Experiment 1

Aim: To get the input from user and perform numerical operations (MAX, MIN, AVG, SUM, SQRT, ROUND) using R.

Code:

```
vec1 <- scan ()
# minimum value present in vector
min (vec1)
# maximum value present in vector
max (vec1)
# average of values present in vector
mean (vec1)
# sum of values present in vector
sum(vec1)
#square root of values in vector
sqr <- sqrt(vec1)
print(sqr)
# round function
round (sqr)
```

Output:

```
> vec1 <- scan ()
1: 1 2 3 4 5 6 7 8 9 10
11:
Read 10 items
> # minimum value present in vector
> min (vec1)
[1] 1
> # maximum value present in vector
> max (vec1)
[1] 10
> # average of values present in vector
> mean (vec1)
[1] 5.5
> # sum of values present in vector
> sum(vec1)
[1] 55
> #square root of values in vector
> sqr <- sqrt(vec1)
> print(sqr)
 [1] 1.0 1.4 1.7 2.0 2.2 2.4 2.6 2.8 3.0 3.2
> # round function
> round (sqr)
 [1] 1 1 2 2 2 2 3 3 3 3
```

Experiment 2

Aim: To perform data import/export (.CSV, .XLS, .TXT) operations using data frames in R.

Code:

```
#importing the csv file
data<- read.csv("C:\\Users\\DELL\\Downloads\\samplecsv.csv")
View(data)

#importing the xls files
library(readxl)
xlsfile <-
read_excel("C:\\Users\\DELL\\Downloads\\samplexls.xls")
View(xlsfile)

#importing the text file
library(readr)
textfile <-
read_csv("C:\\Users\\DELL\\Downloads\\sampletxt.txt")
View(textfile)


#exporting data frames to csv in r
Rollno <- c("1","2","3","4","5")
Name <- c("Rohit", "Rayirth", "Riya", "Himanshu", "Nikhil")
Marks <- c("82","88","90","86","78")
Age <- c("21", "20", "20", "19","22")
data <- data.frame (Rollno, Name, Marks, Age)

#exporting the data frame to csv in r
write.csv(data,
file="C:\\Users\\DELL\\Documents\\rexports\\sampleexcsv.csv",
row.names = FALSE)

#exporting the data frame to xls in r
library('writexl')
write_xlsx(data,
"C:\\Users\\DELL\\Documents\\rexports\\sampleexxls.xlsx")

#exporting the data frame to text file in r
write.table(data,
file="C:\\Users\\DELL\\Documents\\rexports\\sampleextxt.txt",
row.names = FALSE)
```

Output:

```
> #importing the csv file
> data<- read.csv("C:\\Users\\DELL\\Downloads\\samplecsv.csv")
> View(data)
> #importing the xls files
> library(readxl)
> xlsfile <- read_excel("C:\\Users\\DELL\\Downloads\\samplexls.xls")

> View(xlsfile)
> #importing the text file
> library(readr)
> textfile <- read_csv("C:\\Users\\DELL\\Downloads\\sampletxt.txt")
Rows: 4 Columns: 1
— Column specification ——————————————————————————————————————
Delimiter: ","
chr (1): Utilitatis causa amicitia est quaesita.

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
Warning message:
One or more parsing issues, call `problems()` on your data frame for details, e.g.:
  dat <- vroom(...)
  problems(dat)
> View(textfile)
> #exporting data frames to csv in r
> Rollno <- c("1","2","3","4","5")
> Name <- c("Rohit", "Rayirth", "Riya", "Himanshu", "Nikhil")
> Marks <- c("82","88","90","86","78")
> Age <- c("21", "20", "20", "19","22")
> data <- data.frame (Rollno, Name, Marks, Age)
> #exporting the data frame to csv in r
> write.csv(data, file="C:\\Users\\DELL\\Documents\\rexports\\sampleexcsv.csv", row.names = FALSE)
> #exporting the data frame to xls in r
> library('writexl')
> write_xlsx(data, "C:\\Users\\DELL\\Documents\\rexports\\sampleexxls.xlsx")
> #exporting the data frame to text file in r
> write.table(data, file="C:\\Users\\DELL\\Documents\\rexports\\sampleextxt.txt", row.names = FALSE)
```

Import .csv

Filter

	Rollno	Name	Marks	Age
1	1	Rohit	82	21
2	2	Rayirth	88	20
3	3	Riya	90	20
4	4	Himanshu	86	19
5	5	Nikhil	78	22

Import .xls

Filter

	0	First Name	Last Name	Gender	Country	Age	Date	Id
1	1	Dulce	Abril	Female	United States	32	15/10/2017	1562
2	2	Mara	Hashimoto	Female	Great Britain	25	16/08/2016	1582
3	3	Philip	Gent	Male	France	36	21/05/2015	2587
4	4	Kathleen	Hanner	Female	United States	25	15/10/2017	3549
5	5	Nereida	Magwood	Female	United States	58	16/08/2016	2468
6	6	Gaston	Brumm	Male	United States	24	21/05/2015	2554
7	7	Etta	Hurn	Female	Great Britain	56	15/10/2017	3598
8	8	Earlean	Melgar	Female	United States	27	16/08/2016	2456
9	9	Vincenza	Weiland	Female	United States	40	21/05/2015	6548

Import .txt

	Utilitatis causa amicitia est quaesita.
1	Lorem ipsum dolor sit amet, consectetur adipiscing elit. Coll...
2	Quamquam id quidem licebit iis existimare
3	, quo modo. Quid sequatur
4	, vident. Iam id ipsum absurdum

Export .csv

	A	B	C	D
1	Rollno	Name	Marks	Age
2	1	Rohit	82	21
3	2	Rayirth	88	20
4	3	Riya	90	20
5	4	Himanshu	86	19
6	5	Nikhil	78	22
7				

Export.xls

	A	B	C	D
1	**Rollno**	**Name**	**Marks**	**Age**
2	1	Rohit	82	21
3	2	Rayirth	88	20
4	3	Riya	90	20
5	4	Himanshu	86	19
6	5	Nikhil	78	22
7				

Export.txt

sampleextxt

File Edit View

```
"Rollno" "Name" "Marks" "Age"
"1" "Rohit" "82" "21"
"2" "Rayirth" "88" "20"
"3" "Riya" "90" "20"
"4" "Himanshu" "86" "19"
"5" "Nikhil" "78" "22"
```

Experiment 3

Aim: To get the input matrix from user and perform Matrix addition, subtraction, multiplication, inverse transpose and division operations using vector concept in R.

Code:

```
x <- scan()
y <- scan ()

A <- matrix(c(x), nrow=3, ncol=3)
B <- matrix(c(y), nrow=3, ncol=3)

print (A)
print(B)

# Matrix Addition
print (A + B)

# Matrix Subtraction
print(A - B)

# Matrix Multiplication
print (A %% B)

# Matrix Division
print(A/B)

#Original Matrix
print(A)

#transpose of matrix
print(t(A))

# inverse of matrix
library('matlib')
print(inv(A))
```

Output:

```
> x <- scan()
1: 5 7 9
4: 4 3 8
7: 7 5 6
10:
Read 9 items
> y <- scan ()
1: 6 3 2
4: 8 7 4
7: 2 5 8
10:
Read 9 items
> A <- matrix(c(x), nrow=3, ncol=3)
> B <- matrix(c(y), nrow=3, ncol=3)
> print (A)
     [,1] [,2] [,3]
[1,]    5    4    7
[2,]    7    3    5
[3,]    9    8    6
> print(B)
     [,1] [,2] [,3]
[1,]    6    8    2
[2,]    3    7    5
[3,]    2    4    8
> # Matrix Addition
> print (A + B)
     [,1] [,2] [,3]
[1,]   11   12    9
[2,]   10   10   10
[3,]   11   12   14
> # Matrix Subtraction
> print(A - B)
     [,1] [,2] [,3]
[1,]   -1   -4    5
[2,]    4   -4    0
[3,]    7    4   -2
```

```
> # Matrix Multiplication
> print (A %% B)
     [,1] [,2] [,3]
[1,]    5    4    1
[2,]    1    3    0
[3,]    1    0    6
> # Matrix Division
> print(A/B)
     [,1] [,2] [,3]
[1,] 0.83 0.50 3.50
[2,] 2.33 0.43 1.00
[3,] 4.50 2.00 0.75
> #Original Matrix
> print(A)
     [,1] [,2] [,3]
[1,]    5    4    7
[2,]    7    3    5
[3,]    9    8    6
> #transpose of matrix
> print(t(A))
     [,1] [,2] [,3]
[1,]    5    7    9
[2,]    4    3    8
[3,]    7    5    6
> # inverse of matrix
> library('matlib')
> print(inv(A))
       [,1]   [,2]    [,3]
[1,] -0.210  0.305 -0.0095
[2,]  0.029 -0.314  0.2286
[3,]  0.276 -0.038 -0.1238
```

Experiment 4

Aim: To perform statistical operations (Mean, Median, Mode and Standard deviation) using R.

Code:

```
x <- c(42, 32, 35, 38, 38, 40, 36, 40, 20, 36, 33, 36, 45, 46)

#Mean
print(mean(x))

# Median
print(median (x))

# Mode
getmode <- function(x) {
  uniqx <- unique (x)
  uniqx [which.max(tabulate(match(x, uniqx)))]
}
print(getmode(x))

# Variance
print(var(x))

# Standard Deviation
print(sd(x))
```

Output:

```
> x <- c(42, 32, 35, 38, 38, 40, 36, 40, 20, 36, 33, 36, 45, 46)
> #Mean
> print(mean(x))
[1] 37
> # Median
> print(median (x))
[1] 37
> # Mode
> getmode <- function(x) {
+   uniqx <- unique (x)
+   uniqx [which.max(tabulate(match(x, uniqx)))]
+ }
> print(getmode(x))
[1] 36
> # Variance
> print(var(x))
[1] 41
> # Standard Deviation
> print(sd(x))
[1] 6.4
```

Experiment 5

Aim: To perform data pre-processing operations i) Handling Missing data ii) Min-Max normalization.

(i)Handling Missing Data

Code:

```
x <- c(NA, 1, 9, 7, NA, 0, 1, 10, 0, NA, NA)
is.nan(x) # checking whether some missing values are there in
vector or not

# method 1 to deal with missing values
# extract values except for NA values from the vector
d <- is.na(x)
x[!d]

# method 2 (Imputation)
library('mice')
input_data = nhanes
# handling hyp columns separately as it can contain only 2
values (1 or 2)
input_data$hyp = as.factor (input_data$hyp)
summary (input_data)

# mean substitution
input_data$bmi [which (is.na (input_data$bmi))] = mean
(input_data$bmi, na.rm=TRUE)
input_data$chl [which (is.na (input_data$chl))] = mean
(input_data$chl, na.rm=TRUE)

# for hyp column we cannot replace NA value with mean of that
column
# so just marking NA values with 1 in hyp column
input_data$hyp [which (is.na (input_data$hyp))] = 1

# median substitution
input_data$bmi [which (is.na(input_data$bmi))] = median
(input_data$bmi, na.rm=TRUE)
input_data$chl [which (is.na(input_data$chl))] = median
(input_data$chl, na.rm=TRUE)
```

Output:

```
> x <- c(NA, 1, 9, 7, NA, 0, 1, 10, 0, NA, NA)
> is.nan(x) # checking whether some missing values are there in vector or not
 [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
> # method 1 to deal with missing values
> # extract values except for NA values from the vector
> d <- is.na(x)
> x[!d]
[1]  1  9  7  0  1 10  0
> # method 2 (Imputation)
> library('mice')
> input_data = nhanes
> View(input_data)
> # handling hyp columns separately as it can contain only 2 values (1 or 2)
> input_data$hyp = as.factor (input_data$hyp)
> summary (input_data)
      age             bmi         hyp          chl
 Min.   :1.00   Min.   :20   1   :13   Min.   :113
 1st Qu.:1.00   1st Qu.:23   2   : 4   1st Qu.:185
 Median :2.00   Median :27   NA's: 8   Median :187
 Mean   :1.76   Mean   :27             Mean   :191
 3rd Qu.:2.00   3rd Qu.:29             3rd Qu.:212
 Max.   :3.00   Max.   :35             Max.   :284
                NA's   :9              NA's   :10
> # mean substitution
> input_data$bmi [which (is.na (input_data$bmi))] = mean (input_data$bmi, na.rm=TRUE)
> input_data$chl [which (is.na (input_data$chl))] = mean (input_data$chl, na.rm=TRUE)
> # for hyp column we cannot replace NA value with mean of that column
> # so just marking NA values with 1 in hyp column
> input_data$hyp [which (is.na (input_data$hyp))] = 1
> # median substitution
> input_data$bmi [which (is.na(input_data$bmi))] = median (input_data$bmi, na.rm=TRUE)
> input_data$chl [which (is.na(input_data$chl))] = median (input_data$chl, na.rm=TRUE)
```

NHANES Dataset

	age	bmi	hyp	chl
1	1	NA	NA	NA
2	2	23	1	187
3	1	NA	1	187
4	3	NA	NA	NA
5	1	20	1	113
6	3	NA	NA	184
7	1	22	1	118
8	1	30	1	187
9	2	22	1	238
10	2	NA	NA	NA
11	1	NA	NA	NA
12	2	NA	NA	NA
13	3	22	1	206
14	2	29	2	204
15	1	30	1	NA
16	1	NA	NA	NA
17	3	27	2	284
18	2	26	2	199
19	1	35	1	218
20	3	26	2	NA
21	1	NA	NA	NA

