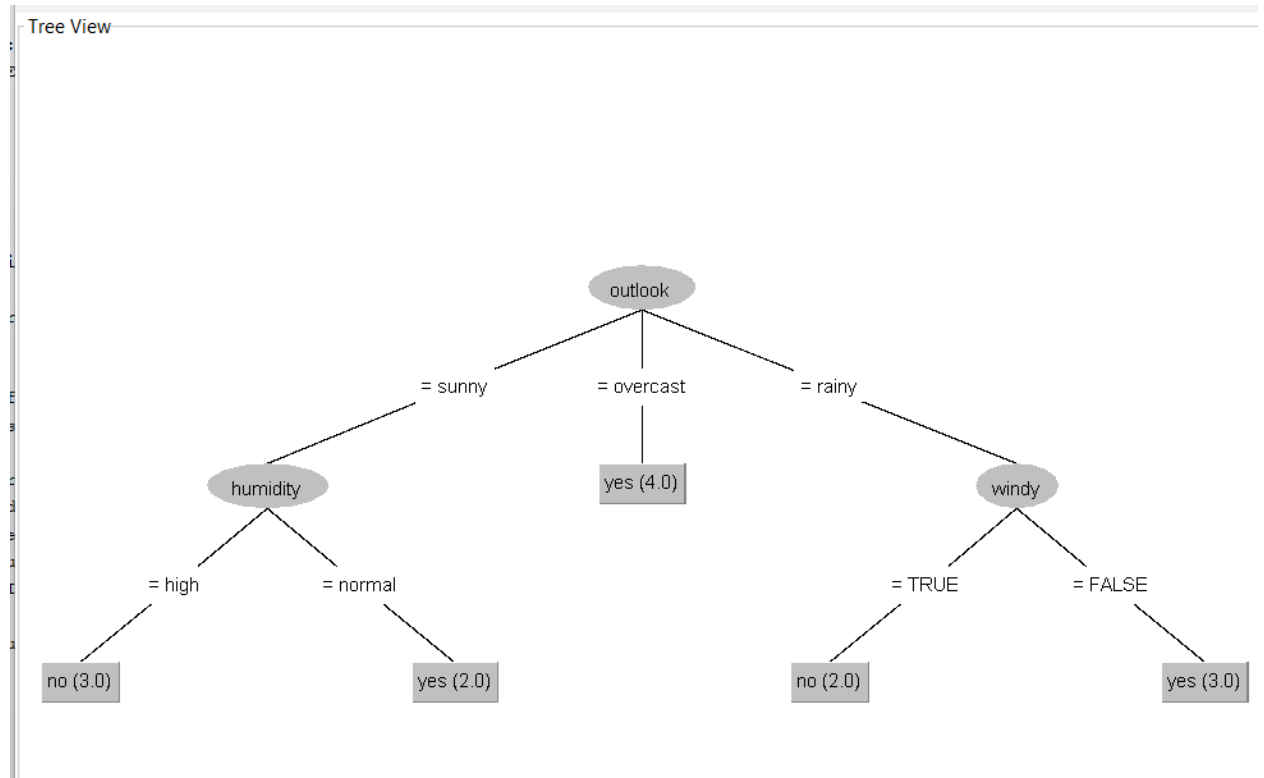
Size of set of large itemsets L(3): 39

Size of set of large itemsets L(4): 6

Best rules found:

1. outlook=overcast 4 ==> play=yes 4    <conf:(1)> lift:(1.56) lev:(0.1) [1] conv:(1.43)
2. temperature=cool 4 ==> humidity=normal 4    <conf:(1)> lift:(2) lev:(0.14) [2] conv:(2)
3. humidity=normal windy=FALSE 4 ==> play=yes 4    <conf:(1)> lift:(1.56) lev:(0.1) [1] conv:(1.43)
4. outlook=sunny play=no 3 ==> humidity=high 3    <conf:(1)> lift:(2) lev:(0.11) [1] conv:(1.5)
5. outlook=sunny humidity=high 3 ==> play=no 3    <conf:(1)> lift:(2.8) lev:(0.14) [1] conv:(1.93)
6. outlook=rainy play=yes 3 ==> windy=FALSE 3    <conf:(1)> lift:(1.75) lev:(0.09) [1] conv:(1.29)
7. outlook=rainy windy=FALSE 3 ==> play=yes 3    <conf:(1)> lift:(1.56) lev:(0.08) [1] conv:(1.07)
8. temperature=cool play=yes 3 ==> humidity=normal 3    <conf:(1)> lift:(2) lev:(0.11) [1] conv:(1.5)
9. outlook=sunny temperature=hot 2 ==> humidity=high 2    <conf:(1)> lift:(2) lev:(0.07) [1] conv:(1)
10. temperature=hot play=no 2 ==> outlook=sunny 2    <conf:(1)> lift:(2.8) lev:(0.09) [1] conv:(1.29)
```

13.DECISION TREE



TREE VISUALISATION

eprocess

Classify

Cluster

Associate

Select attributes

Visualize

Classifier

ChooseJ48 -C 0.25 -M 2

Options

Use training set

Supplied test set

Cross-validation Folds10

Percentage split %66

More options...

Start

Stop

Result list (right-click for options)

0:20:10 - trees.J48

Classifier output

outlook = Overcast: yes (1.0)

outlook = rainy

| windy = TRUE: no (2.0)

| windy = FALSE: yes (3.0)

Number of Leaves : 5

Size of the tree : 8

Time taken to build model: 0 seconds

=== Stratified cross-validation ===

=== Summary ===

Correctly Classified Instances750%

Incorrectly Classified Instances750%

Kappa statistic-0.0426

Mean absolute error0.4167

Root mean squared error0.5984

Relative absolute error87.5%

Root relative squared error121.2987%

Total Number of Instances14

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
	0.556	0.600	0.625	0.556	0.588	-0.043	0.633	0.758	yes
	0.400	0.444	0.333	0.400	0.364	-0.043	0.633	0.457	no
Weighted Avg.	0.500	0.544	0.521	0.500	0.508	-0.043	0.633	0.650	

=== Confusion Matrix ===

a b <-- classified as

5 4 | a = yes

3 2 | b = no

status

<

OUTPUT INFO.

16. CREATE THE ARFF FILE FOR THE DIABETES DATABASE AND PERFORM SVM BASED CLASSIFICATION

eprocess **Classify** Cluster Associate Select attributes Visualize

Classifier

Choose **Logistic -R 1.0E-8 -M -1 -num-decimal-places 4**

Options

Use training set

Supplied test set

Cross-validation Folds

Percentage split %

om) class

ult list (right-click for options)

Classifier output