Mean Substitution

	age	bmi	hyp	chl
1	1	27	NA	191
2	2	23	1	187
3	1	27	1	187
4	3	27	NA	191
5	1	20	1	113
6	3	27	NA	184
7	1	22	1	118
8	1	30	1	187
9	2	22	1	238
10	2	27	NA	191
11	1	27	NA	191
12	2	27	NA	191
13	3	22	1	206
14	2	29	2	204
15	1	30	1	191
16	1	27	NA	191
17	3	27	2	284
18	2	26	2	199
19	1	35	1	218
20	3	26	2	191
21	1	27	NA	191

Median Substitution

	age	bmi	hyp	chl
1	1	27	1	191
2	2	23	1	187
3	1	27	1	187
4	3	27	1	191
5	1	20	1	113
6	3	27	1	184
7	1	22	1	118
8	1	30	1	187
9	2	22	1	238
10	2	27	1	191
11	1	27	1	191
12	2	27	1	191
13	3	22	1	206
14	2	29	2	204
15	1	30	1	191
16	1	27	1	191
17	3	27	2	284
18	2	26	2	199
19	1	35	1	218
20	3	26	2	191
21	1	27	1	191

(ii) Min-Max Normalization

Code:

```
# Min-max normalization

# data set
data = data.frame(
  var1 = c(120, 345, 145, 122, 596, 285, 211),
  var2 = c(10, 15, 45, 22, 53, 28, 12),
  var3 = c(-34, 0.05, 0.15, 0.12, -6, 0.85, 0.11)
)
  # custom function to implement min max scaling
minMax <- function(x) {
  (x - min(x)) / (max(x) - min(x))
}

# normalize data using custom function
normalisedMydata <- as.data.frame (lapply (data, minMax))
head (normalisedMydata)
```

Output:

```
> # data set
> data = data.frame(
+   var1 = c(120, 345, 145, 122, 596, 285, 211),
+   var2 = c(10, 15, 45, 22, 53, 28, 12),
+   var3 = c(-34, 0.05, 0.15, 0.12, -6, 0.85, 0.11)
+ )
>   # custom function to implement min max scaling
> minMax <- function(x) {
+   (x - min(x)) / (max(x) - min(x))
+ }
> # normalize data using custom function
> normalisedMydata <- as.data.frame (lapply (data, minMax))
> head (normalisedMydata)
    var1 var2 var3
1 0.0000 0.00 0.00
2 0.4727 0.12 0.98
3 0.0525 0.81 0.98
4 0.0042 0.28 0.98
5 1.0000 1.00 0.80
6 0.3466 0.42 1.00
```

Experiment 6

Aim: To perform dimensionality reduction operation using PCA for Houses Data Set.

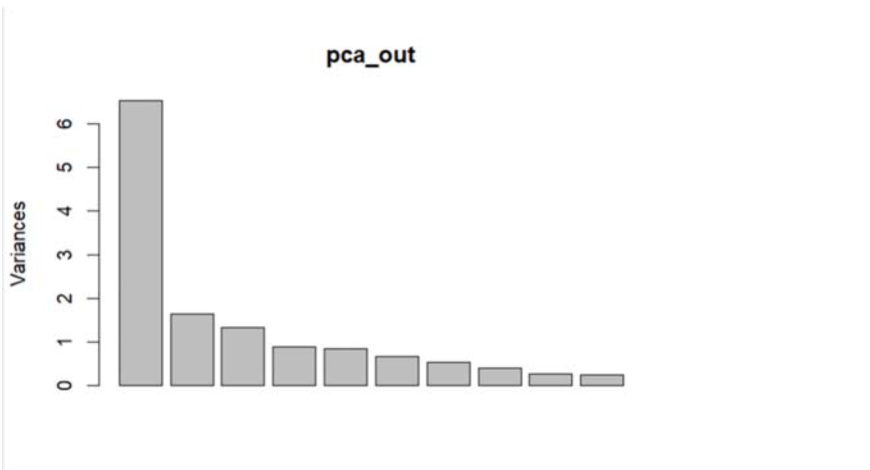
Code:

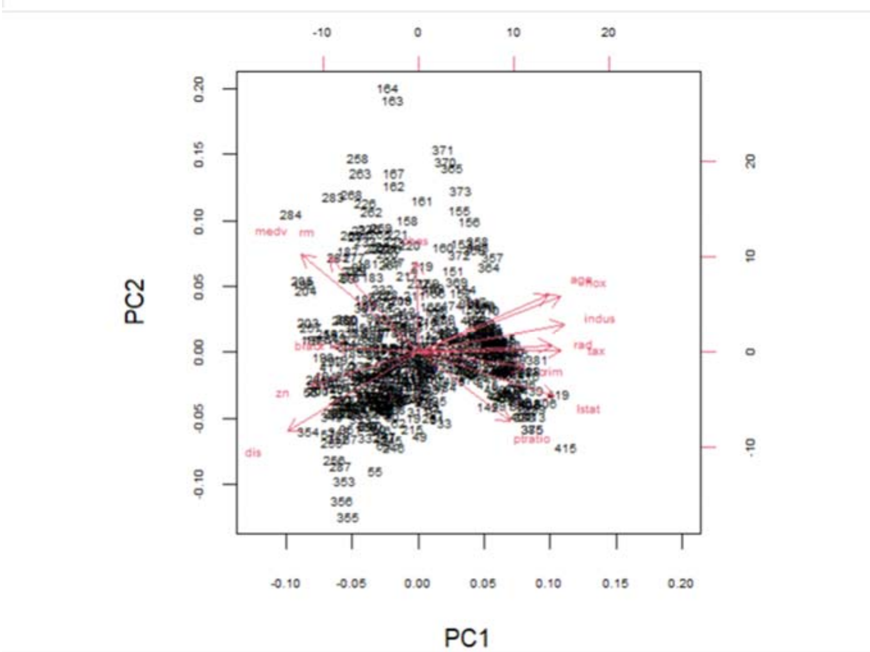
```
data(Boston, package="MASS")

#PCA
pca_out<- prcomp (Boston, scale. =T)
pca_out

boston_pc<- pca_out$x
head(boston_pc)

summary(pca_out)

plot(pca_out)

par(mar=c(4,4,2,2))
biplot(pca_out, cex=0.5, cex.axis=0.5)
```

Output:

```
> data(Boston, package="MASS")
> #PCA
> pca_out<- prcomp (Boston, scale. =T)
> pca_out
Standard deviations (1, .., p=14):
 [1] 2.56 1.28 1.16 0.94 0.92 0.81 0.73 0.63 0.53 0.50 0.46 0.43 0.37 0.25

Rotation (n x k) = (14 x 14):
           PC1     PC2     PC3     PC4     PC5    PC6     PC7    PC8     PC9    PC10   PC11   PC12    PC13
crim     0.242 -0.0659  0.3951  0.1004 -0.0050  0.225 -0.7771  0.157 -0.2542 -0.0714 -0.071 -0.063 -0.0970
zn      -0.245 -0.1480  0.3945  0.3430 -0.1145  0.336  0.2742 -0.380 -0.3829  0.2456 -0.128  0.221  0.1324
indus    0.332  0.1271 -0.0661 -0.0096  0.0226  0.081  0.3403  0.172 -0.6270 -0.2548  0.274 -0.348 -0.0837
chas    -0.005  0.4107 -0.1253  0.7004  0.5352 -0.163 -0.0741 -0.033  0.0186 -0.0417 -0.010  0.019  0.0499
nox      0.325  0.2543 -0.0465  0.0537 -0.1946  0.149  0.1981  0.047  0.0430 -0.2116 -0.437  0.449 -0.5250
rm      -0.203  0.4340  0.3534 -0.2934  0.0083 -0.131 -0.0741 -0.438  0.0037 -0.5261  0.224  0.126  0.0499
age      0.297  0.2603 -0.2008 -0.0784 -0.1498  0.061 -0.1186 -0.588  0.0433  0.2456 -0.330 -0.486  0.0515
dis     -0.298 -0.3591  0.1571  0.1847  0.1062 -0.012  0.1044 -0.128  0.1758 -0.2994 -0.115 -0.494 -0.5523
rad      0.303  0.0311  0.4185 -0.0514  0.2304  0.135  0.1371  0.075  0.4634  0.1158  0.042 -0.019  0.0063
tax      0.324  0.0089  0.3432 -0.0268  0.1634  0.188  0.3140  0.071  0.1794 -0.0084  0.043 -0.170  0.2430
ptratio  0.208 -0.3146  0.0004 -0.3420  0.6157 -0.279 -0.0015 -0.283 -0.2745  0.1605 -0.100  0.232 -0.1883
black   -0.197  0.0265 -0.3614 -0.2017  0.3675  0.786 -0.0748 -0.044  0.0610 -0.1463  0.039  0.042  0.0211
lstat    0.311 -0.2012 -0.1611  0.2426 -0.1784  0.092 -0.0832 -0.357  0.1718  0.0666  0.683  0.182 -0.2495
medv    -0.267  0.4449  0.1632 -0.1803  0.0507  0.054  0.0100  0.152 -0.0708  0.5755  0.242 -0.098 -0.4696
           PC14
crim     0.0591
zn      -0.0963
indus   -0.2355
chas     0.0235
nox      0.0876
rm       0.0072
age     -0.0382
dis      0.0471
rad     -0.6350
tax      0.6988
ptratio  0.0557
black   -0.0162
lstat    0.0831
medv     0.1341
```

```
> boston_pc<- pca_out$x
> head(boston_pc)
   PC1   PC2    PC3    PC4    PC5   PC6   PC7    PC8    PC9     PC10    PC11  PC12  PC13   PC14
1 -2.1  0.49 -0.336 -0.028 -1.012  0.26 -0.33  0.160  0.471 -0.20557 -0.7793  0.11  0.49  0.248
2 -1.4 -0.17 -0.965 -0.432 -0.254 -0.30 -0.56 -0.288  0.196 -0.24600 -0.2773 -0.59  0.11 -0.113
3 -2.4  0.91 -0.090 -1.123  0.033 -0.51 -0.49  0.082 -0.054 -0.19481  0.0289 -0.42 -0.36  0.051
4 -2.8  0.19  0.060 -1.065  0.460 -0.71 -0.62  0.239  0.358 -0.15574 -0.2443 -0.13 -0.58  0.090
5 -2.8  0.43  0.064 -1.129  0.382 -0.66 -0.70 -0.103  0.408 -0.00042  0.0078 -0.22 -0.78  0.148
6 -2.3 -0.33 -0.450 -0.693  0.300 -0.58 -0.65  0.132  0.466  0.11007 -0.4832 -0.35 -0.43  0.024
> summary(pca_out)
Importance of components:
                         PC1   PC2    PC3    PC4    PC5    PC6    PC7    PC8    PC9  PC10   PC11   PC12
Standard deviation     2.559 1.284 1.1614 0.9416 0.9224 0.8124 0.7317 0.6349 0.5266 0.502 0.4613 0.4278
Proportion of Variance 0.468 0.118 0.0964 0.0633 0.0608 0.0471 0.0382 0.0288 0.0198 0.018 0.0152 0.0131
Cumulative Proportion  0.468 0.585 0.6817 0.7451 0.8058 0.8530 0.8912 0.9200 0.9398 0.958 0.9730 0.9861
                          PC13    PC14
Standard deviation     0.36607 0.24561
Proportion of Variance 0.00957 0.00431
Cumulative Proportion  0.99569 1.00000
> plot(pca_out)
> par(mar=c(4,4,2,2))
> biplot(pca_out, cex=0.5, cex.axis=0.5)
```





Experiment 7

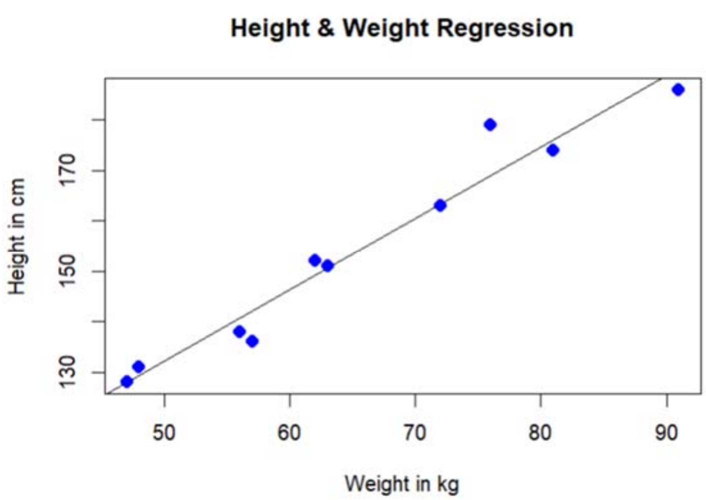
Aim: To perform Simple Linear Regression with R.

Code:

```
x <- c(151, 174, 138, 186, 128, 136, 179, 163, 152, 131)
y <- c(63, 81, 56, 91, 47, 57, 76, 72, 62, 48)

rel<- lm(y~x)
print (summary(rel))

plot(y,x,col = "blue",main = "Height & Weight Regression",
abline (lm(x~y)), cex = 1.3, pch = 16, xlab="Weight in
kg",ylab = "Height in cm")
```

Output:

```
> x <- c(151, 174, 138, 186, 128, 136, 179, 163, 152, 131)
> y <- c(63, 81, 56, 91, 47, 57, 76, 72, 62, 48)
> rel<- lm(y~x)
> print (summary(rel))

Call:
lm(formula = y ~ x)

Residuals:
   Min     1Q Median     3Q    Max 
-6.300 -1.663  0.041  1.894  3.978 

Coefficients:
             Estimate Std. Error t value Pr(>|t|)    
(Intercept) -38.4551     8.0490   -4.78   0.0014 ** 
x             0.6746     0.0519   13.00  1.2e-06 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 3.2 on 8 degrees of freedom
Multiple R-squared:  0.955,	Adjusted R-squared:  0.949 
F-statistic:  169 on 1 and 8 DF,  p-value: 1.16e-06

> plot(y,x,col = "blue",main = "Height & Weight Regression",
+ abline (lm(x~y)), cex = 1.3, pch = 16, xlab="Weight in kg",ylab = "Height in cm")
```



Experiment 8

Aim: To perform K-Means clustering operation and visualize for iris data set.

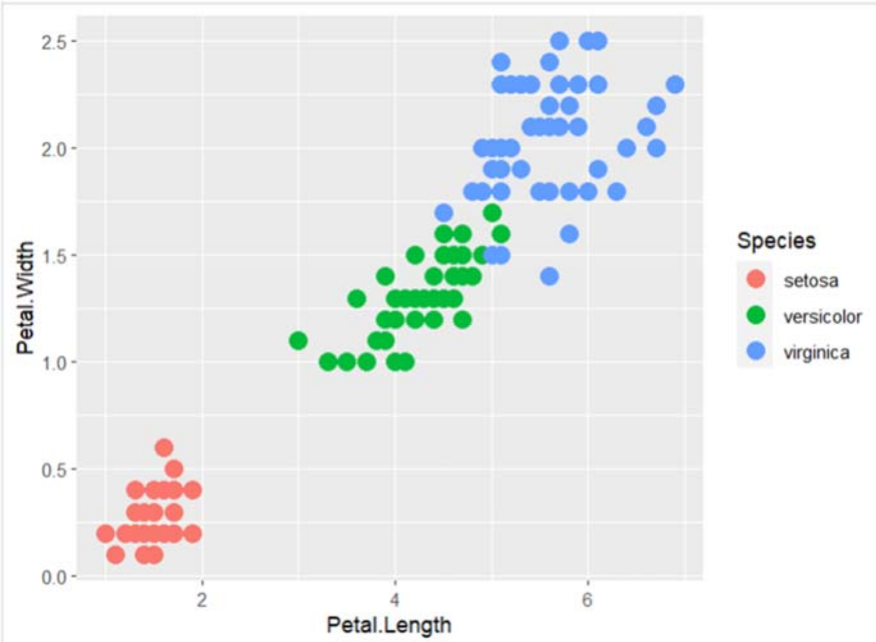
Code:

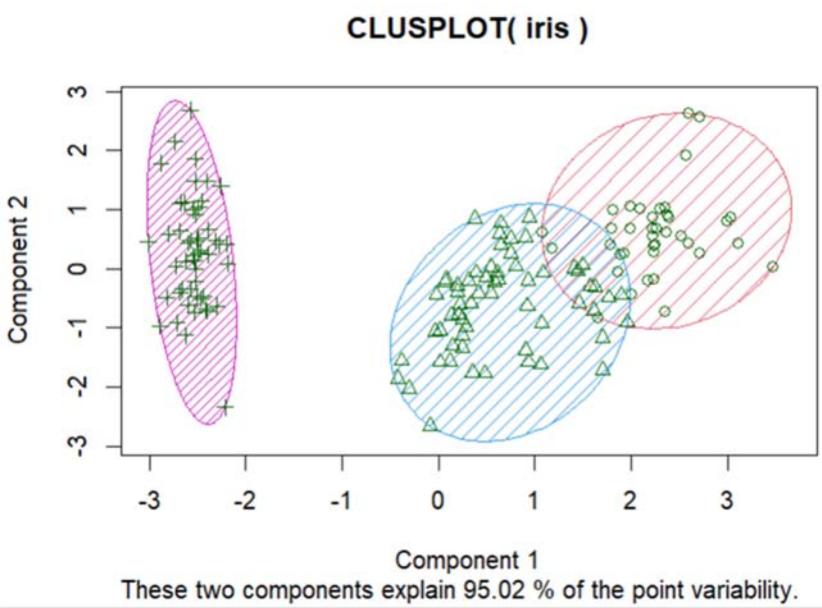
```
library(ggplot2)
library(cluster)

df <-iris
ggplot(df, aes(Petal.Length, Petal.Width)) +
geom_point(aes(col=Species), size=4)

set.seed(101)
irisCluster <- kmeans(df[,1:4], center=3, nstart=20)
irisCluster

table(irisCluster$cluster, df$Species)

clusplot(iris, irisCluster$cluster, color=T, shade=T,
labels=0, lines=0)

# visualize
tot.withinss <- vector(mode="character", length=10)
for (i in 1:10){
  irisCluster <- kmeans(df[,1:4], center=i, nstart=20)
  tot.withinss[i] <- irisCluster$tot.withinss
}
plot(1:10, tot.withinss, type="b", pch=19)
```

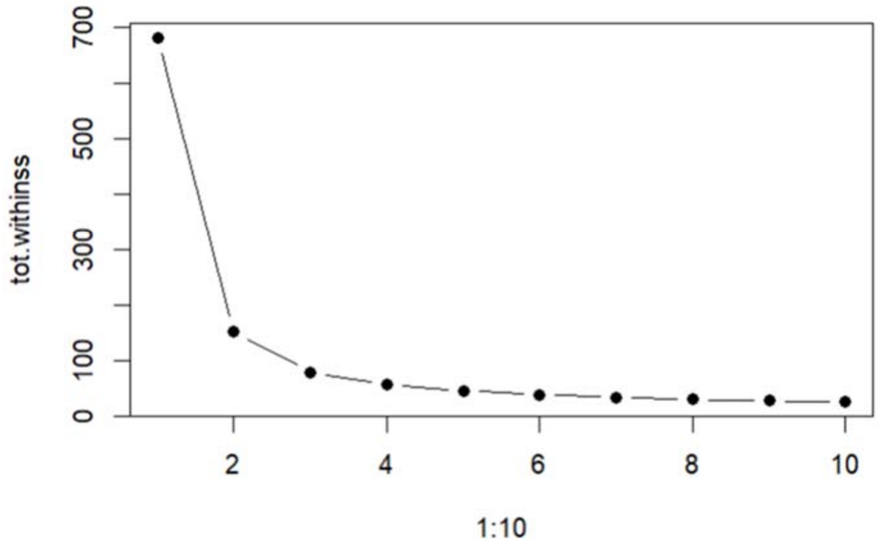
Output:

```
> library(ggplot2)
> library(cluster)
> df <-iris
> ggplot(df, aes(Petal.Length, Petal.Width)) + geom_point(aes(col=Species), size=4)
> set.seed(101)
> irisCluster <- kmeans(df[,1:4], center=3, nstart=20)
> irisCluster
K-means clustering with 3 clusters of sizes 38, 62, 50

Cluster means:
  Sepal.Length Sepal.Width Petal.Length Petal.Width
1          6.8         3.1          5.7        2.07
2          5.9         2.7          4.4        1.43
3          5.0         3.4          1.5        0.25

Clustering vector:
  [1] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 2
 [53] 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 1 1
[105] 1 1 2 1 1 1 1 1 1 2 2 1 1 1 1 2 1 2 1 2 1 1 2 2 1 1 1 1 1 2 1 1 1 1 2 1 1 1 2 1 1 1 2 1 1 2

Within cluster sum of squares by cluster:
[1] 24 40 15
 (between_SS / total_SS =  88.4 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss" "betweenss"    "size"
[8] "iter"         "ifault"
> table(irisCluster$cluster, df$Species)

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

```
> clusplot(iris, irisCluster$cluster, color=T, shade=T, labels=0, lines=0)
> # visualize
> tot.withinss <- vector(mode="character", length=10)
> for (i in 1:10){
+   irisCluster <- kmeans(df[,1:4], center=i, nstart=20)
+   tot.withinss[i] <- irisCluster$tot.withinss
+ }
> plot(1:10, tot.withinss, type="b", pch=19)
```







tot.withinss
0
100
300
500
700
2
4
6
8
10
1:10

Experiment 9

Aim: Write R script to diagnose any disease using KNN classification and plot the results.

Code:

```
# Splitting dataset into train and test set
library(caret)
# for importing knn classifier
library(class)
# for visualisations
library(ggplot2)

data <-
read.csv("C:\\Users\\DELL\\Downloads\\diabetesknn.csv")

head(data)
summary(data)

# checking columns wise nun values
colSums(is.na(data))

correlation_matrix <- cor(data[, -9])

correlation_data <- reshape2::melt(correlation_matrix)

ggplot(data = correlation_data, aes(x = Var1, y = Var2, fill =
value)) +
  geom_tile() +
  scale_fill_gradient2(low = "red", high = "green") +
  labs(title = "Correlation Heatmap", x = "", y = "")+
  theme(axis.text.x = element_text(angle = 45, hjust = 1, size
= 10))

data_scaled <- scale(data[, -9])
head(data_scaled)

set.seed(40)

index <- createDataPartition(data$Outcome, p = 0.85,
list=FALSE)

train_data <- data_scaled[index, ]
test_data <- data_scaled[-index, ]
train_labels <- factor(data$Outcome[index], levels = c(0,1))
test_labels <- factor(data$Outcome[-index], levels = c(0,1))

for (i in 1:10){
  cat("----------------- For k =", i, "-----------------\n")
```

```
  k <- i

  knn_model <- knn(train_data, test_data, train_labels, k=k)

  confusion <- confusionMatrix(knn_model, test_labels)

  confusion_table <- as.table(confusion)
  accuracy <- confusion$overall["Accuracy"]
  cat("Accuracy: ", accuracy, "\n")

  incorrect_classified <- confusion_table[1, 2] +
confusion_table[2, 1]
  total <- sum(confusion_table)
  error <- (incorrect_classified/total)*100

  cat("The error rate is ", error, "%\n")
}

k <- 4

knn_model <- knn(train_data, test_data, train_labels, k=k)

confusion <- confusionMatrix(knn_model, test_labels)

confusion

confusion_table <- as.table(confusion) #getting confusion
table
print("Confusion Table:")
confusion_table # printing confusion table

# getting number of incorrect predictions
incorrect_classified <- confusion_table[1, 2] +
confusion_table[2, 1]

# calculating total number of predictions
total <- sum(confusion_table)

# calculating error rate
error <- (incorrect_classified/total)*100

# printing the error rate
cat("The error rate is ", error, "%")
```

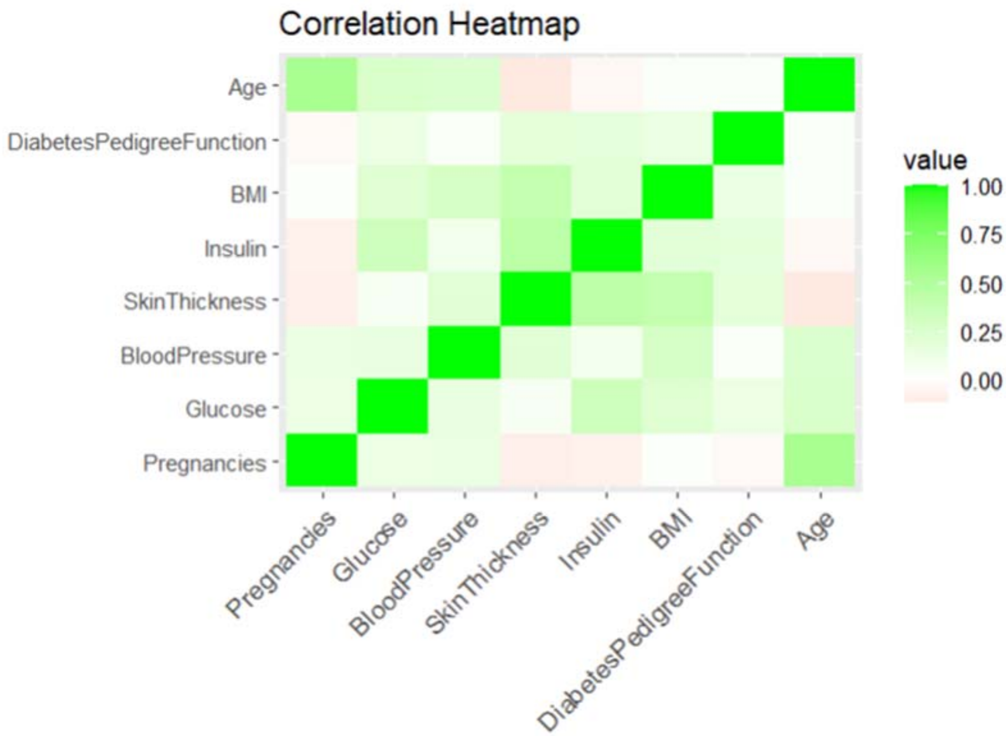
Output:

```
> # Splitting dataset into train and test set
> library(caret)
Loading required package: lattice
> # for importing knn classifier
> library(class)
> # for visualisations
> library(ggplot2)
> data <- read.csv("C:\\Users\\DELL\\Downloads\\diabetesknn.csv")
> head(data)
  Pregnancies Glucose BloodPressure SkinThickness Insulin BMI DiabetesPedigreeFunction Age Outcome
1           6     148            72            35       0  34                     0.63  50       1
2           1      85            66            29       0  27                     0.35  31       0
3           8     183            64             0       0  23                     0.67  32       1
4           1      89            66            23      94  28                     0.17  21       0
5           0     137            40            35     168  43                     2.29  33       1
6           5     116            74             0       0  26                     0.20  30       0
> summary(data)
  Pregnancies      Glucose     BloodPressure SkinThickness     Insulin         BMI       DiabetesPedigreeFunction
 Min.   : 0.0   Min.   :  0   Min.   :  0   Min.   : 0    Min.   :  0   Min.   : 0   Min.   :0.08
 1st Qu.: 1.0   1st Qu.: 99   1st Qu.: 62   1st Qu.: 0    1st Qu.:  0   1st Qu.:27   1st Qu.:0.24
 Median : 3.0   Median :117   Median : 72   Median :23    Median : 30   Median :32   Median :0.37
 Mean   : 3.8   Mean   :121   Mean   : 69   Mean   :21    Mean   : 80   Mean   :32   Mean   :0.47
 3rd Qu.: 6.0   3rd Qu.:140   3rd Qu.: 80   3rd Qu.:32    3rd Qu.:127   3rd Qu.:37   3rd Qu.:0.63
 Max.   :17.0   Max.   :199   Max.   :122   Max.   :99    Max.   :846   Max.   :67   Max.   :2.42
      Age        Outcome
 Min.   :21   Min.   :0.00
 1st Qu.:24   1st Qu.:0.00
 Median :29   Median :0.00
 Mean   :33   Mean   :0.35
 3rd Qu.:41   3rd Qu.:1.00
 Max.   :81   Max.   :1.00
> # checking columns wise nun values
> colSums(is.na(data))
             Pregnancies                  Glucose            BloodPressure            SkinThickness
                       0                        0                        0                        0
                 Insulin                      BMI DiabetesPedigreeFunction                      Age
                       0                        0                        0                        0
                 Outcome
                       0
```