```

66 2:tested_positive 2:tested_positive 0.864
67 2:tested_positive 2:tested_positive 0.863
68 2:tested_positive 2:tested_positive 0.786
69 2:tested_positive 2:tested_positive 0.666
70 2:tested_positive 2:tested_positive 0.514
71 2:tested_positive 1:tested_negative + 0.785
72 2:tested_positive 2:tested_positive 0.815
73 2:tested_positive 1:tested_negative + 0.652
74 2:tested_positive 1:tested_negative + 0.52
75 2:tested_positive 1:tested_negative + 0.634
76 2:tested_positive 1:tested_negative + 0.851

=== Stratified cross-validation ===
=== Summary ===
Correctly Classified Instances      597      77.7344 %
Incorrectly Classified Instances    171      22.2656 %
Kappa statistic                    0.4855
Mean absolute error                 0.319
Root mean squared error             0.3996
Relative absolute error             70.1878 %
Root relative squared error        83.8438 %
Total Number of Instances          768

=== Detailed Accuracy By Class ===
                TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                0.884    0.422    0.796     0.884    0.838     0.492    0.825    0.886    tested_negative
                0.578    0.116    0.728     0.578    0.644     0.492    0.825    0.697    tested_positive
Weighted Avg.   0.777    0.315    0.772     0.777    0.770     0.492    0.825    0.820

=== Confusion Matrix ===
  a  b  <-- classified as
442 58 |  a = tested_negative
113 155 | b = tested_positive

```

Output information of data diabetes with logistics.

14. PREDICTION OF CATEGORICAL DATA USING SMO ALGORITHM USING WEKA.

Preprocess Classify Cluster Associate Select attributes Visualize

Classifier

Choose **SMO** -C 1.0 -L 0.001 -P 1.0E-12 -N 0 -V -1 -W 1 -K "weka.classifiers.functions.supportVector.PolyKernel -E 1.0 -C 250007" -calibrator "weka.classifiers.functions.Logistic -R 1.0E-8 -M -1 -num-

Test options

☐ Use training set

☐ Supplied test set

☒ Cross-validation Folds

☐ Percentage split %

(Nom) class

Result list (right-click for options)

17:34:24 - functions.SMO

Classifier output

```

67 2:tested_positive 2:tested_positive 1
68 2:tested_positive 2:tested_positive 1
69 2:tested_positive 2:tested_positive 1
70 2:tested_positive 1:tested_negative + 1
71 2:tested_positive 1:tested_negative + 1
72 2:tested_positive 2:tested_positive 1
73 2:tested_positive 1:tested_negative + 1
74 2:tested_positive 1:tested_negative + 1
75 2:tested_positive 1:tested_negative + 1
76 2:tested_positive 1:tested_negative + 1

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances      590           76.8229 %
Incorrectly Classified Instances    178           23.1771 %
Kappa statistic                    0.453
Mean absolute error                 0.2318
Root mean squared error             0.4814
Relative absolute error             50.994 %
Root relative squared error        101.0033 %
Total Number of Instances          768

=== Detailed Accuracy By Class ===

          TP Rate  FP Rate  Precision  Recall   F-Measure  MOC       ROC Area  PRC Area  Class
          0.900    0.478    0.779     0.900    0.835      0.467    0.711    0.766    tested_negative
          0.522    0.100    0.737     0.522    0.611      0.467    0.711    0.552    tested_positive
Weighted Avg.   0.768    0.346    0.764     0.768    0.757      0.467    0.711    0.691

=== Confusion Matrix ===

  a   b   <-- classified as
450  50 |   a = tested_negative
128 140 |   b = tested_positive

```

The output information of diabetes data using smo..

17. PREDICTION OF CATEGORICAL DATA USING BAYESIAN ALGORITHM USING WEKA

Weka Explorer

Preprocess Classify Cluster Associate Select attributes Visualize

Classifier: Choose **NaiveBayes**

Test options

- ☐ Use training set
- ☐ Supplied test set
- ☒ Cross-validation Folds
- ☐ Percentage split %

(Nom) play

Result list (right-click for options)

17:49:10 - bayes.NaiveBayes

Classifier output

```

TRUE      4.0    4.0
FALSE     7.0    3.0
[total]   11.0   7.0

Time taken to build model: 0 seconds

=== Stratified cross-validation ===
=== Summary ===


Correctly Classified Instances      9      64.2857 %
Incorrectly Classified Instances    5      35.7143 %
Kappa statistic                    0.1026
Mean absolute error                 0.4649
Root mean squared error             0.543
Relative absolute error             97.6254 %
Root relative squared error         110.051 %
Total Number of Instances          14

=== Detailed Accuracy By Class ===

          TP Rate  FP Rate  Precision  Recall  F-Measure  MCC   ROC Area  PRC Area  Class
          0.889   0.800   0.667     0.889   0.762     0.122  0.444    0.633    yes
          0.200   0.111   0.500     0.200   0.286     0.122  0.444    0.397    no
Weighted Avg.   0.643   0.554   0.607     0.643   0.592     0.122  0.444    0.548