```
> correlation_matrix <- cor(data[, -9])
> correlation_data <- reshape2::melt(correlation_matrix)
> ggplot(data = correlation_data, aes(x = Var1, y = Var2, fill = value)) +
+   geom_tile() +
+   scale_fill_gradient2(low = "red", high = "green") +
+   labs(title = "Correlation Heatmap", x = "", y = "")+
+   theme(axis.text.x = element_text(angle = 45, hjust = 1, size = 10))
> data_scaled <- scale(data[, -9])
> head(data_scaled)
     Pregnancies Glucose BloodPressure SkinThickness Insulin   BMI DiabetesPedigreeFunction   Age
[1,]        0.64    0.85          0.15          0.91   -0.69  0.20                     0.47  1.43
[2,]       -0.84   -1.12         -0.16          0.53   -0.69 -0.68                    -0.36 -0.19
[3,]        1.23    1.94         -0.26         -1.29   -0.69 -1.10                     0.60 -0.11
[4,]       -0.84   -1.00         -0.16          0.15    0.12 -0.49                    -0.92 -1.04
[5,]       -1.14    0.50         -1.50          0.91    0.77  1.41                     5.48 -0.02
[6,]        0.34   -0.15          0.25         -1.29   -0.69 -0.81                    -0.82 -0.28
> set.seed(40)
> index <- createDataPartition(data$Outcome, p = 0.85, list=FALSE)
> train_data <- data_scaled[index, ]
> test_data <- data_scaled[-index, ]
> train_labels <- factor(data$Outcome[index], levels = c(0,1))
> test_labels <- factor(data$Outcome[-index], levels = c(0,1))
> for (i in 1:10){
+   cat("---------------- For k =", i, "----------------\n")
+   k <- i
+
+   knn_model <- knn(train_data, test_data, train_labels, k=k)
+
+   confusion <- confusionMatrix(knn_model, test_labels)
+
+   confusion_table <- as.table(confusion)
+   accuracy <- confusion$overall["Accuracy"]
+   cat("Accuracy: ", accuracy, "\n")
+
+   incorrect_classified <- confusion_table[1, 2] + confusion_table[2, 1]
+   total <- sum(confusion_table)
+   error <- (incorrect_classified/total)*100
+
+   cat("The error rate is ", error, "%\n")
+ }
```

```
----------------- For k = 1 -----------------
Accuracy:  0.71
The error rate is  29 %
----------------- For k = 2 -----------------
Accuracy:  0.74
The error rate is  26 %
----------------- For k = 3 -----------------
Accuracy:  0.75
The error rate is  25 %
----------------- For k = 4 -----------------
Accuracy:  0.77
The error rate is  23 %
----------------- For k = 5 -----------------
Accuracy:  0.76
The error rate is  24 %
----------------- For k = 6 -----------------
Accuracy:  0.77
The error rate is  23 %
----------------- For k = 7 -----------------
Accuracy:  0.77
The error rate is  23 %
----------------- For k = 8 -----------------
Accuracy:  0.76
The error rate is  24 %
----------------- For k = 9 -----------------
Accuracy:  0.74
The error rate is  26 %
----------------- For k = 10 -----------------
Accuracy:  0.74
The error rate is  26 %
> k <- 4
> knn_model <- knn(train_data, test_data, train_labels, k=k)
> confusion <- confusionMatrix(knn_model, test_labels)
> confusion
```

```
Confusion Matrix and Statistics

          Reference
Prediction  0  1
         0 62 22
         1  6 25

               Accuracy : 0.757
                 95% CI : (0.668, 0.832)
    No Information Rate : 0.591
    P-Value [Acc > NIR] : 0.000151

                  Kappa : 0.468

 Mcnemar's Test P-Value : 0.004586

            Sensitivity : 0.912
            Specificity : 0.532
         Pos Pred Value : 0.738
         Neg Pred Value : 0.806
             Prevalence : 0.591
         Detection Rate : 0.539
   Detection Prevalence : 0.730
      Balanced Accuracy : 0.722

       'Positive' Class : 0

> confusion_table <- as.table(confusion) #getting confusion table
> print("Confusion Table:")
[1] "Confusion Table:"
> confusion_table # printing confusion table
          Reference
Prediction  0  1
         0 62 22
         1  6 25

> # getting number of incorrect predictions
> incorrect_classified <- confusion_table[1, 2] + confusion_table[2, 1]
> # calculating total number of predictions
> total <- sum(confusion_table)
> # calculating error rate
> error <- (incorrect_classified/total)*100
> # printing the error rate
> cat("The error rate is ", error, "%")
The error rate is  24 %
```


Correlation Heatmap
Age
DiabetesPedigreeFunction
BMI
Insulin
SkinThickness
BloodPressure
Glucose
Pregnancies
value
1.00
0.75
0.50
0.25
0.00
Pregnancies
Glucose
BloodPressure
SkinThickness
Insulin
BMI
DiabetesPedigreeFunction
Age

Experiment 10

Aim: To perform market basket analysis using Association Rules (Apriori).

Code:

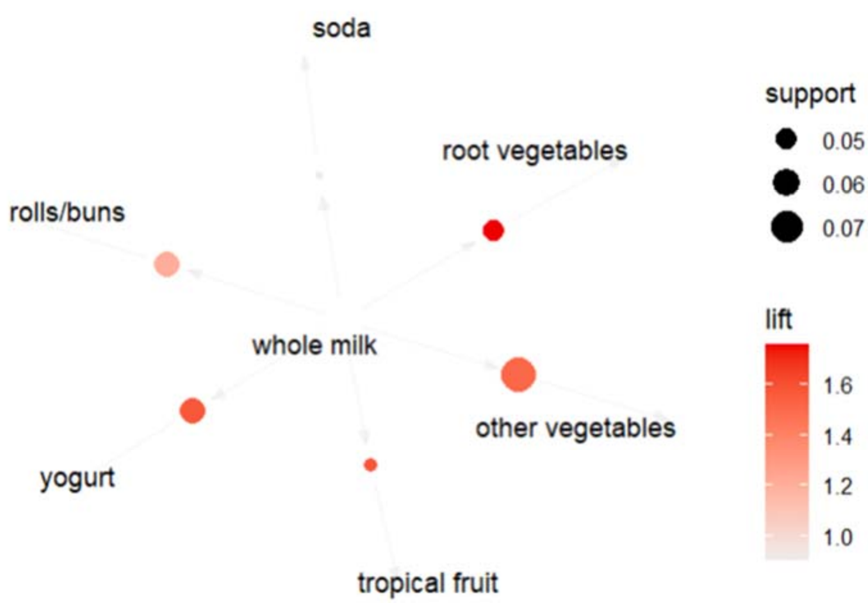
```
library(arules)
library(arulesViz)
library(dataset)

data(Groceries)
rules<- apriori(Groceries, parameter = list (supp=0.001,
conf=0.8))
options (digits=2)
inspect (rules [1:5])

rules<-sort(rules, by="confidence", decreasing=TRUE)
rules
rules <- apriori(Groceries, parameter = list(supp=0.001,
conf=0.8,maxlen=3))

subset.matrix <- is.subset(rules, rules)
subset.matrix[lower.tri(subset.matrix, diag=T)] <- NA
redundant <- colSums (subset.matrix, na.rm=T) >= 1
rules.pruned <- rules [!redundant]
rules<-rules.pruned

rules<-apriori(data=Groceries, parameter=list (supp=0.001,
conf=0.08),
  appearance=list (default="lhs", rhs="whole milk"),
  control=list (verbose=F))
rules<-sort(rules, decreasing=TRUE, by="confidence")
inspect (rules[1:5])

rules<-apriori (data=Groceries, parameter=list(supp=0.001,
conf = 0.15, minlen=2),
  appearance=list (default="rhs", lhs="whole milk"),
  control=list (verbose=F))
rules<-sort(rules, decreasing=TRUE, by="confidence")
inspect (rules [1:5])

plot (rules, method="graph")
```

Output:

```
> library(arules)
> library(arulesViz)
> library(dataset)
> data(Groceries)
> rules<- apriori(Groceries, parameter = list (supp=0.001, conf=0.8))
Apriori

Parameter specification:
 confidence minval smax arem  aval originalSupport maxtime support minlen maxlen target  ext
        0.8    0.1    1 none FALSE            TRUE       5   0.001      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: 9

set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[169 item(s), 9835 transaction(s)] done [0.01s].
sorting and recoding items ... [157 item(s)] done [0.00s].
creating transaction tree ... done [0.00s].
checking subsets of size 1 2 3 4 5 6 done [0.02s].
writing ... [410 rule(s)] done [0.00s].
creating S4 object  ... done [0.00s].
> options (digits=2)
> inspect (rules [1:5])
    lhs                        rhs             support confidence coverage lift count
[1] {liquor, red/blush wine} => {bottled beer} 0.0019  0.90       0.0021   11.2 19
[2] {curd, cereals}          => {whole milk}   0.0010  0.91       0.0011    3.6 10
[3] {yogurt, cereals}        => {whole milk}   0.0017  0.81       0.0021    3.2 17
[4] {butter, jam}            => {whole milk}   0.0010  0.83       0.0012    3.3 10
[5] {soups, bottled beer}    => {whole milk}   0.0011  0.92       0.0012    3.6 11
> rules<-sort(rules, by="confidence", decreasing=TRUE)
> rules
set of 410 rules
```

```
> rules <- apriori(Groceries, parameter = list(supp=0.001, conf=0.8,maxlen=3))
Apriori

Parameter specification:
 confidence minval smax arem  aval originalSupport maxtime support minlen maxlen target  ext
        0.8    0.1    1 none FALSE            TRUE       5   0.001      1      3  rules TRUE

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

Absolute minimum support count: 9

set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[169 item(s), 9835 transaction(s)] done [0.01s].
sorting and recoding items ... [157 item(s)] done [0.00s].
creating transaction tree ... done [0.00s].
checking subsets of size 1 2 3 done [0.01s].
writing ... [29 rule(s)] done [0.00s].
creating S4 object  ... done [0.00s].
Warning message:
In apriori(Groceries, parameter = list(supp = 0.001, conf = 0.8,  :
  Mining stopped (maxlen reached). Only patterns up to a length of 3 returned!
> subset.matrix <- is.subset(rules, rules)
> subset.matrix[lower.tri(subset.matrix, diag=T)] <- NA
Warning message:
In `[<-`(`*tmp*`, as.vector(i), value = NA) :
  x[.] <- val: x is "ngTMatrix", val not in {TRUE, FALSE} is coerced; NA |--> TRUE.
> redundant <- colSums (subset.matrix, na.rm=T) >= 1
> rules.pruned <- rules [!redundant]
> rules<-rules.pruned
> rules<-apriori(data=Groceries, parameter=list (supp=0.001, conf=0.08),
+   appearance=list (default="lhs", rhs="whole milk"),
+   control=list (verbose=F))
> rules<-sort(rules, decreasing=TRUE, by="confidence")
> inspect (rules[1:5])
```

```
    lhs                                                  rhs           support confidence coverage lift count
[1] {rice, sugar}                                     => {whole milk} 0.0012  1          0.0012   3.9  12
[2] {canned fish, hygiene articles}                   => {whole milk} 0.0011  1          0.0011   3.9  11
[3] {root vegetables, butter, rice}                   => {whole milk} 0.0010  1          0.0010   3.9  10
[4] {root vegetables, whipped/sour cream, flour}      => {whole milk} 0.0017  1          0.0017   3.9  17
[5] {butter, soft cheese, domestic eggs}              => {whole milk} 0.0010  1          0.0010   3.9  10
> rules<-apriori (data=Groceries, parameter=list(supp=0.001, conf = 0.15, minlen=2),
+   appearance=list (default="rhs", lhs="whole milk"),
+   control=list (verbose=F))
> rules<-sort(rules, decreasing=TRUE, by="confidence")
> inspect (rules [1:5])
    lhs             rhs                support confidence coverage lift count
[1] {whole milk} => {other vegetables} 0.075   0.29       0.26     1.5  736
[2] {whole milk} => {rolls/buns}       0.057   0.22       0.26     1.2  557
[3] {whole milk} => {yogurt}           0.056   0.22       0.26     1.6  551
[4] {whole milk} => {root vegetables}  0.049   0.19       0.26     1.8  481
[5] {whole milk} => {tropical fruit}   0.042   0.17       0.26     1.6  416
> plot (rules, method="graph")
```

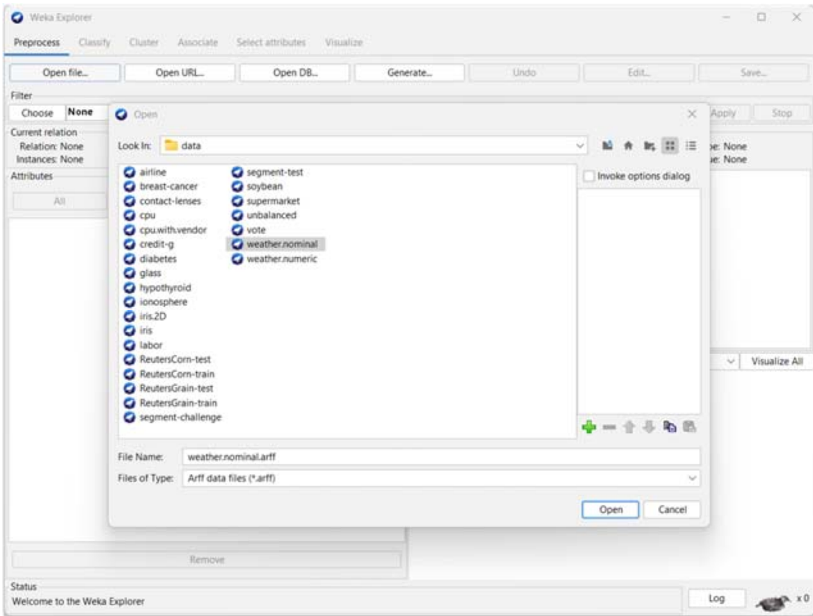


Experiment 11

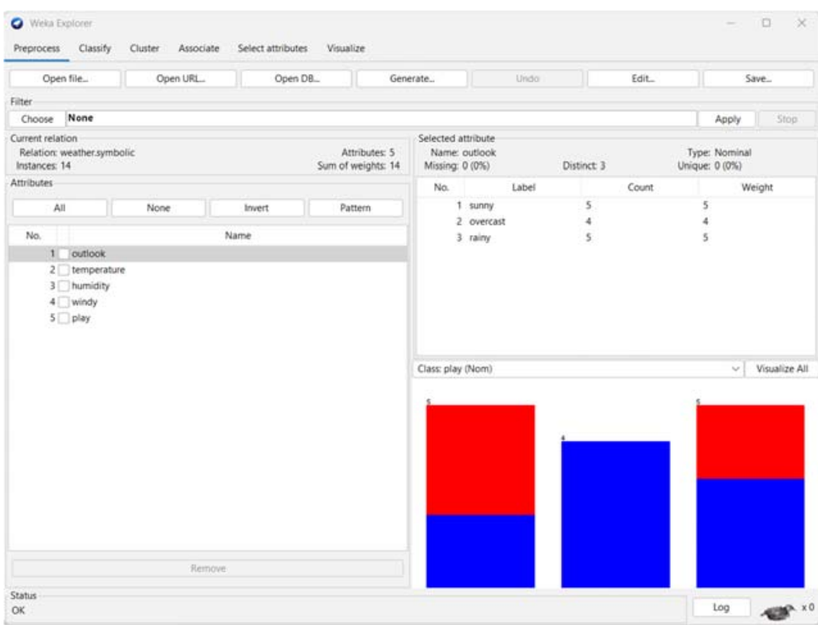
Aim: Implement data pre-processing in Weka.

Steps:

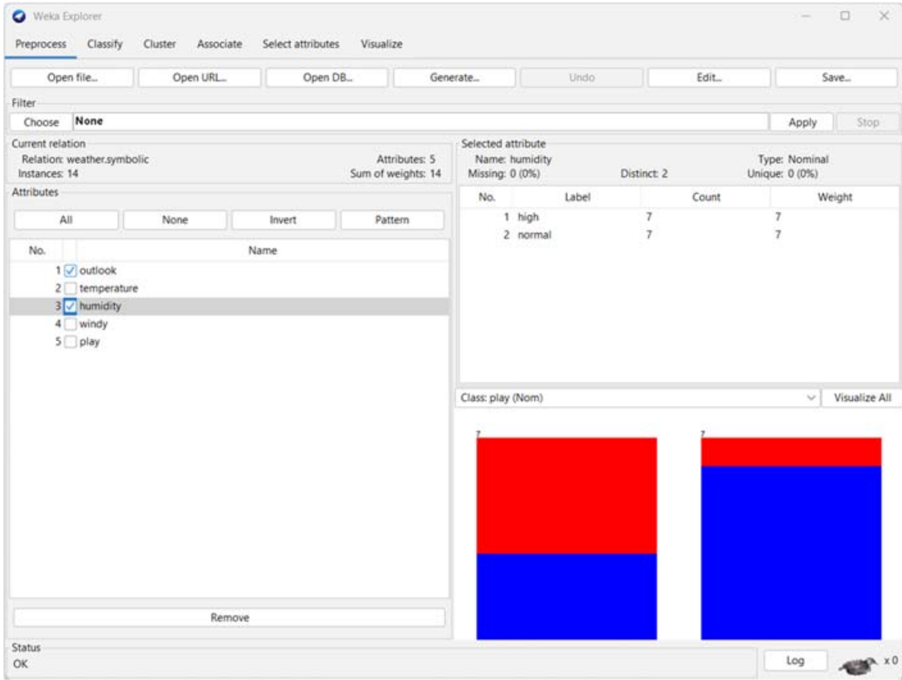
- Using the **Open file ...** option under the **Preprocess** tag select the **weather-nominal.arff** file.

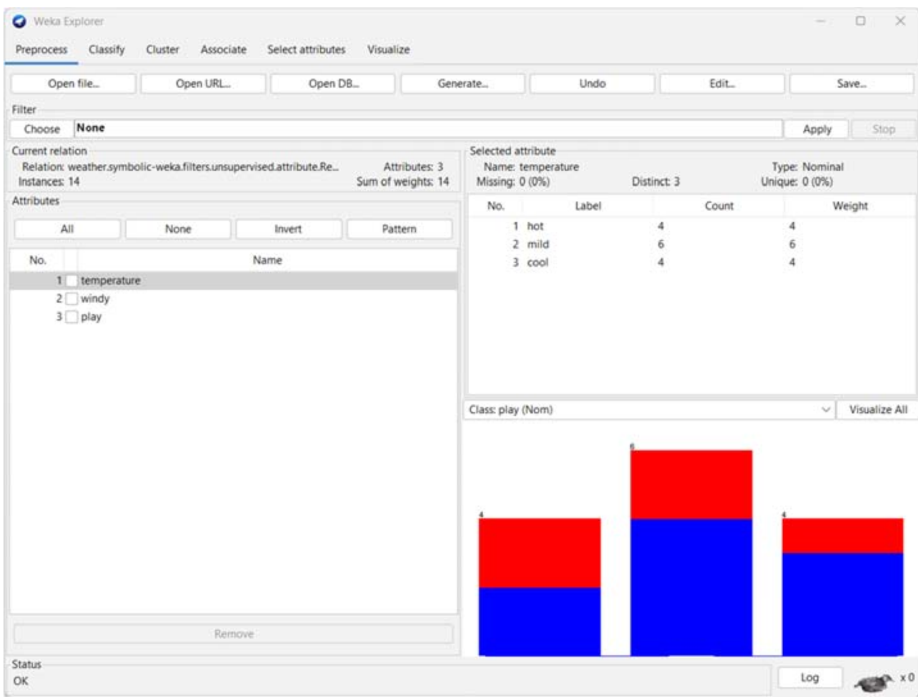


- When you open the file, your screen looks like as shown here:

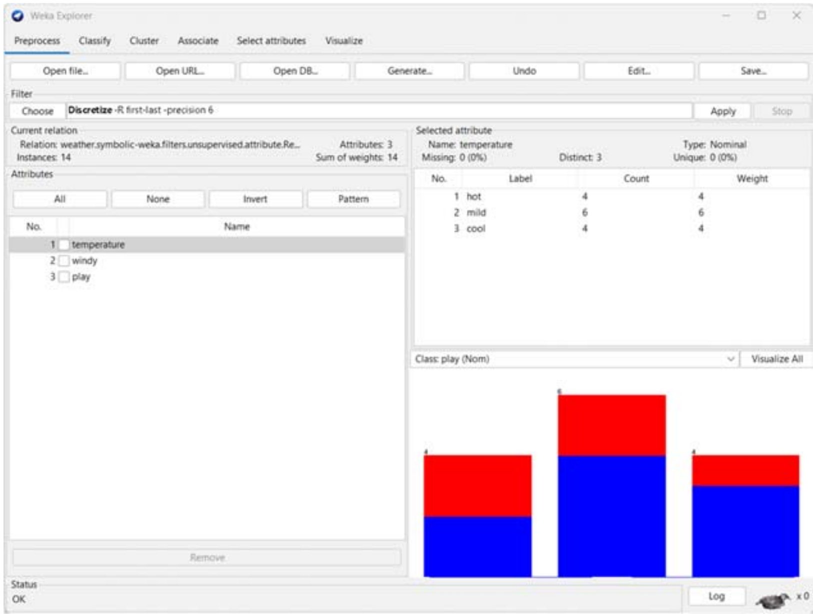


- To remove Attribute/s select them and click on the **Remove** button at the bottom.

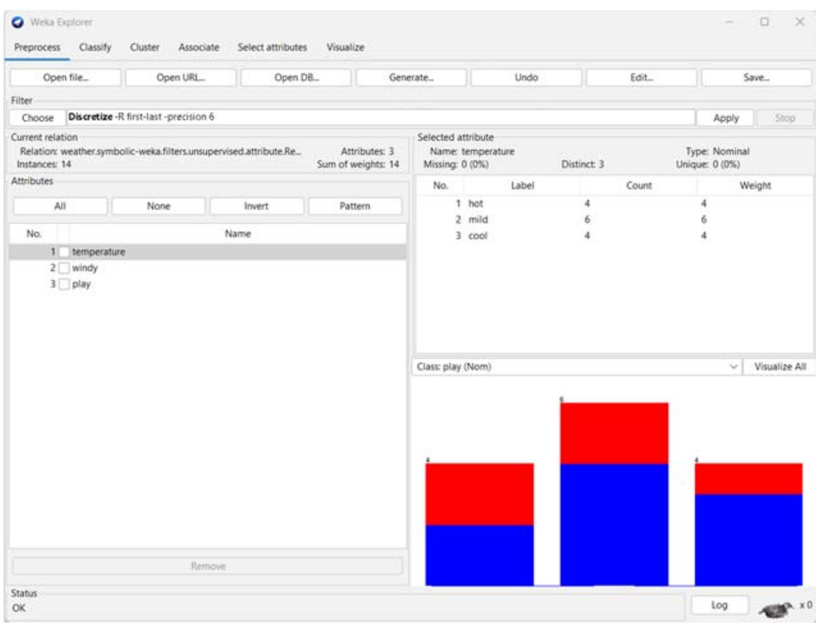