=== Confusion Matrix ===

a b  <-- classified as
8 1 | a = yes
4 1 | b = no
    
```

Status: OK  x0

OUTPUT INFORMATION WEATHER NUMERIC BASED ON NAVIES BAYES RULE..

18. DATA ANALYSIS BY DENSITY BASED CLUSTERING ALGORITHM USING WEKA.

Weka Explorer

Preprocess Classify **Cluster** Associate Select attributes Visualize

Clusterer

Choose **EM** -I 100 -N -1 -X 10 -max -1 -ll-cv 1.0E-6 -ll-iter 1.0E-6 -M 1.0E-6 -K 10 -num-slots 1 -S 100

Cluster mode

☐ Use training set

☐ Supplied test set Set...

☐ Percentage split % 66

☒ **Classes to clusters evaluation**

(Nom) Train on a percentage of the data and cluster the remainder

☒ Store clusters for visualization

Ignore attributes

Start Stop

Result list (right-click for options)

08:53:20 - EM

Clusterer output

high	8
normal	8
[total]	16
windy	
more	7
less	9
[total]	16

Time taken to build model (full training data) : 0.04 seconds

=== Model and evaluation on training set ===

Clustered Instances

0 14 (100%)

Log likelihood: -3.54934

Class attribute: play

Classes to Clusters:

0 <-- assigned to cluster

9 | yes

5 | no

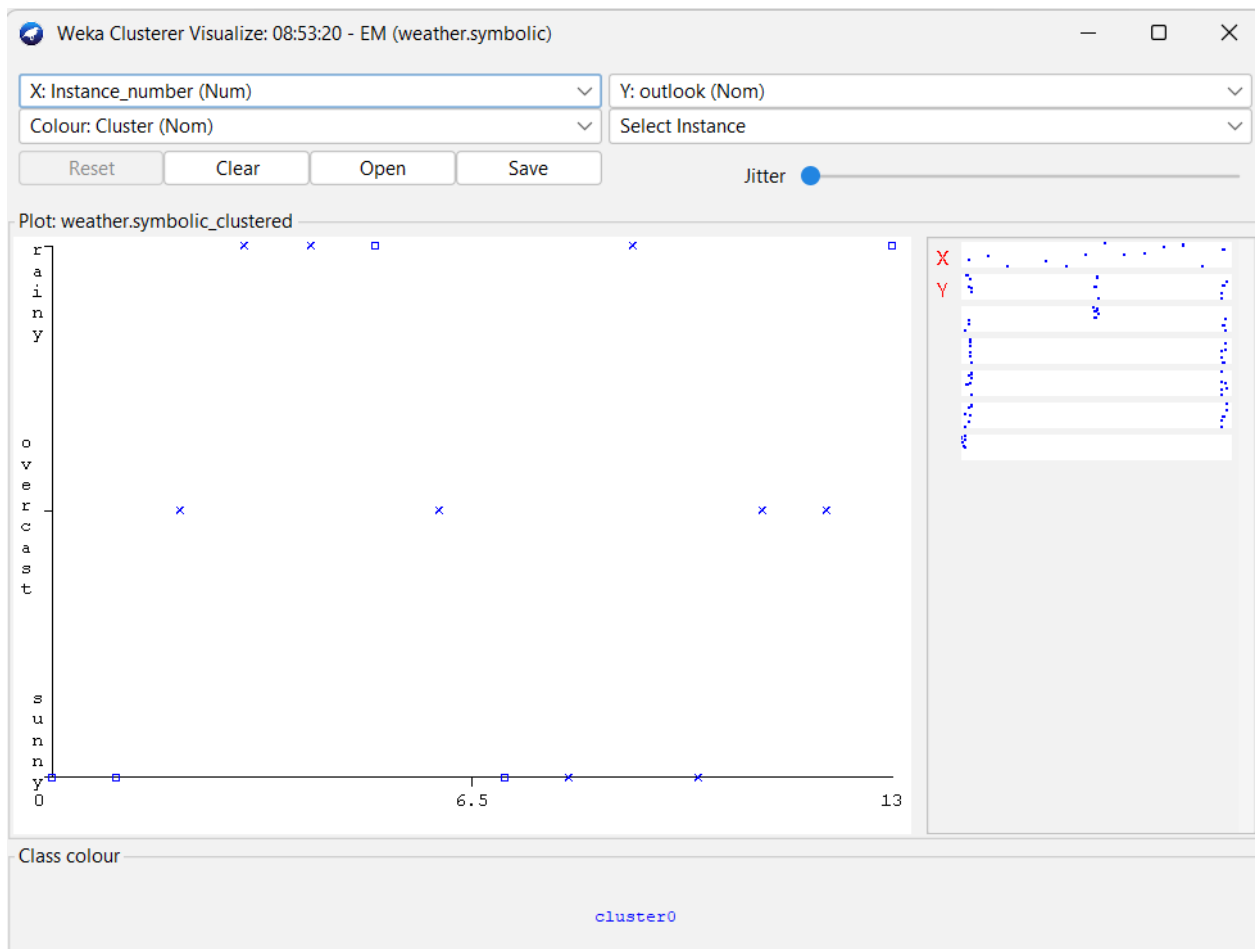
Cluster 0 <-- yes

Incorrectly clustered instances : 5.0 35.7143 %

Status

OK

Log



Output information...

19. CREATE A BOXPLOT GRAPH FOR THE RELATION BETWEEN "MPG"(MILES PER GALLOON) AND "CYL"(NUMBER OF CYLINDERS) FOR THE DATASET "MTCARS" AVAILABLE IN R ENVIRONMENT

```
Untitled - R Editor
#####code for creating boxplots#####
# Give the chart file a name.
png(file = "boxplot.png")
# Plot the chart.
boxplot(mpg ~ cyl, data = mtcars, xlab = "Number of Cylinders",
        ylab = "Miles Per Gallon", main = "Mileage Data")
# Save the file.
dev.off()
```

Code for creating boxplots in r programming.