- To illustrate the use of filters, we will use **weather-numeric.arff** database that contains two **numeric** attributes - **temperature** and **humidity**.
- We will convert these to **nominal** by applying a filter on our raw data. Click on the **Choose** button in the **Filter** subwindow and select the following filter: **weka->filters->supervised->attribute->Discretize**



- Click on the **Apply** button and examine the **temperature** and/or **humidity** attribute. You will notice that these have changed from numeric to nominal types.

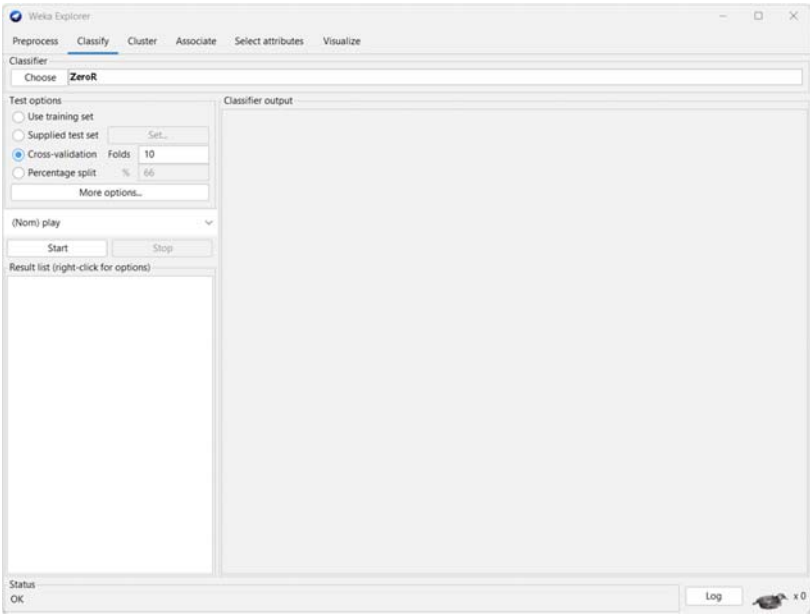


Experiment 12

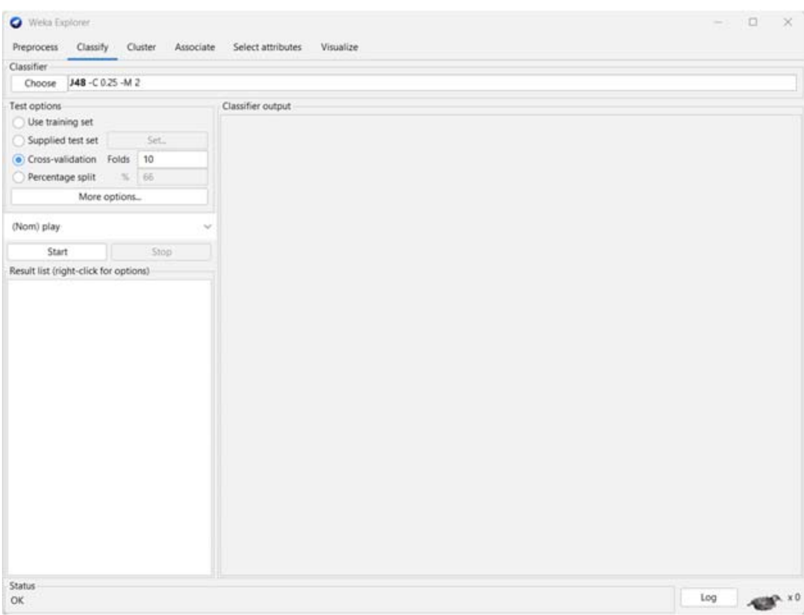
Aim: Implement classification in Weka.

Steps:

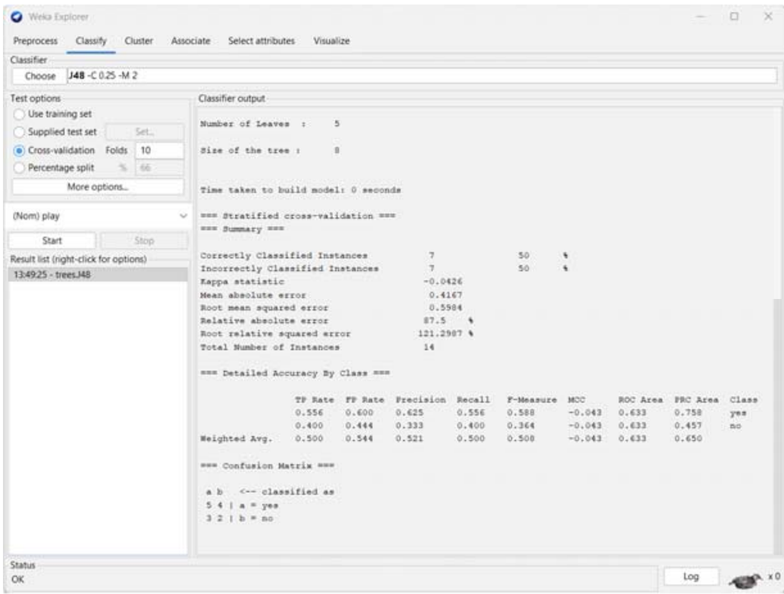
- We will use the pre-processed weather data file from the previous experiment. Open the saved file by using the **Open file ...** option under the **Preprocess** tab, click on the **Classify** tab, and you would see the following screen:



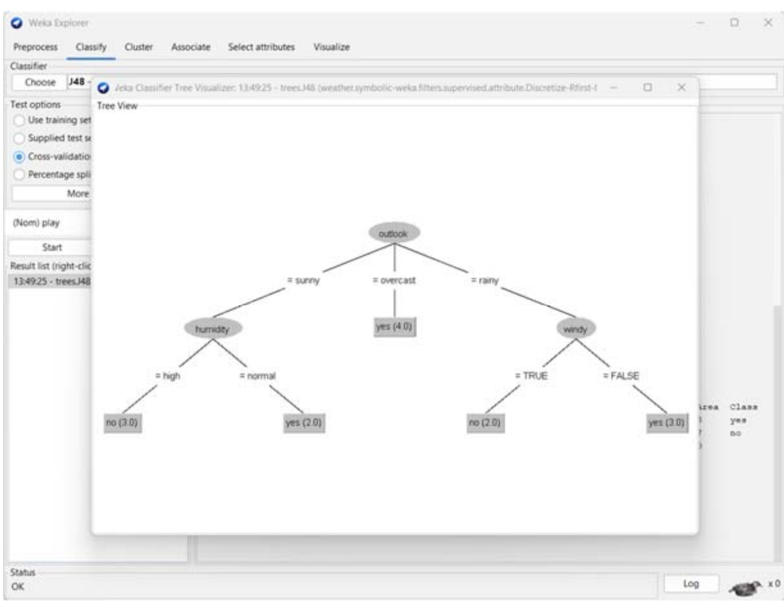
- Keep the default **play** option for the output class:



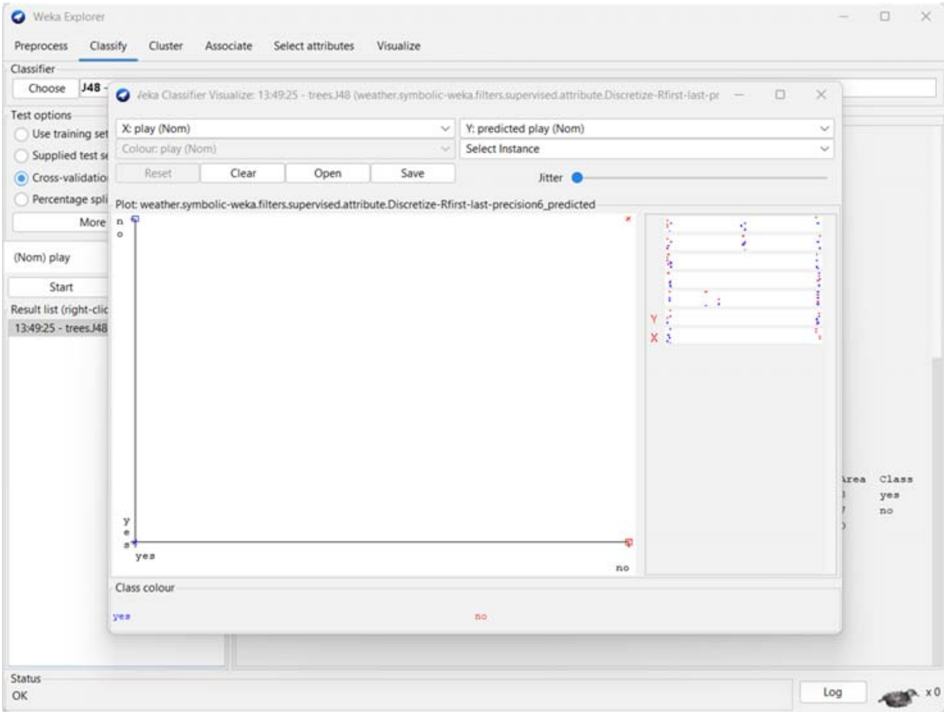
- Click on the **Choose** button and select the following classifier: **weka->classifiers>trees>J48**
- Click on the **Start** button to start the classification process. After a while, the classification results would be presented on your screen as shown here:



- To see the visual representation of the results, right click on the result in the **Result list** box. Several options would pop up on the screen.
- Select **Visualize tree** to get a visual representation of the traversal tree as seen in the screenshot below:



- Selecting **Visualize classifier errors** would plot the results of classification as shown here:

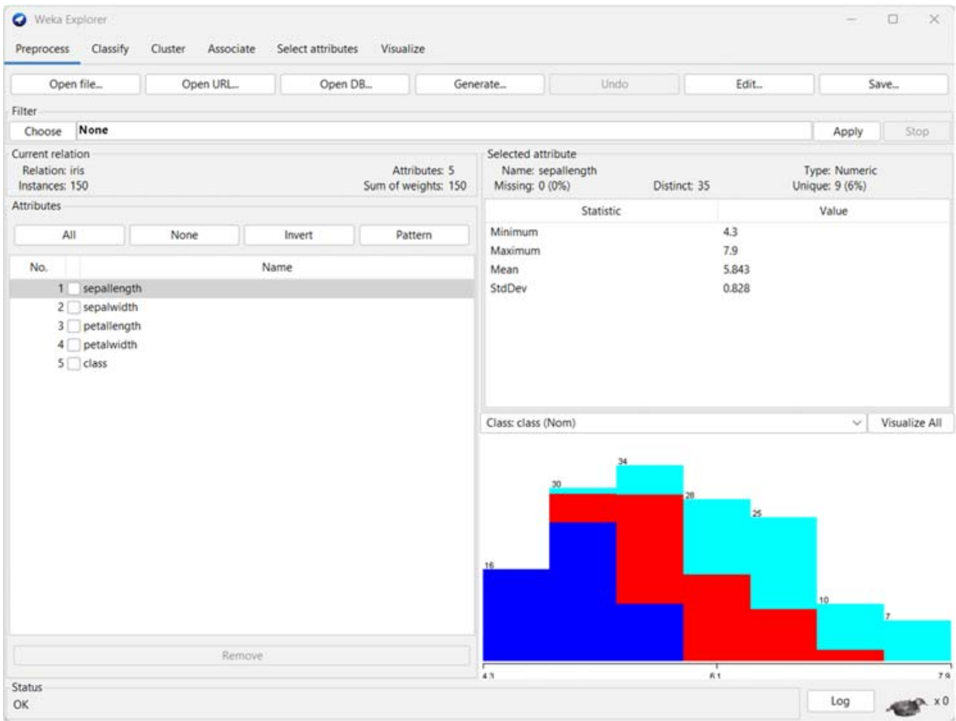


Experiment 13

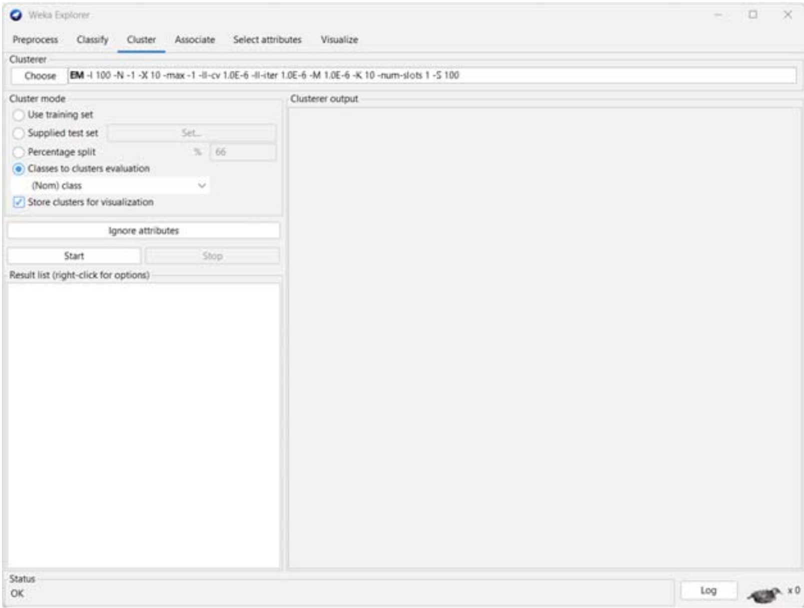
Aim: Implement clustering in Weka.

Steps:

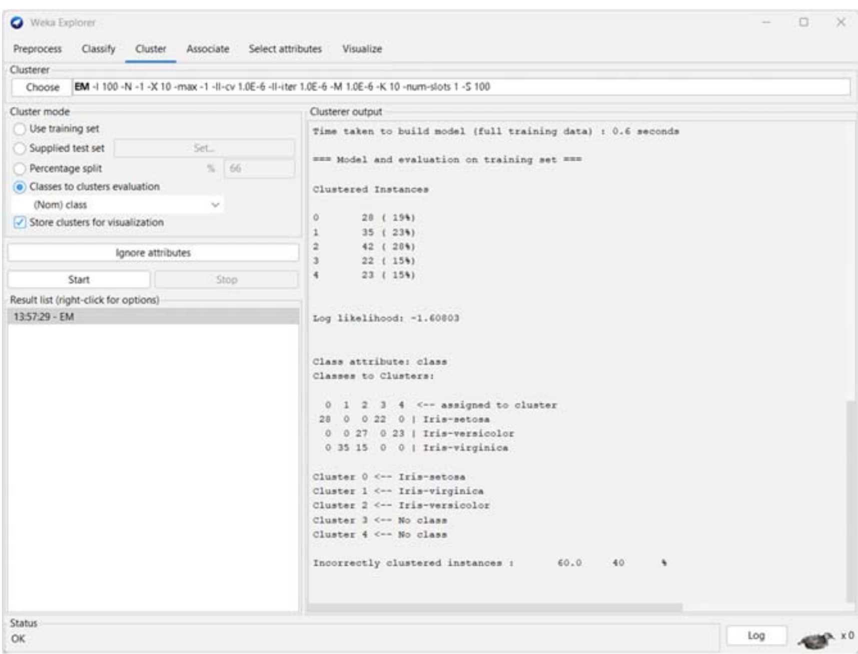