```
R Console

Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

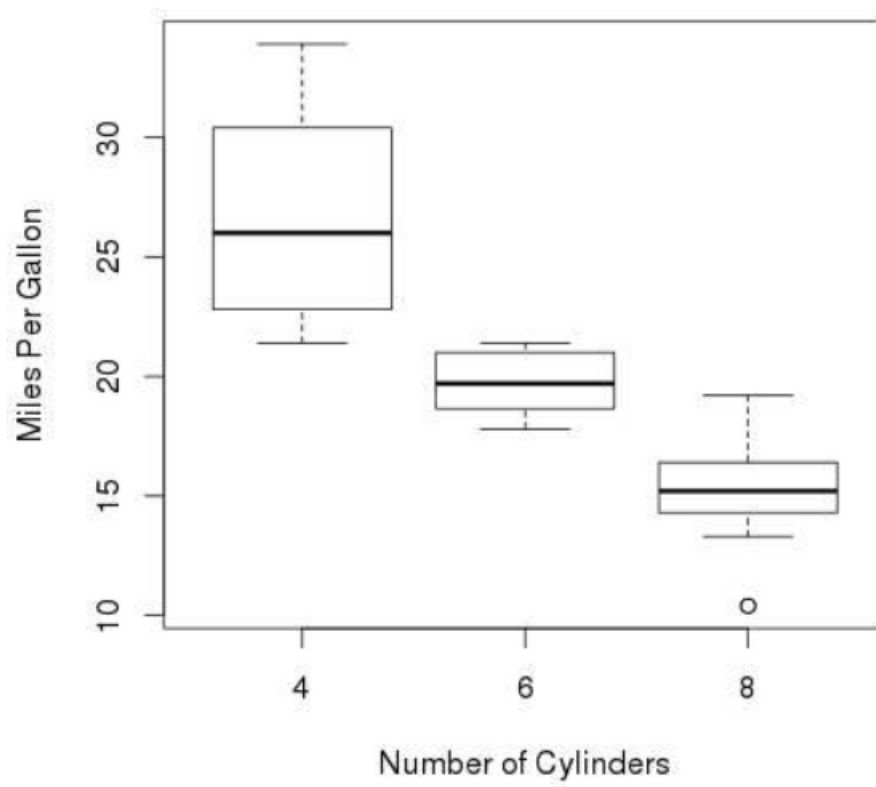
Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

[Previously saved workspace restored]

> # Give the chart file a name.
> png(file = "boxplot.png")
> # Plot the chart.
> boxplot(mpg ~ cyl, data = mtcars, xlab = "Number of Cylinders",
+        ylab = "Miles Per Gallon", main = "Mileage Data")
> # Save the file.
> dev.off()
null device
```

Output result of created boxplot..

Mileage Data

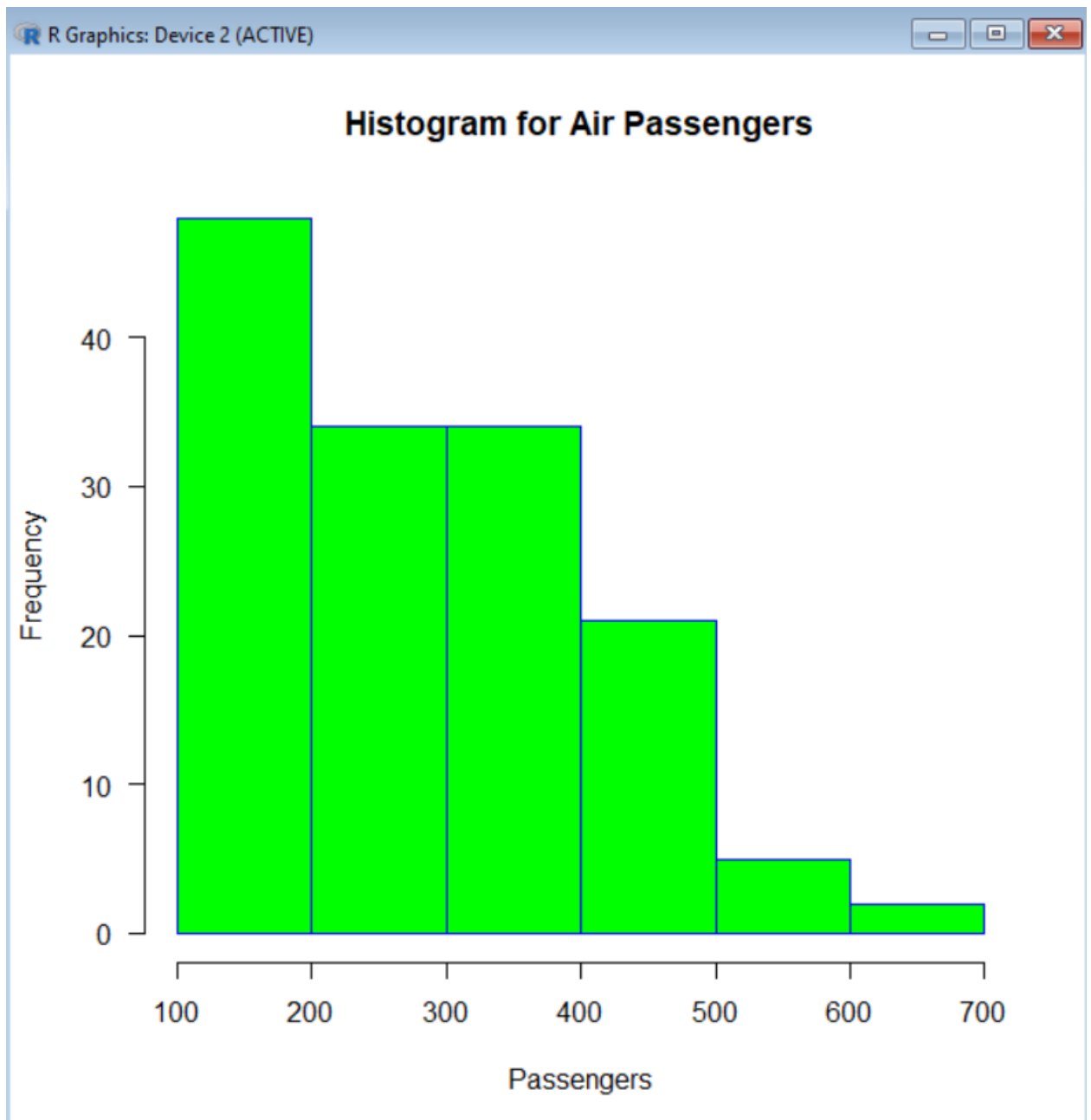


Output graph of boxplot..

22. USING R PROGRAM MAKE A HISTOGRAM FOR THE “AIRPASSENGERS “DATASET, START AT 100 ON THE X-AXIS, AND FROM VALUES 200 TO 700, MAKE THE BINS 150 WIDE

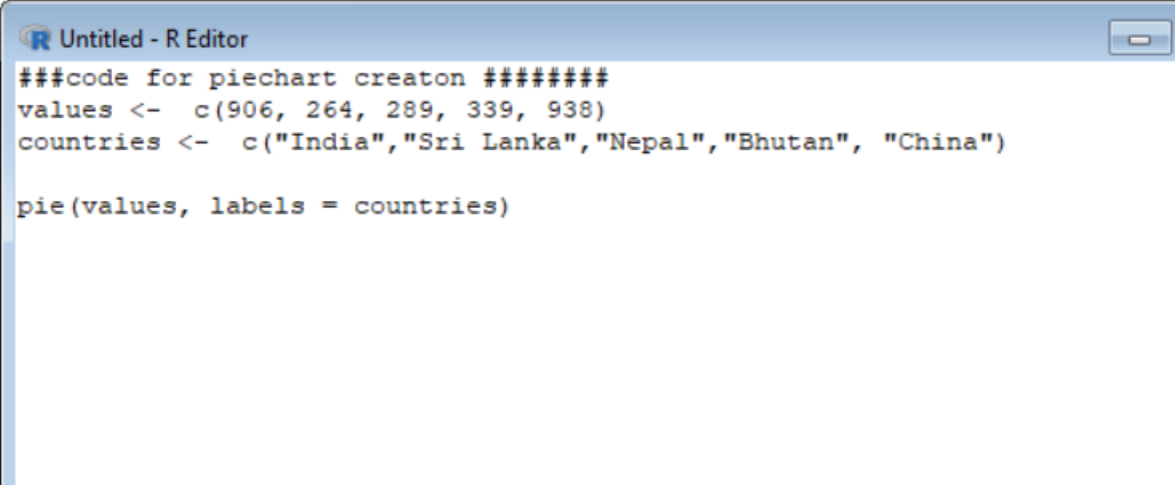
```
Untitled - R Editor
#####code for creating an histogram####
hist(AirPassengers,
      main="Histogram for Air Passengers",
      xlab="Passengers",
      border="blue",
      col="green",
      xlim=c(100,700),
      las=1,
      breaks=5)
```

Code for creating an histogram..



Output of histogram graphical representation..

**23. USING R PROGRAM CREATE A 3D PIE CHART FOR THE DATASET
“POLITICAL KNOWLEDGE” WITH SUITABLE LABELS AND COLOURS.**



```
##code for piechart creaton #####  
values <- c(906, 264, 289, 339, 938)  
countries <- c("India","Sri Lanka","Nepal","Bhutan", "China")  
  
pie(values, labels = countries)
```

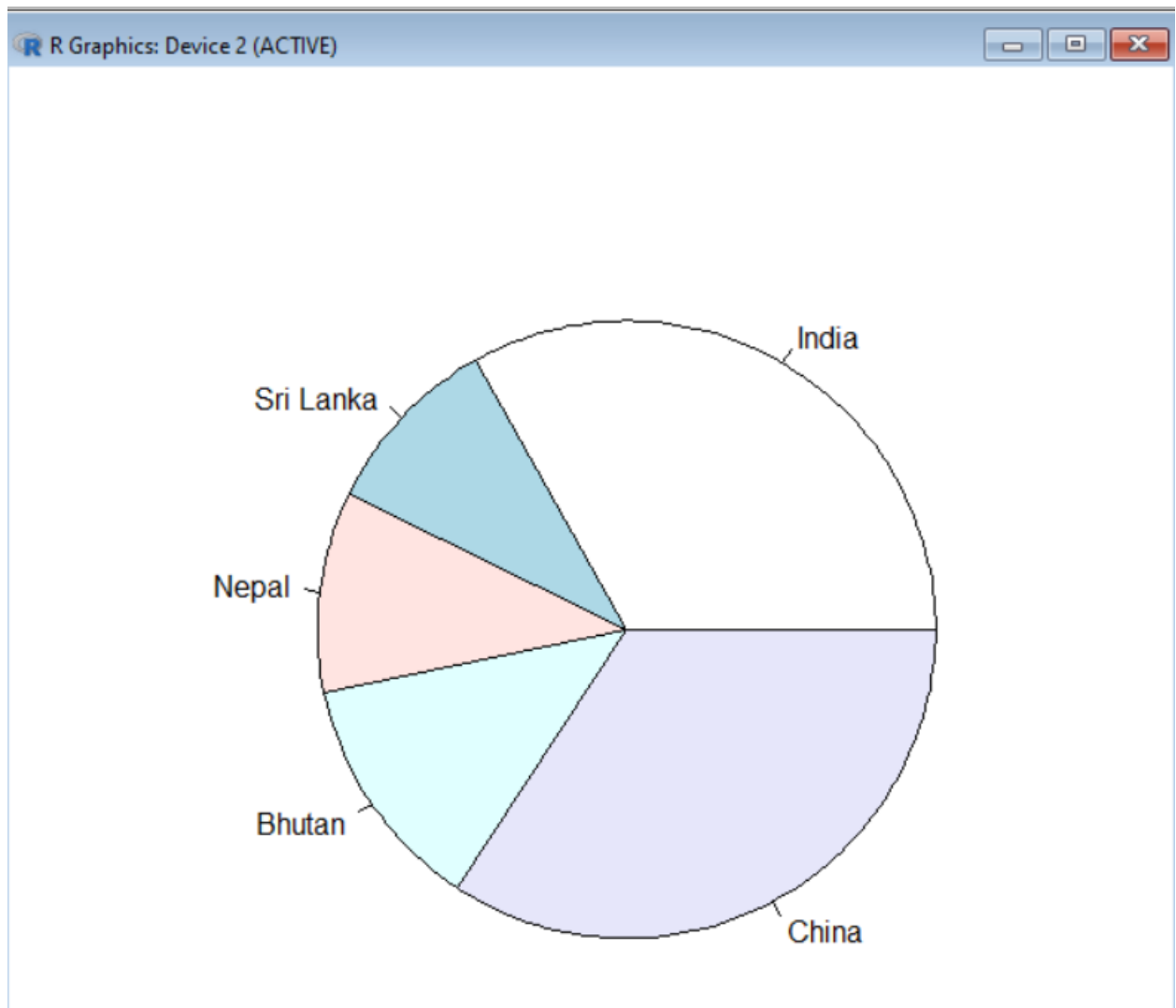
The screenshot shows an R Editor window titled "Untitled - R Editor". The code in the editor is as follows:

```
##code for piechart creaton #####  
values <- c(906, 264, 289, 339, 938)  
countries <- c("India","Sri Lanka","Nepal","Bhutan", "China")  
  
pie(values, labels = countries)
```

Below the code, there is a list of labels for the pie chart segments, each preceded by a red asterisk:

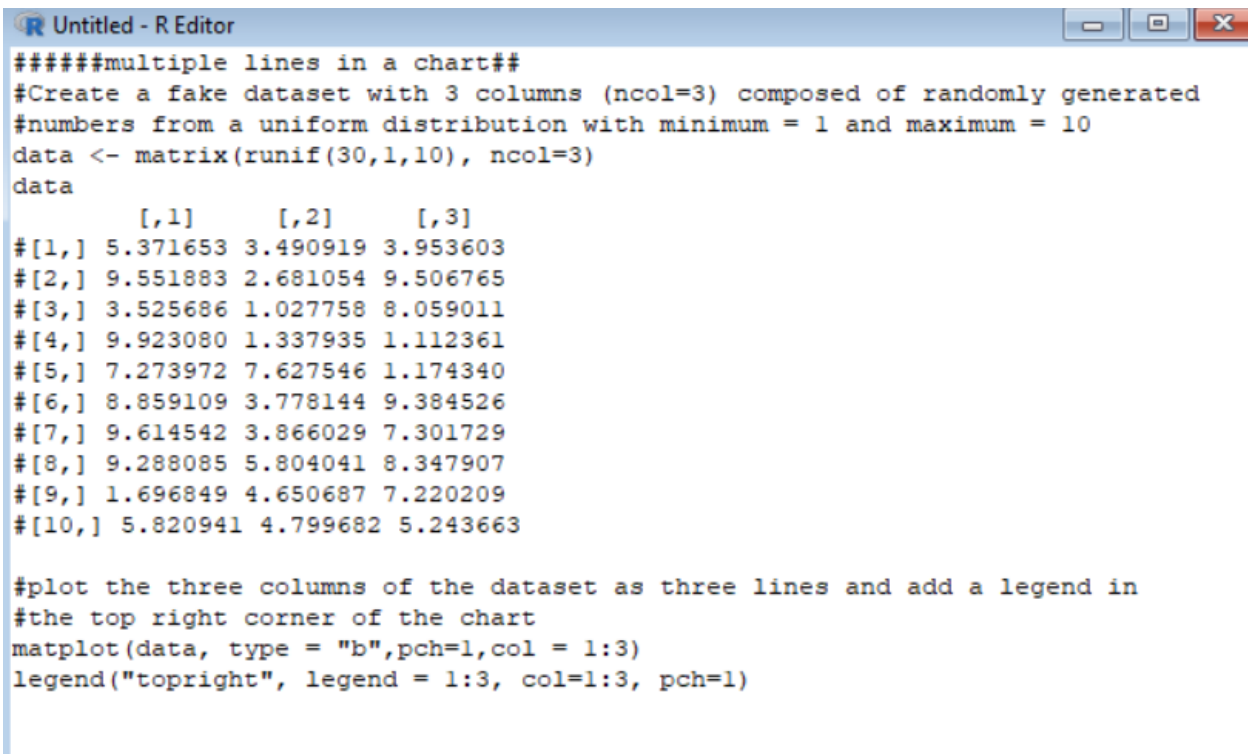
- *Ch
- *289
- *ries
- *Ch

Code for creating the pie charts



Output of created pie chart.

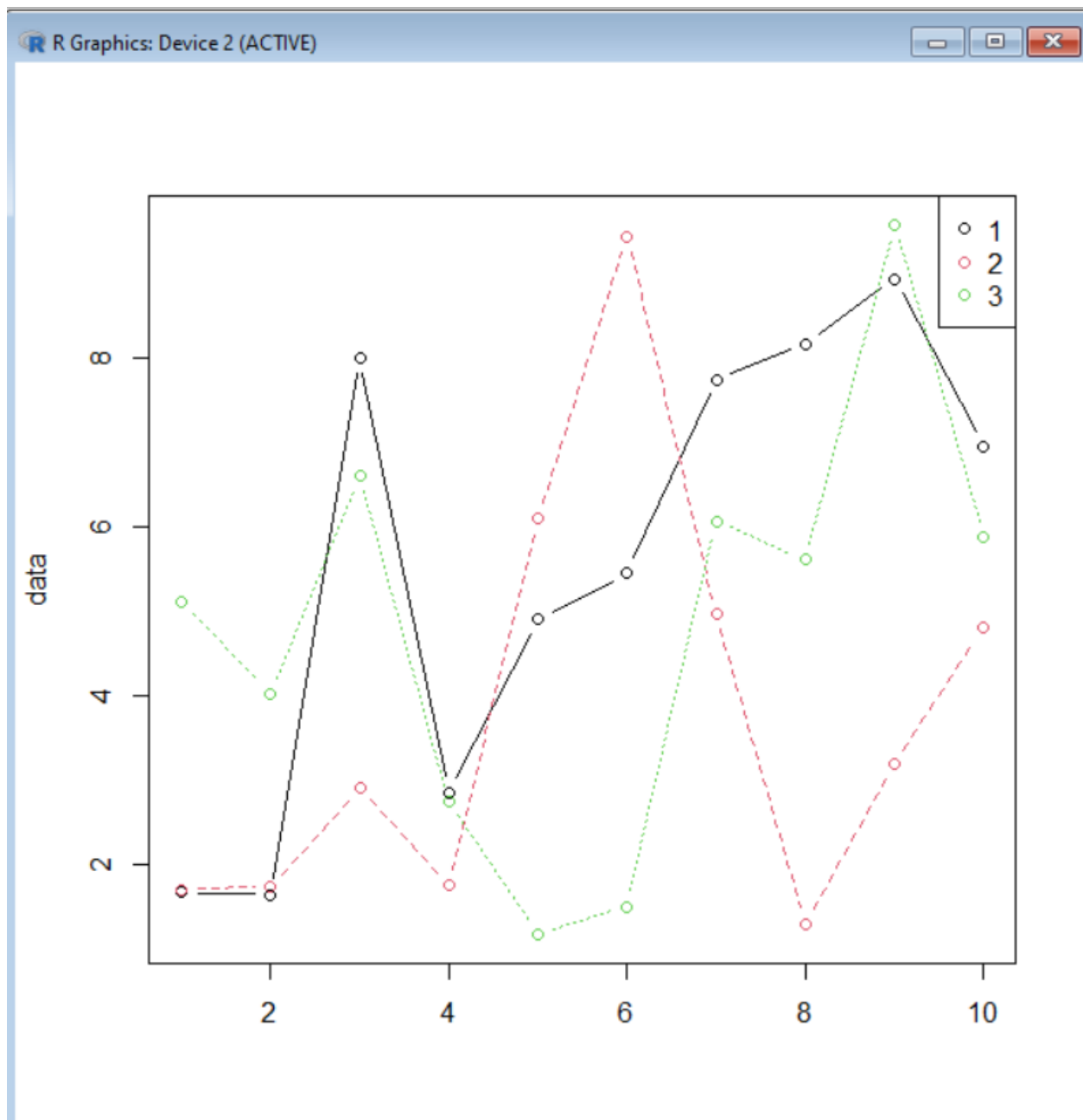
24. OBTAIN MULTIPLE LINES IN LINE CHART USING A SINGLE PLOT FUNCTION IN R.USE ATTRIBUTES “MPG”AND “QSEC” OF THE DATASET “MTCARS”



```
#####multiple lines in a chart##
#Create a fake dataset with 3 columns (ncol=3) composed of randomly generated
#numbers from a uniform distribution with minimum = 1 and maximum = 10
data <- matrix(runif(30,1,10), ncol=3)
data
      [,1]      [,2]      [,3]
#[1,] 5.371653 3.490919 3.953603
#[2,] 9.551883 2.681054 9.506765
#[3,] 3.525686 1.027758 8.059011
#[4,] 9.923080 1.337935 1.112361
#[5,] 7.273972 7.627546 1.174340
#[6,] 8.859109 3.778144 9.384526
#[7,] 9.614542 3.866029 7.301729
#[8,] 9.288085 5.804041 8.347907
#[9,] 1.696849 4.650687 7.220209
#[10,] 5.820941 4.799682 5.243663

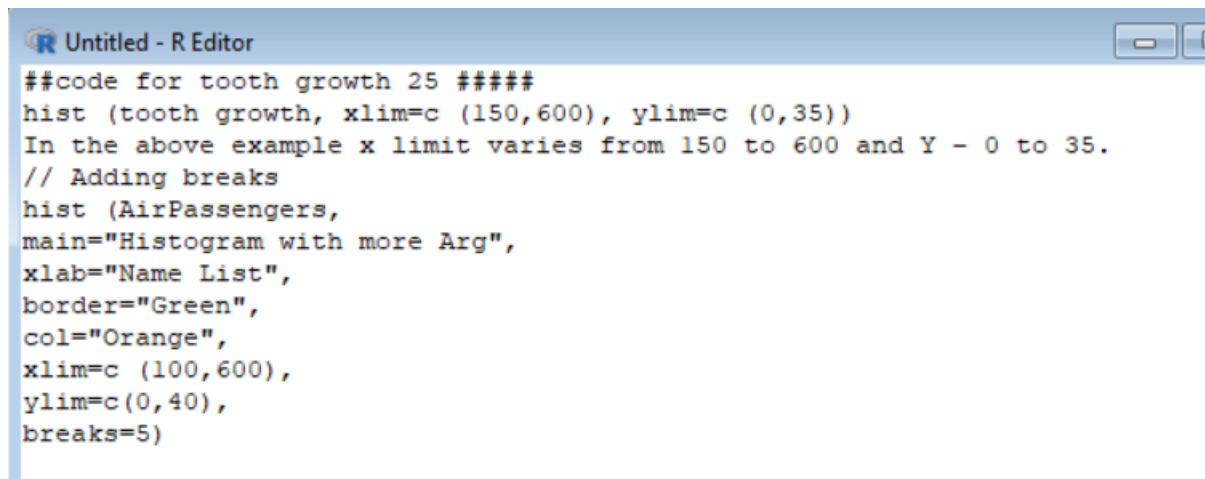
#plot the three columns of the dataset as three lines and add a legend in
#the top right corner of the chart
matplot(data, type = "b",pch=1,col = 1:3)
legend("topright", legend = 1:3, col=1:3, pch=1)
```

code for getting multiple lines in chart.



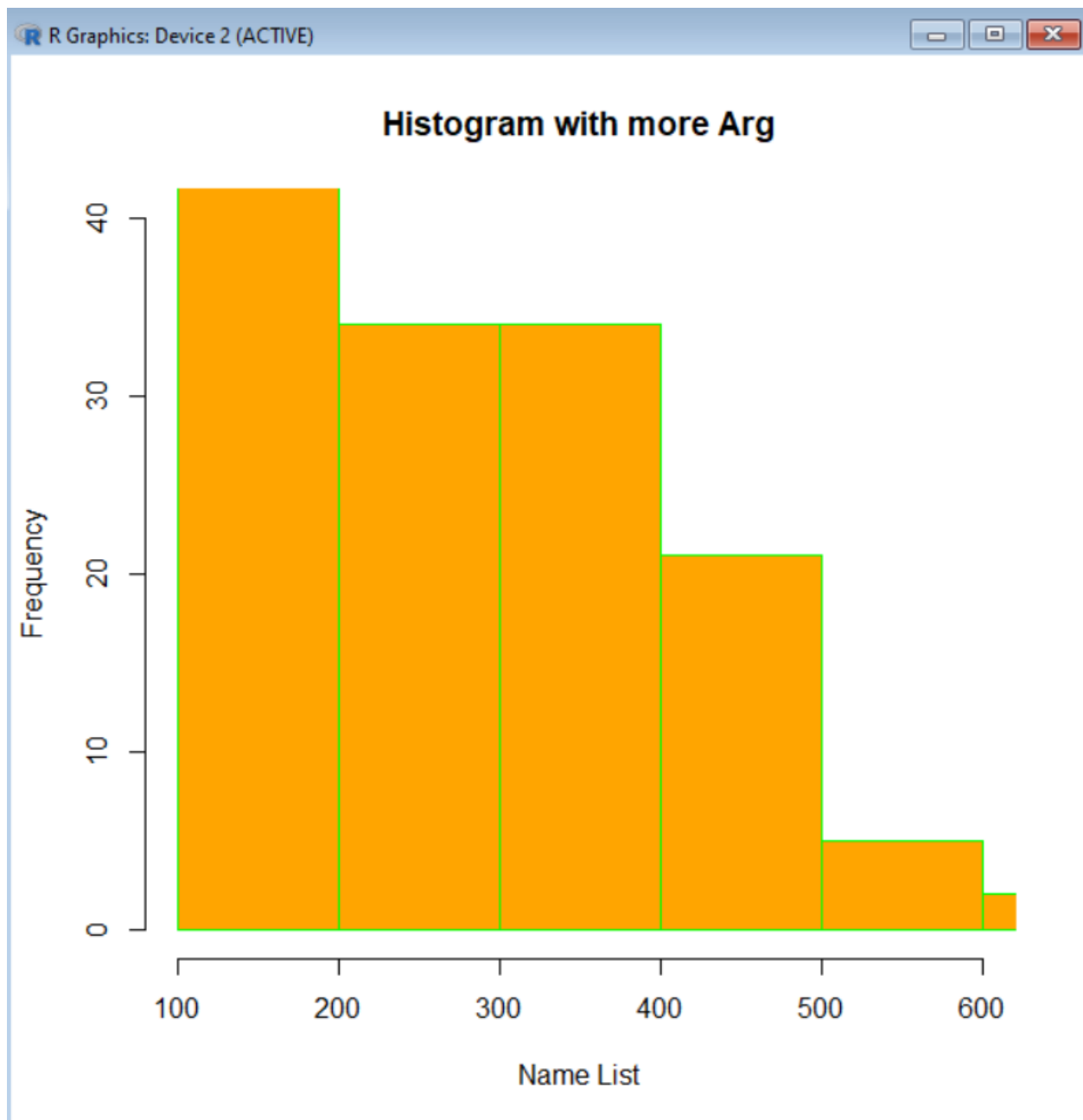
Output of multiple lines in chart.

25. USING R PROGRAM MAKE A HISTOGRAM FOR THE “TOOTHGROWTH” DATASET, START AT 100 ON THE X-AXIS, AND FROM VALUES 200 TO 700, MAKE THE BINS 150 WIDE



```
##code for tooth growth 25 ####
hist (tooth growth, xlim=c (150,600), ylim=c (0,35))
In the above example x limit varies from 150 to 600 and Y - 0 to 35.
// Adding breaks
hist (AirPassengers,
main="Histogram with more Arg",
xlab="Name List",
border="Green",
col="Orange",
xlim=c (100,600),
ylim=c(0,40),
breaks=5)
```

Code for creating a histogram for tooth growth.



Output of histogram, for tooth growth.