- In the WEKA explorer select the **Preprocess** tab. Click on the **Open file ...** option and select the **iris.arff** file in the file selection dialog. When you load the data, the screen looks like as shown below:



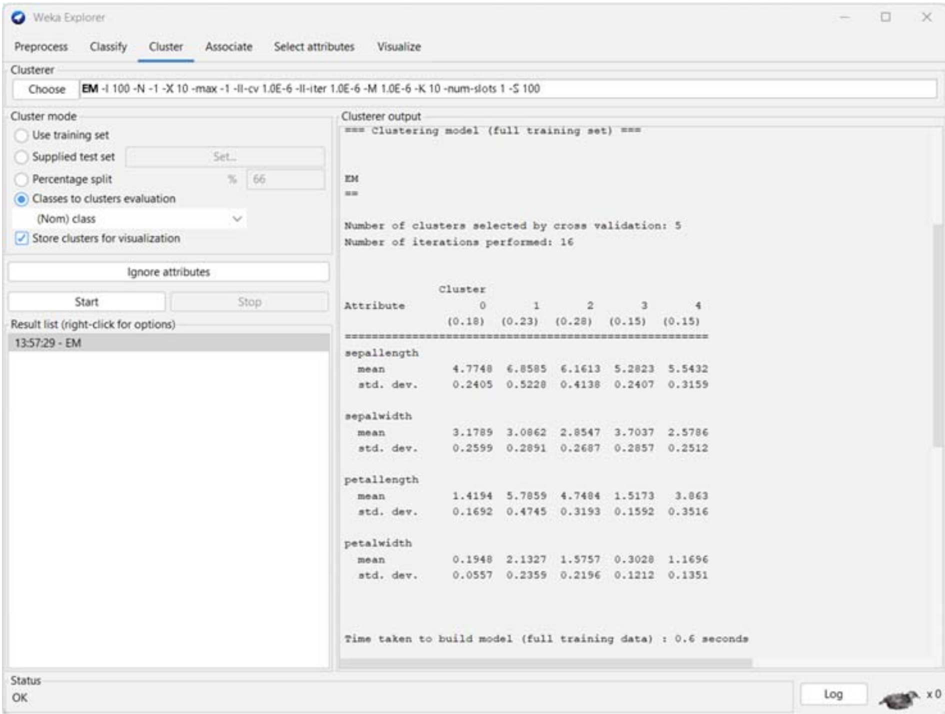
- Click on the **Cluster** TAB to apply the clustering algorithms to our loaded data. Click on the **Choose** button.
- Now, select **EM** as the clustering algorithm. In the **Cluster mode** sub window, select the **Classes to clusters evaluation** option as shown in the screenshot below:



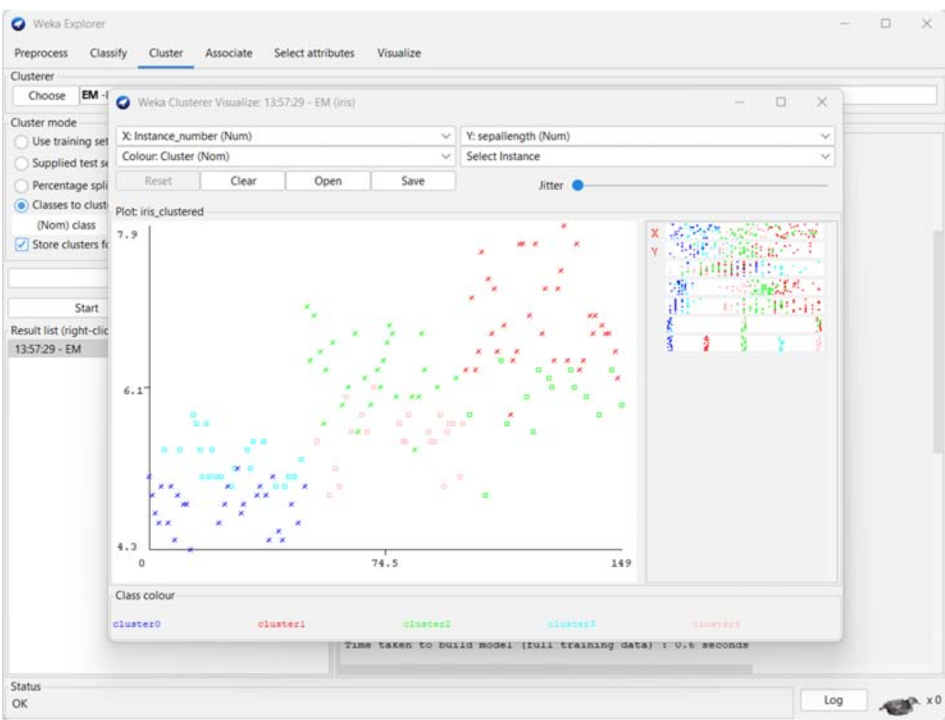
- Click on the **Start** button to process the data. After a while, the results will be presented on the screen.
- The output of the data processing is shown in the screen below:



- If you scroll up the output window, you will also see some statistics that gives the mean and standard deviation for each of the attributes in the various detected clusters. This is shown in the screenshot given below:



- Select **Visualize cluster assignments**. You will see the following output:

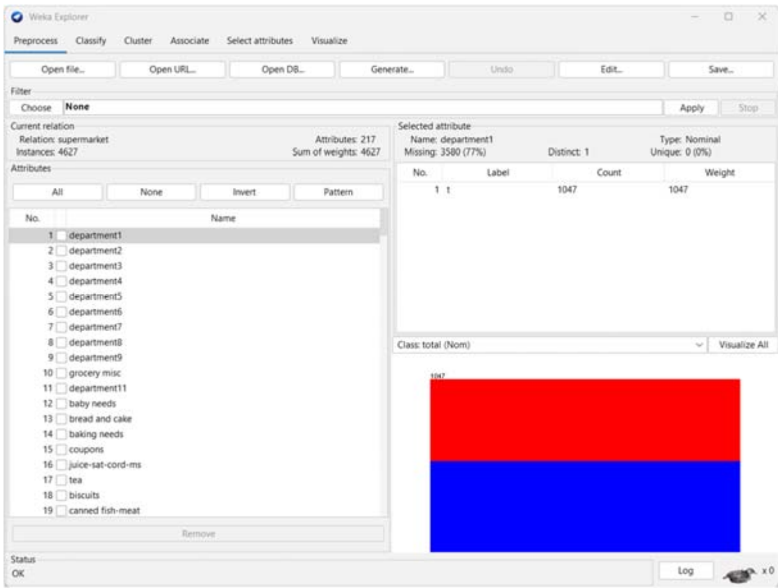


Experiment 14

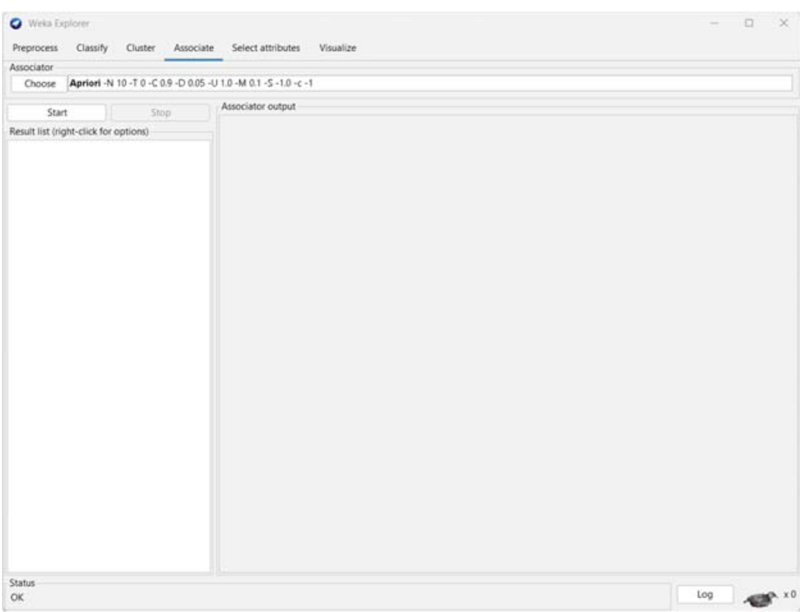
Aim: Implement association analysis in Weka.

Steps:

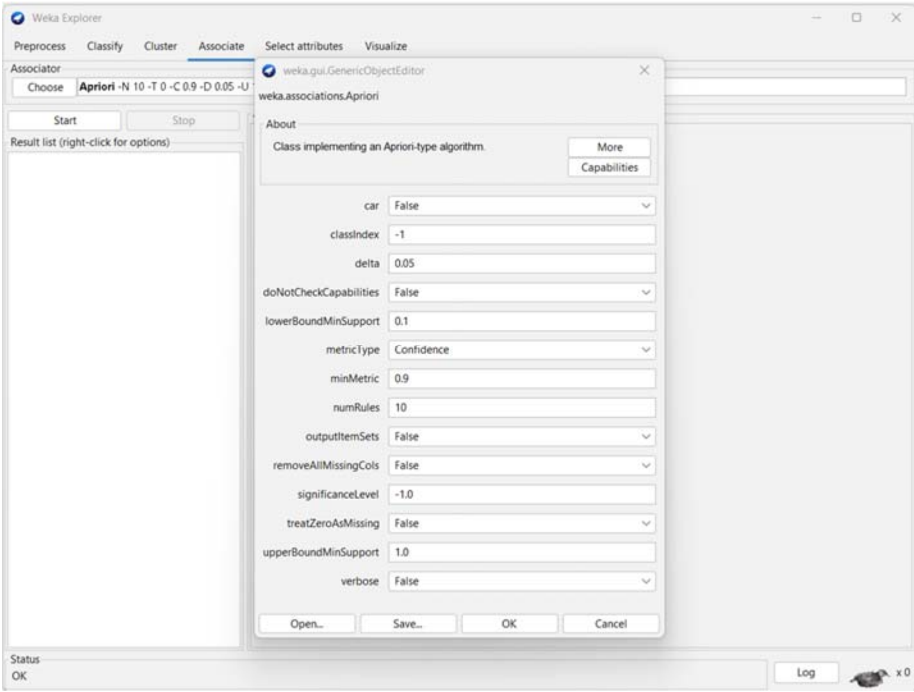
- In the WEKA explorer, open the **Preprocess** tab, click on the **Open file ...** button and select **supermarket.arff** database from the installation folder. After the data is loaded you will see the following screen:



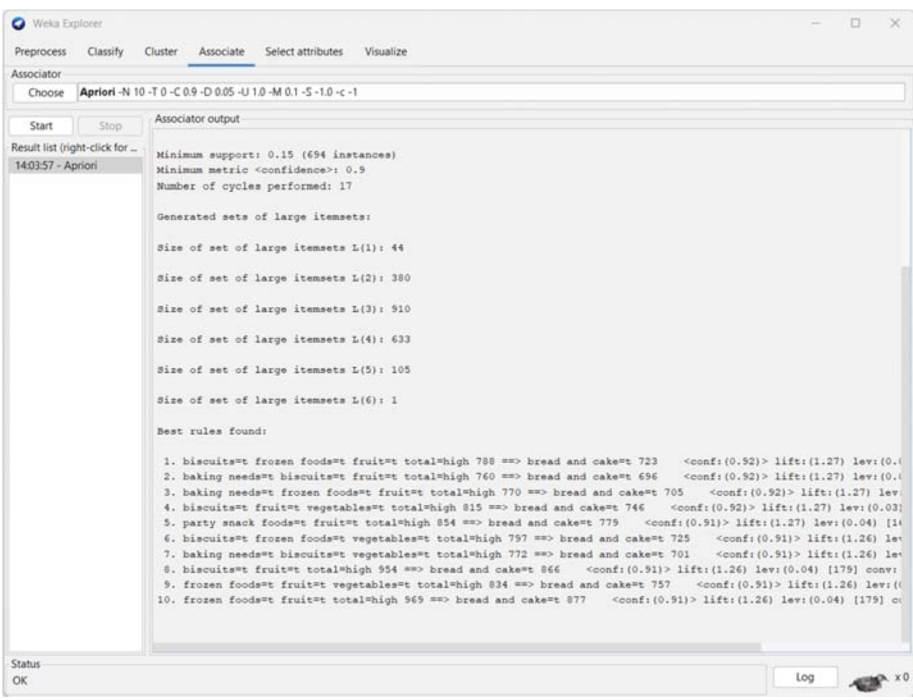
- Click on the **Associate** TAB and click on the **Choose** button. Select the **Apriori** association as shown in the screenshot:



- To set the parameters for the Apriori algorithm, click on its name, a window will pop up as shown below that allows you to set the parameters:



- After you set the parameters, click the **Start** button. After a while you will see the results as shown in the screenshot below:

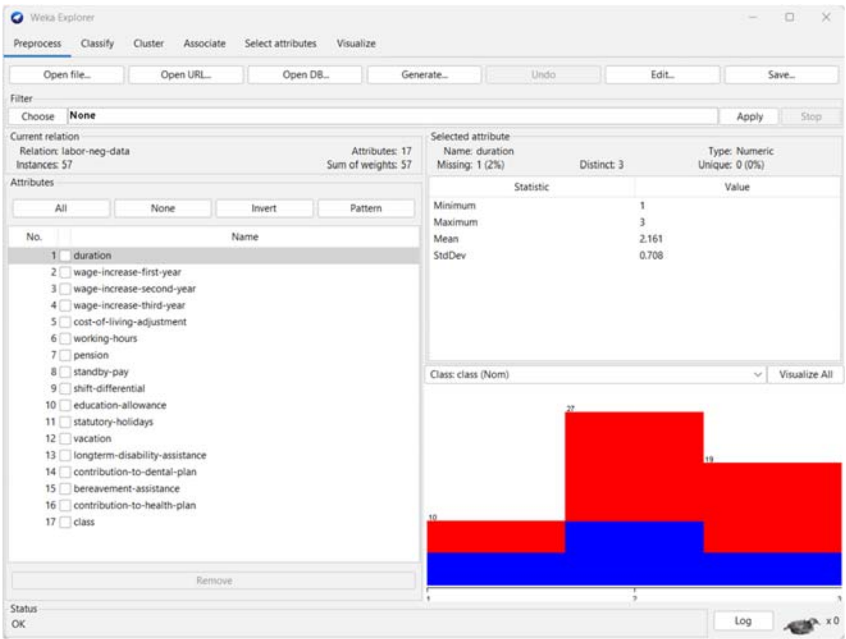


Experiment 15

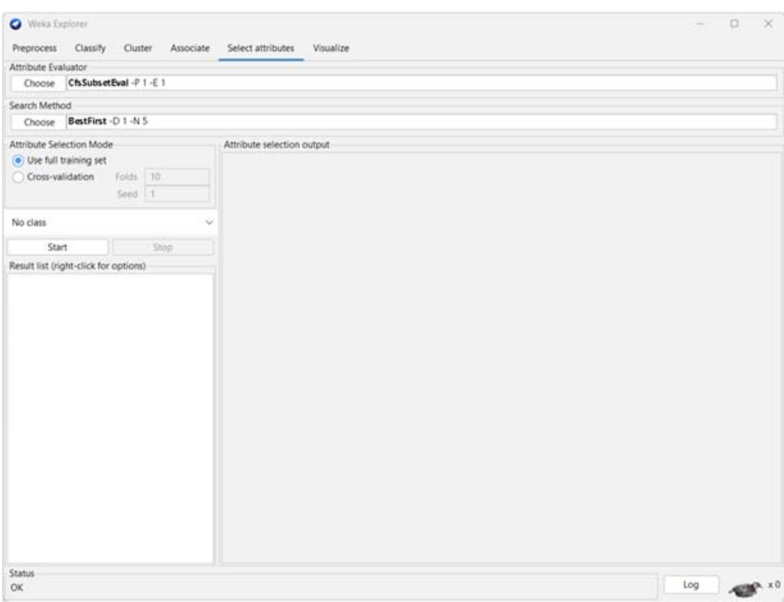
Aim: Implement attribute selection in Weka.

Steps:

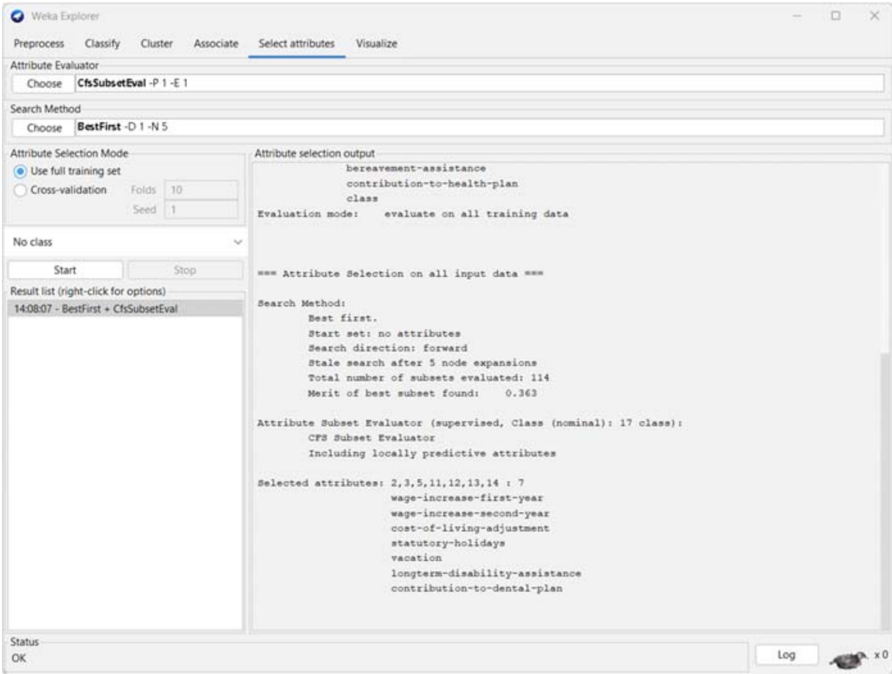
- In the **Preprocess** tag of the WEKA explorer, select the **labor.arff** file for loading into the system. When you load the data, you will see the following screen:



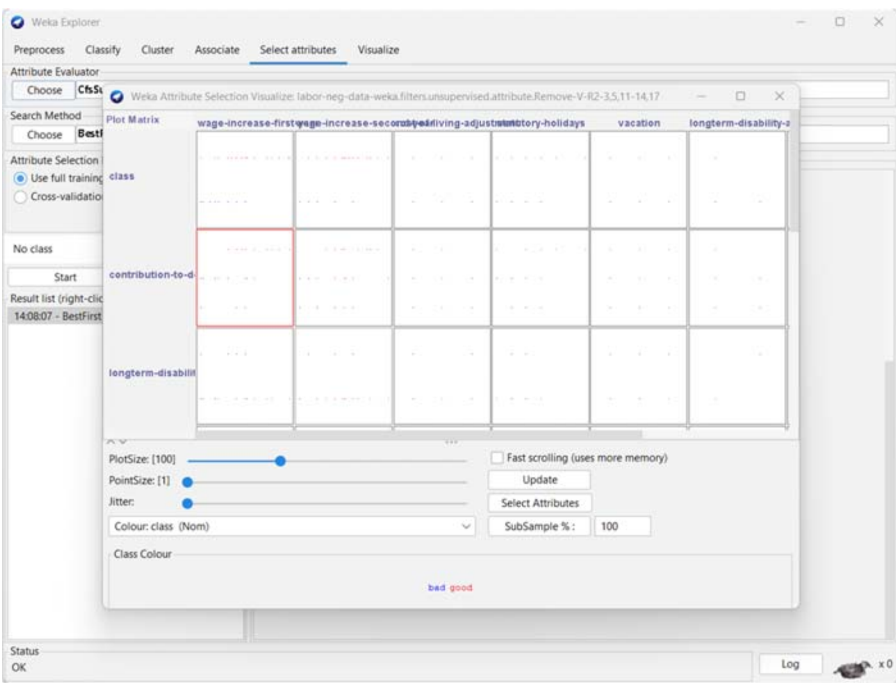
- Click on the **Select attributes** TAB.
- Under the **Attribute Evaluator** and **Search Method**, you will find several options. We will just use the defaults here. In the **Attribute Selection Mode**, use full training set option.



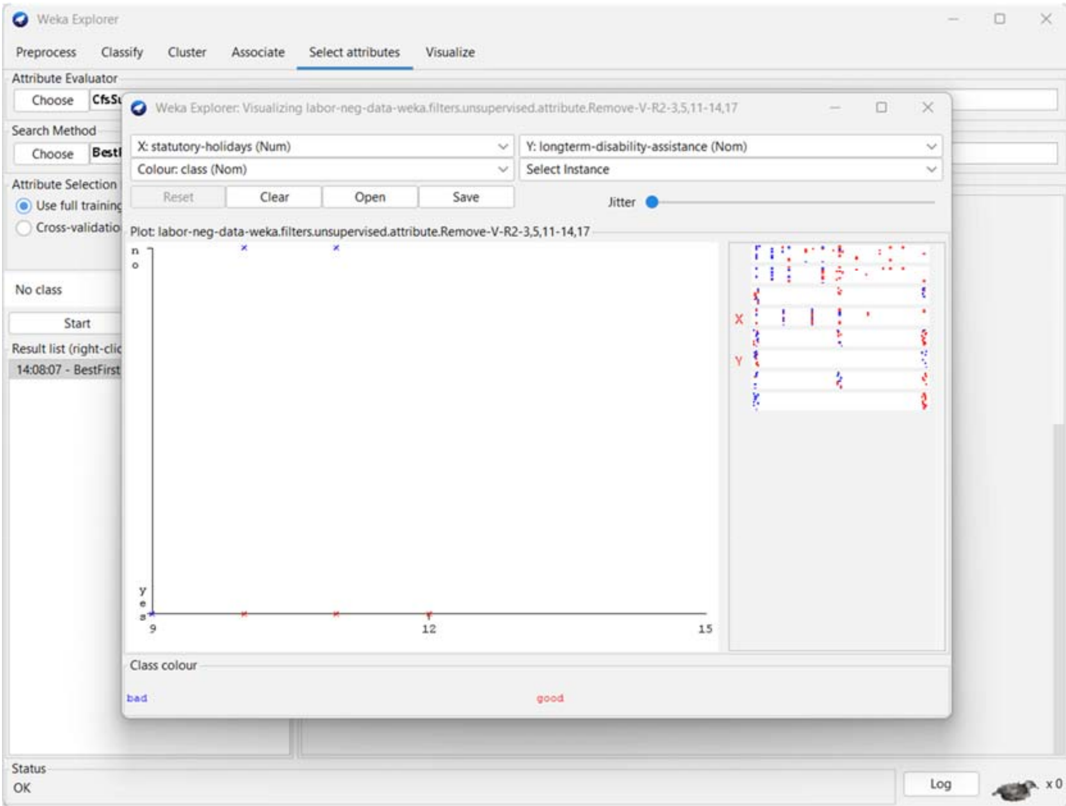
- Click on the **Start** button to process the dataset. You will see the following output:



- At the bottom of the result window, you will get the list of **Selected** attributes. To get the visual representation, right click on the result in the **Result** list.



- Clicking on any of the squares will give you the data plot for your further analysis. A typical data plot is shown below:

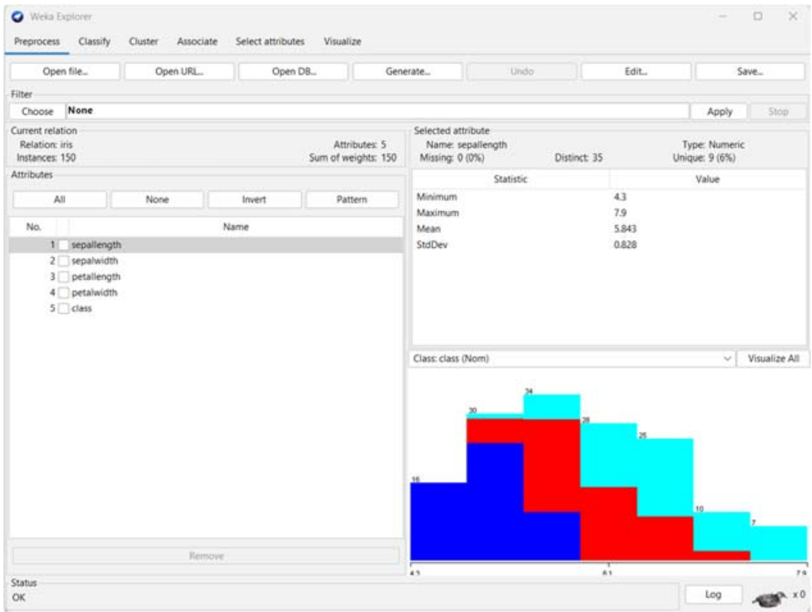


Experiment 16

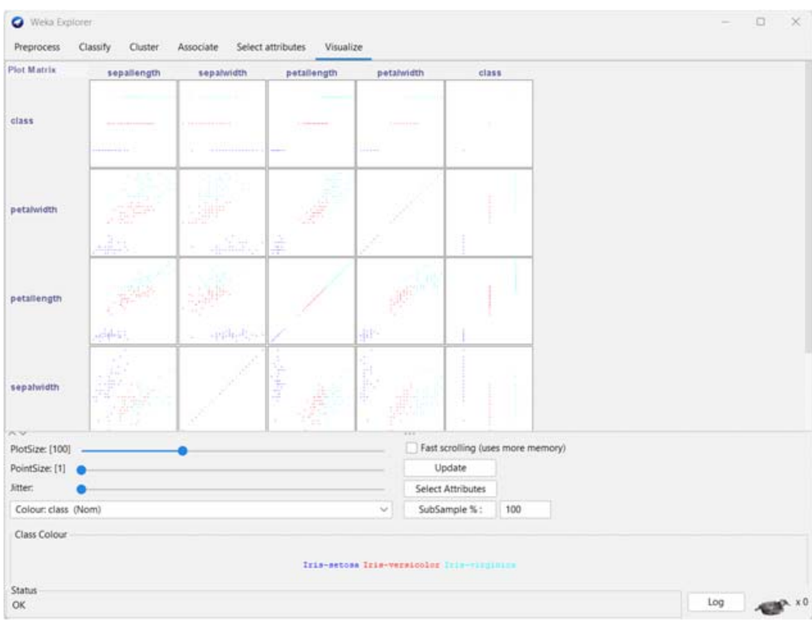
Aim: Implement data visualization in Weka.

Steps:

- Go to the Preprocess tab and open IRIS.arff dataset.

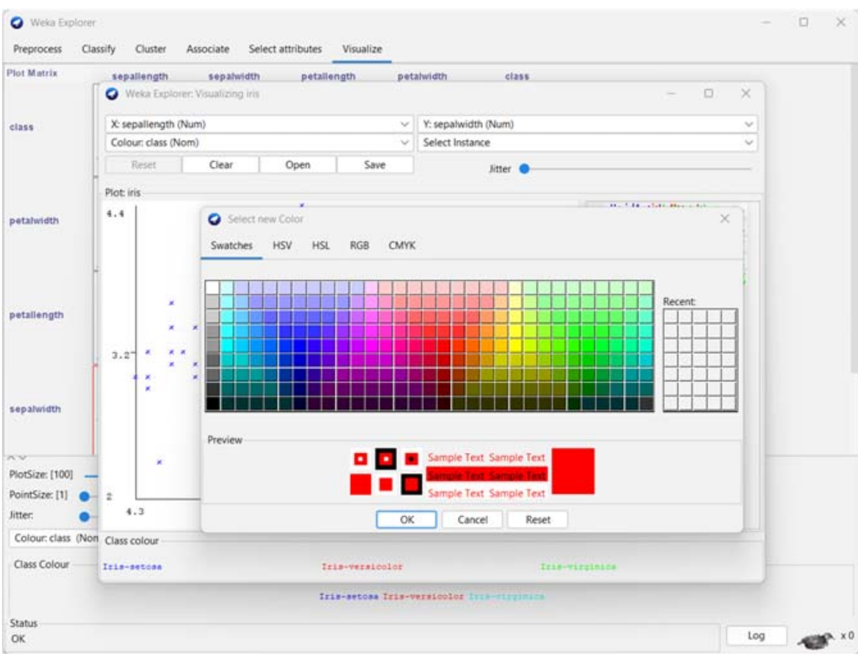


- To visualize the dataset, go to the Visualize tab. The tab shows the attributes plot matrix. The dataset attributes are marked on the x-axis and y-axis while the instances are plotted. The box with the x-axis attribute and y-axis attribute can be enlarged.



- Click on the box of the plot to enlarge. For example, x: petal length and y:petalwidth. The class labels are represented in different colors.

- These colors can be changed. To change the color, click on the class label at the bottom, and a color window will appear.



- Click on the instance represented by 'x' in the plot. It will give the instance details. For example: