MUFFAKHAM JAH COLLEGE OF ENGINEERING AND TECHNOLOGY

(Affiliated to Osmania University and Recognized by AICTE)

Mount Pleasant, 8-2-249, Road No. 3, Banjara Hills,Hyderabad, Telangana-500034.

DEPARTMENT OF COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE (CS&AI)

DATA SCIENCE LAB (PC453AD)

LAB MANUAL

B.E IV SEM (2021-2022)

I N D E X

Experiment – 1

Calculator Application

R operators – R has many operators to carry out different mathematical and logical operations. Operators in R can mainly be classified into the following categories.

1. Arithmetic operators - +, -, x, /
2. Assignment operators – <- , ->
3. Relational operators - <, >, ==, !=, <=, >=
4. Logical operators - !, &, &&, |, ||

We use the four fundamental arithmetic operations of mathematics for building a calculator application. Those functions are –

1. Addition
2. Subtraction
3. Multiplication
4. Division

User-defined Functions in R – In R programming, **user-defined functions** are functions that are created by the user for a specific use that the already built-in functions of R don't provide.

Syntax - functionName <- function (arguments) {

commands to perform

}

Parameters –
functionname: every function is generally given a name
function(argument): here the variables are mentioned
commands to perform: the block of code is written here.

1. **Aim:** To implement Calculator Application in R

a. Using with and without R objects on console
b. Using mathematical functions on console
c. Write an R script, to create an R object for calculator application and save in a specified location in disk.

Program:

```
1+2
3-1
4*2
5*2
a<-1
b<-4
c<-2
a+b
a-b
a*b
b/c
add<-function(x,y)
{
print(x+y)
}
add(2,3)
subt<-function(x,y)
{
print(x-y)
}
subt(7,2)
mul<-function(x,y)
{
print(x*y)
}
mul(6,3)
div<-function(x,y)
{
print(x/y);
}
div(10,2)
choice=readline(prompt="Enter add for addition
subt for subtraction
mul for multiplication
div for division
Choice: ");
num1=readline(prompt = "Enter first number : ");
```

num2=readline(prompt = "Enter second number : ");
num1=as.integer(num1)
num2=as.integer(num2)
cal<-switch(choice,"add"=print(num1+num2),
"subt"=print(num1-num2),
"mul"=print(num1*num2),
"div"=print(num1/num2))

Output –

```
> 7+6
[1] 13
> 32-30
[1] 2
> 4*2
[1] 8
> 5*22
[1] 110
> a<-1
> b<-2
> c<-2
> a+b
[1] 3
> a-b
[1] -1
> a*b
[1] 2
> b/c
[1] 1
> add = function(x, y) {
+   return(x + y)
+ }
> subtract = function(x, y) {
+   return(x - y)
+ }
> multiply = function(x, y) {
+   return(x * y)
+ }
> divide = function(x, y) {
+   return(x / y)
+ }
>
> print("Select operation.")
[1] "Select operation."
> print("1.Add")
[1] "1.Add"
> print("2.Subtract")
[1] "2.Subtract"
> print("3.Multiply")
[1] "3.Multiply"
> print("4.Divide")
[1] "4.Divide"
> choice = as.integer(readline(prompt="Enter choice[1/2/3/4]: "))
Enter choice[1/2/3/4]: 3
> num1 = as.integer(readline(prompt="Enter first number: "))
Enter first number: 5
> num2 = as.integer(readline(prompt="Enter second number: "))
Enter second number: 8
> operator <- switch(choice,"+","-","*","/")
> result <- switch(choice, add(num1, num2), subtract(num1, num2), multiply(num1, num2),  divide(num1,
 num2))
> print(paste(num1, operator, num2, "=", result))
[1] "5 * 8 = 40"
>
```

Experiment – 2

Descriptive Analysis

Dataset – mtcars

Description – The **mtcars** dataset is a built-in dataset in R that contains measurements on 11 aspects of automobile design and performance for 32 cars. The data was extracted from the 1974 *Motor Trend* US magazine.

Attributes –

1. Cyl
2. Disp
3. Hp
4. Drat
5. Wt
6. Qsec
7. Vs
8. Am
9. Gear
10. Carb

Dataset – cars

Description – This dataset contains 50 observations of 2 variables. It shows various readings on "speed" and "distance" collected.

Attributes –

1. speed
2. distance

Dataset – iris

Description – The data set contains 3 classes of 50 instances each, where each class refers to a type of iris plant. One class is linearly separable from the other 2; the latter are NOT linearly separable from each other.

Attributes -

1. sepal length in cm
2. sepal width in cm
3. petal length in cm
4. petal width in cm
5. species:
-- Iris Setosa
-- Iris Versicolour
-- Iris Virginica

Subset function –

subset() function in R programming is used to create a subset of vectors, matrices, or data frames.

Syntax – subset(x,subset,select)

Parameters –

- ***x:*** *indicates the object*
- ***subset:*** *indicates the logical expression on the basis of which subsetting has to be done*
- ***select:*** *indicates columns to select*

Aggregate function –

Aggregate functions are often used to derive descriptive statistics.

Syntax – aggregate(x, by, FUN, …, simplify=TRUE, drop=TRUE)

Parameters –

- x: R object
- by: List of variables
- FUN: Function to be applied for summary statistics
- ... : Additional arguments to be passed to FUN
- Simplify: Whether to simplify results as much as possible or not
- Drop: Whether to drop unused combinations of grouping values or not

mean() function – This will simply calculate the total mean of all the observations present in the data of that particular mentioned attribute.

min() function – This will give us the least valued observation from the data being used.

max() function - This will give us the maximum valued observation from the data being used.

summary() function – The summary of all the attributes are shown separately. The factors used in doing so are minimum value, 1st quartile, Median, Mean, 3rd Quartile, Maximum value.

2. Aim: To perform Descriptive Statistics in R

a. To write an R script to find basic descriptive statistics using summary, str, quartile function on metacars
b. To apply the above functions on cars data sets
b. To apply subset(), aggregate() functions on iris dataset.

Datasets used:

1. mtcars
2. cars
3. iris

Program :

a. Descriptive Statistics Analysis on mtcars dataset

```
data(mtcars)
head(mtcars)
tail(mtcars)
head(mtcars,10)str(mtcars)
mtcars[1]
mtcars[15]
mtcars[1:4]
mtcars[c(1,4)]
mtcars[-2]
max(mtcars$cyl)
min(mtcars$mpg)
mean(mtcars$mpg)
median(mtcars$mpg)
summary(mtcars)
```

Output:

```
> data(cars)
> head(cars,10)
   speed dist
1      4    2
2      4   10
3      7    4
4      7   22
5      8   16
6      9   10
7     10   18
8     10   26
9     10   34
10    11   17
> tail(cars,20)
   speed dist
31    17   50
32    18   42
33    18   56
34    18   76
35    18   84
36    19   36
37    19   46
38    19   68
39    20   32
40    20   48
41    20   52
```

```
> str(mtcars)
'data.frame':   32 obs. of  11 variables:
 $ mpg : num  21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 ...
 $ cyl : num  6 6 4 6 8 6 8 4 4 6 ...
 $ disp: num  160 160 108 258 360 ...
 $ hp  : num  110 110 93 110 175 105 245 62 95 123 ...
 $ drat: num  3.9 3.9 3.85 3.08 3.15 2.76 3.21 3.69 3.92 3.92 ...
 $ wt  : num  2.62 2.88 2.32 3.21 3.44 ...
 $ qsec: num  16.5 17 18.6 19.4 17 ...
 $ vs  : num  0 0 1 1 0 1 0 1 1 1 ...
 $ am  : num  1 1 1 0 0 0 0 0 0 0 ...
 $ gear: num  4 4 4 3 3 3 3 4 4 4 ...
 $ carb: num  4 4 1 1 2 1 4 2 2 4 ...
> mtcars[1:4]
                    mpg cyl  disp  hp
Mazda RX4          21.0   6 160.0 110
Mazda RX4 Wag      21.0   6 160.0 110
Datsun 710         22.8   4 108.0  93
Hornet 4 Drive     21.4   6 258.0 110
Hornet Sportabout  18.7   8 360.0 175
Valiant            18.1   6 225.0 105
Duster 360         14.3   8 360.0 245
Merc 240D          24.4   4 146.7  62
Merc 230           22.8   4 140.8  95
Merc 280           19.2   6 167.6 123
Merc 280C          17.8   6 167.6 123
Merc 450SE         16.4   8 275.8 180
Merc 450SL         17.3   8 275.8 180
Merc 450SLC        15.2   8 275.8 180
Cadillac Fleetwood 10.4   8 472.0 205
```

```
> mtcars[c(1,4)]
                     mpg  hp
Mazda RX4           21.0 110
Mazda RX4 Wag       21.0 110
Datsun 710          22.8  93
Hornet 4 Drive      21.4 110
Hornet Sportabout   18.7 175
Valiant             18.1 105
Duster 360          14.3 245
Merc 240D           24.4  62
Merc 230            22.8  95
Merc 280            19.2 123
Merc 280C           17.8 123
Merc 450SE          16.4 180
Merc 450SL          17.3 180
Merc 450SLC         15.2 180
Cadillac Fleetwood  10.4 205
Lincoln Continental 10.4 215
Chrysler Imperial   14.7 230
Fiat 128            32.4  66
Honda Civic         30.4  52
Toyota Corolla      33.9  65
Toyota Corona       21.5  97
Dodge Challenger    15.5 150
AMC Javelin         15.2 150
Camaro Z28          13.3 245
Pontiac Firebird    19.2 175
Fiat X1-9           27.3  66
Porsche 914-2       26.0  91
```

```
> mtcars[-2]
                     mpg  disp  hp drat    wt  qsec vs am gear carb
Mazda RX4           21.0 160.0 110 3.90 2.620 16.46  0  1    4    4
Mazda RX4 Wag       21.0 160.0 110 3.90 2.875 17.02  0  1    4    4
Datsun 710          22.8 108.0  93 3.85 2.320 18.61  1  1    4    1
Hornet 4 Drive      21.4 258.0 110 3.08 3.215 19.44  1  0    3    1
Hornet Sportabout   18.7 360.0 175 3.15 3.440 17.02  0  0    3    2
Valiant             18.1 225.0 105 2.76 3.460 20.22  1  0    3    1
Duster 360          14.3 360.0 245 3.21 3.570 15.84  0  0    3    4
Merc 240D           24.4 146.7  62 3.69 3.190 20.00  1  0    4    2
Merc 230            22.8 140.8  95 3.92 3.150 22.90  1  0    4    2
Merc 280            19.2 167.6 123 3.92 3.440 18.30  1  0    4    4
Merc 280C           17.8 167.6 123 3.92 3.440 18.90  1  0    4    4
Merc 450SE          16.4 275.8 180 3.07 4.070 17.40  0  0    3    3
Merc 450SL          17.3 275.8 180 3.07 3.730 17.60  0  0    3    3
Merc 450SLC         15.2 275.8 180 3.07 3.780 18.00  0  0    3    3
Cadillac Fleetwood  10.4 472.0 205 2.93 5.250 17.98  0  0    3    4
Lincoln Continental 10.4 460.0 215 3.00 5.424 17.82  0  0    3    4
Chrysler Imperial   14.7 440.0 230 3.23 5.345 17.42  0  0    3    4
Fiat 128            32.4  78.7  66 4.08 2.200 19.47  1  1    4    1
Honda Civic         30.4  75.7  52 4.93 1.615 18.52  1  1    4    2
Toyota Corolla      33.9  71.1  65 4.22 1.835 19.90  1  1    4    1
Toyota Corona       21.5 120.1  97 3.70 2.465 20.01  1  0    3    1
Dodge Challenger    15.5 318.0 150 2.76 3.520 16.87  0  0    3    2
AMC Javelin         15.2 304.0 150 3.15 3.435 17.30  0  0    3    2
Camaro Z28          13.3 350.0 245 3.73 3.840 15.41  0  0    3    4
Pontiac Firebird    19.2 400.0 175 3.08 3.845 17.05  0  0    3    2
Fiat X1-9           27.3  79.0  66 4.08 1.935 18.90  1  1    4    1
Porsche 914-2       26.0 120.3  91 4.43 2.140 16.70  0  1    5    2
```

```
> max(mtcars$cyl)
[1] 8
> min(mtcars$mpg)
[1] 10.4
> mean(mtcars$mpg)
[1] 20.09062
> median(mtcars$mgp)
NULL
> summary(mtcars)
      mpg             cyl             disp             hp             drat      
 Min.   :10.40   Min.   :4.000   Min.   : 71.1   Min.   : 52.0   Min.   :2.760  
 1st Qu.:15.43   1st Qu.:4.000   1st Qu.:120.8   1st Qu.: 96.5   1st Qu.:3.080  
 Median :19.20   Median :6.000   Median :196.3   Median :123.0   Median :3.695  
 Mean   :20.09   Mean   :6.188   Mean   :230.7   Mean   :146.7   Mean   :3.597  
 3rd Qu.:22.80   3rd Qu.:8.000   3rd Qu.:326.0   3rd Qu.:180.0   3rd Qu.:3.920  
 Max.   :33.90   Max.   :8.000   Max.   :472.0   Max.   :335.0   Max.   :4.930  
       wt             qsec             vs               am              gear      
 Min.   :1.513   Min.   :14.50   Min.   :0.0000   Min.   :0.0000   Min.   :3.000  
 1st Qu.:2.581   1st Qu.:16.89   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:3.000  
 Median :3.325   Median :17.71   Median :0.0000   Median :0.0000   Median :4.000  
 Mean   :3.217   Mean   :17.85   Mean   :0.4375   Mean   :0.4062   Mean   :3.688  
 3rd Qu.:3.610   3rd Qu.:18.90   3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.:4.000  
 Max.   :5.424   Max.   :22.90   Max.   :1.0000   Max.   :1.0000   Max.   :5.000  
      carb      
 Min.   :1.000  
 1st Qu.:2.000  
 Median :2.000  
 Mean   :2.812  
 3rd Qu.:4.000  
 Max.   :8.000  
```

b. Descriptive Statistics Analysis on cars dataset

data(cars)
head(cars,10)
tail(cars,20)
str(cars)
head(cars)
max(cars)
max(cars$speed)
min(cars$speed)
mean(cars$speed)
median(cars$speed)
mode(cars$speed)
summary(cars$speed)
summary(cars)

Output:

```
> data(cars)
> head(cars,10)
   speed dist
1      4    2
2      4   10
3      7    4
4      7   22
5      8   16
6      9   10
7     10   18
8     10   26
9     10   34
10    11   17
> tail(cars,20)
   speed dist
31    17   50
32    18   42
33    18   56
34    18   76
35    18   84
36    19   36
37    19   46
38    19   68
39    20   32
40    20   48
41    20   52
42    20   56
43    20   64
44    22   66
45    23   54
46    24   70
```

```
42    20   56
43    20   64
44    22   66
45    23   54
46    24   70
47    24   92
48    24   93
49    24  120
50    25   85
> str(cars)
'data.frame':   50 obs. of  2 variables:
 $ speed: num  4 4 7 7 8 9 10 10 10 11 ...
 $ dist : num  2 10 4 22 16 10 18 26 34 17 ...
> head(cars)
  speed dist
1     4    2
2     4   10
3     7    4
4     7   22
5     8   16
6     9   10
> max(cars)
[1] 120
> max(cars$speed)
[1] 25
```

```
> min(cars$speed)
[1] 4
> mean(cars$speed)
[1] 15.4
> median(cars$speed)
[1] 15
> mode(cars$speed)
[1] "numeric"
> summary(cars$speed)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
    4.0    12.0    15.0    15.4    19.0    25.0 
> summary(cars)
     speed           dist       
 Min.   : 4.0   Min.   :  2.00  
 1st Qu.:12.0   1st Qu.: 26.00  
 Median :15.0   Median : 36.00  
 Mean   :15.4   Mean   : 42.98  
 3rd Qu.:19.0   3rd Qu.: 56.00  
 Max.   :25.0   Max.   :120.00  
```

c. Applying subset and aggregate functions on iris dataset

data(iris)
head(iris)
tail(iris)
subset(iris,Sepal.Length==6.1)
aggregate(.~Species,data=iris,mean)

Output –

```
> data(iris)
> head(iris)
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
4          4.6         3.1          1.5         0.2  setosa
5          5.0         3.6          1.4         0.2  setosa
6          5.4         3.9          1.7         0.4  setosa
> tail(iris)
    Sepal.Length Sepal.Width Petal.Length Petal.Width   Species
145          6.7         3.3          5.7         2.5 virginica
146          6.7         3.0          5.2         2.3 virginica
147          6.3         2.5          5.0         1.9 virginica
148          6.5         3.0          5.2         2.0 virginica
149          6.2         3.4          5.4         2.3 virginica
150          5.9         3.0          5.1         1.8 virginica
> subset(iris, Sepal.Length==6.1)
    Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
64           6.1         2.9          4.7         1.4 versicolor
72           6.1         2.8          4.0         1.3 versicolor
74           6.1         2.8          4.7         1.2 versicolor
92           6.1         3.0          4.6         1.4 versicolor
128          6.1         3.0          4.9         1.8  virginica
135          6.1         2.6          5.6         1.4  virginica
> aggregate(.~Species, data=iris, mean)
     Species Sepal.Length Sepal.Width Petal.Length Petal.Width
1     setosa        5.006       3.428        1.462       0.246
2 versicolor        5.936       2.770        4.260       1.326
3  virginica        6.588       2.974        5.552       2.026
```

Experiment – 3

Reading and writing different types of data

Package used – readxl

The readxl package makes it easy to get data out of Excel and into R. Compared to many of the existing packages, readxl has no external dependencies, so it's easy to install and use on all operating systems. It is designed to work with tabular data.

Functions for Reading Data into R –

There are a few very useful functions for reading data into R.

1. **read.table()** and **read.csv()** are two popular functions used for reading tabular data into R.
2. **readLines()** is used for reading lines from a text file.
3. **source()** is a very useful function for reading in R code files from a another R program.
4. **dget()** function is also used for reading in R code files.
5. **load()** function is used for reading in saved workspaces
6. **unserialize()** function is used for reading single R objects in binary format.

Functions for Writing Data to Files –

There are similar functions for writing data to files

1. **write.table()** is used for writing tabular data to text files (i.e. CSV).
2. **read.delim()** is **used to read delimited text files in the R Language**.
3. **writeLines()** function is useful for writing character data line-by-line to a file or connection.
4. **dump()** is a function for dumping a textual representation of multiple R objects.
5. **dput()** function is used for outputting a textual representation of an R object.
6. **serialize()** is used for converting an R object into a binary format for outputting to a connection .

3. Aim: To read and write different types of datasets

a. To read different types of datasets from web and disk and writing in file in specific disk location.
b. To read Excel data sheet in R.

```
name=c("a","b","c","d","e")
marks=c(20,30,40,10,15)
id=c(1:5)
st=data.frame(id,name,marks)
View(st)

#1. writing data frame into CSV file
write.csv(student,"student.csv",row.names=FALSE)

#2. reading CSV file
st1=read.csv("student.csv")
View(st1)

#3.writing data frame to a text file
write.table(st1,file="st1.txt",quote=F,row.names=F)

#4. reading from text
st2=read.delim('st1.txt')
View(st2)

#5. reading a file from web
webfile = read.delim("http://www.sthda.com/upload/boxplot_format.txt")
print(webfile)
head(webfile)
write.table(webfile,file="webfile.txt",quote=F,row.names=FALSE)

# install package readxl first

install.packages("readxl")
library(readxl)

#6. reading excel datasheet
df=read_excel("d:/ex1.xlsx",sheet=2)
View(df)
```

Output:

```
> name=c("a","b","c","d","e")
> marks=c(20,30,40,10,15)
> id=c(1:5)
> st=data.frame(id,name,marks)
> View(st)
> #1. writing data frame into CSV file
> write.csv(student,"student.csv",row.names=FALSE)
```

	id	name	marks
1	1	a	20
2	2	b	30
3	3	c	40
4	4	d	10
5	5	e	15

```
> #2. reading CSV file
> st1=read.csv("student.csv")
> View(st1)
```

	id	name	marks
1	1	a	20
2	2	b	30
3	3	c	40
4	4	d	10
5	5	e	15

```
> #3.writing data frame to a text file
> write.table(st1,file="st1.txt",quote=F,row.names=F)
>
> #4. reading from text
> st2=read.delim('st1.txt')
> View(st2)
```

	id.name.marks
1	1 a 20
2	2 b 30
3	3 c 40
4	4 d 10
5	5 e 15

```
#5. reading a file from web
> webfile = read.delim("http://www.sthda.com/upload/boxplot_format.txt")
> print(webfile)
```

```
     Nom variable Group
1  IND1       10     A
2  IND2        7     A
3  IND3       20     A
4  IND4       14     A
5  IND5       14     A
6  IND6       12     A
7  IND7       10     A
8  IND8       23     A
9  IND9       17     A
10 IND10      20     A
11 IND11      14     A
12 IND12      13     A
13 IND13      11     B
14 IND14      17     B
15 IND15      21     B
16 IND16      11     B
17 IND17      16     B
18 IND18      14     B
19 IND19      17     B
20 IND20      17     B
21 IND21      19     B
22 IND22      21     B
23 IND23       7     B
24 IND24      13     B
25 IND25       0     C
26 IND26       1     C
27 IND27       7     C
28 IND28       2     C
29 IND29       3     C
30 IND30       1     C
31 IND31       2     C
32 IND32       1     C
33 IND33       3     C
34 IND34       0     C
35 IND35       1     C
36 IND36       4     C
37 IND37       3     D
38 IND38       5     D
39 IND39      12     D
40 IND40       6     D
41 IND41       4     D
42 IND42       3     D
43 IND43       5     D
44 IND44       5     D
45 IND45       5     D
46 IND46       5     D
47 IND47       2     D
48 IND48       4     D
49 IND49       3     E
50 IND50       5     E
51 IND51       3     E
52 IND52       5     E
53 IND53       3     E
54 IND54       6     E
55 IND55       1     E
56 IND56       1     E
57 IND57       3     E
```

```
58 IND58       2    E
59 IND59       6    E
60 IND60       4    E
61 IND61      11    F
62 IND62       9    F
63 IND63      15    F
64 IND64      22    F
65 IND65      15    F
66 IND66      16    F
67 IND67      13    F
68 IND68      10    F
69 IND69      26    F
70 IND70      26    F
71 IND71      24    F
72 IND72      13    F
> head(webfile)
   Nom variable Group
1 IND1       10     A
2 IND2        7     A
3 IND3       20     A
4 IND4       14     A
5 IND5       14     A
6 IND6       12     A
> write.table(webfile,file="webfile.txt",quote=F,row.names=FALSE)
 >
> # install package readxl first
>
> install.packages("readxl")
Error in install.packages : Updating loaded packages
> library(readxl)
>
> #6. reading excel datasheet
> df=read_excel("d:/ex1.xlsx",sheet=2)
> View(df)
```

	rollno	marks
1	1	11
2	2	22
3	3	33
4	4	44

Experiment – 4
Visualization

Data visualization is an efficient technique for gaining insight about data through a visual medium. With the help of visualization techniques, we can easily obtain information about hidden patterns in data and also we can work with large datasets to efficiently obtain key insights.

Dataset used – mtcars

Description – The **mtcars** dataset is a built-in dataset in R that contains measurements on 11 aspects of automobile design and performance for 32 cars. The data was extracted from the 1974 *Motor Trend* US magazine.

Attributes –

1. Cyl
2. Disp
3. Hp
4. Drat
5. Wt
6. Qsec
7. Vs
8. Am
9. Gear
10. Carb

Package - ggplot2

R allows us to create graphics declaratively. This package is famous for its elegant and qualitygraphs, which sets it apart from other visualization packages.

Boxplot – boxplot() function is used to create a boxplot. These are a measure of how well data is distributed across a data set. This graph represents the minimum, maximum, average, first quartile,and the third quartile in the data set.

Syntax – boxplot(x, data, names, main)

parameters –

- **x** is a vector or a formula.
- **data** is the data frame.
- **names** are the group labels which will be printed under each boxplot.
- **main** is used to give a title to the graph.

Scatterplot –

The scatter plots are used to compare variables. A comparison between variables is requiredwhen we need to define how much one variable is affected by another variable.

Syntax – plot(x, y, main, xlab, ylab)

Parameters –

- **x** is the data set whose values are the horizontal coordinates.

- **y** is the data set whose values are the vertical coordinates.
- **main** is the tile of the graph.
- **xlab** is the label in the horizontal axis.
- **ylab** is the label in the vertical axis.

Outliers using plots –

An outlier is a point or set of points that are different from other points. Sometimes they can be very high or very low. It's often a good idea to detect and remove the outliers. Because outliers are one of the primary reasons for resulting in a less accurate model. Often outliers can be seen with visualizations using a box plot.

R Histogram

A histogram is a type of bar chart which shows the frequency of the number of values which are compared with a set of values ranges. For creating a histogram, R provides hist() function. The histogram is used for the distribution.

Syntax - hist(v,main,xlab,col)

Parameters –

- **v** is a vector containing numeric values used in histogram.
- **main** indicates title of the chart.
- **xlab** is used to give description of x-axis.
- **col** is used to set color of the bars.

R Bar Charts

A bar chart is a pictorial representation in which numerical values of variables are represented by length or height of lines or rectangles of equal width. R provides the barplot() function.

Syntax – barplot(H, xlab, ylab, main, names.arg, col)

Parameters –

- **H** is a vector or matrix containing numeric values used in bar chart.
- **xlab** is the label for x axis.
- **ylab** is the label for y axis.
- **main** is the title of the bar chart.
- **names.arg** is a vector of names appearing under each bar.
- **col** is used to give colours to the bars in the graph.

R Pie Charts

A pie-chart is arepresentation of values in the form of slices of a circle with different colors. Pie charts are created with the help of pie () function, which takes positive numbers as vector input.

Syntax - pie(x, labels, main, col)

Parameters –

- **x** is a vector containing the numeric values used in the pie chart.
- **labels** is used to give description to the slices.
- **main** indicates the title of the chart.
- **col** indicates the colour palette.

4. Aim: To perform visualizations

a. To find the data distribution using box and scatter plot

b. To find the outliers using plot.

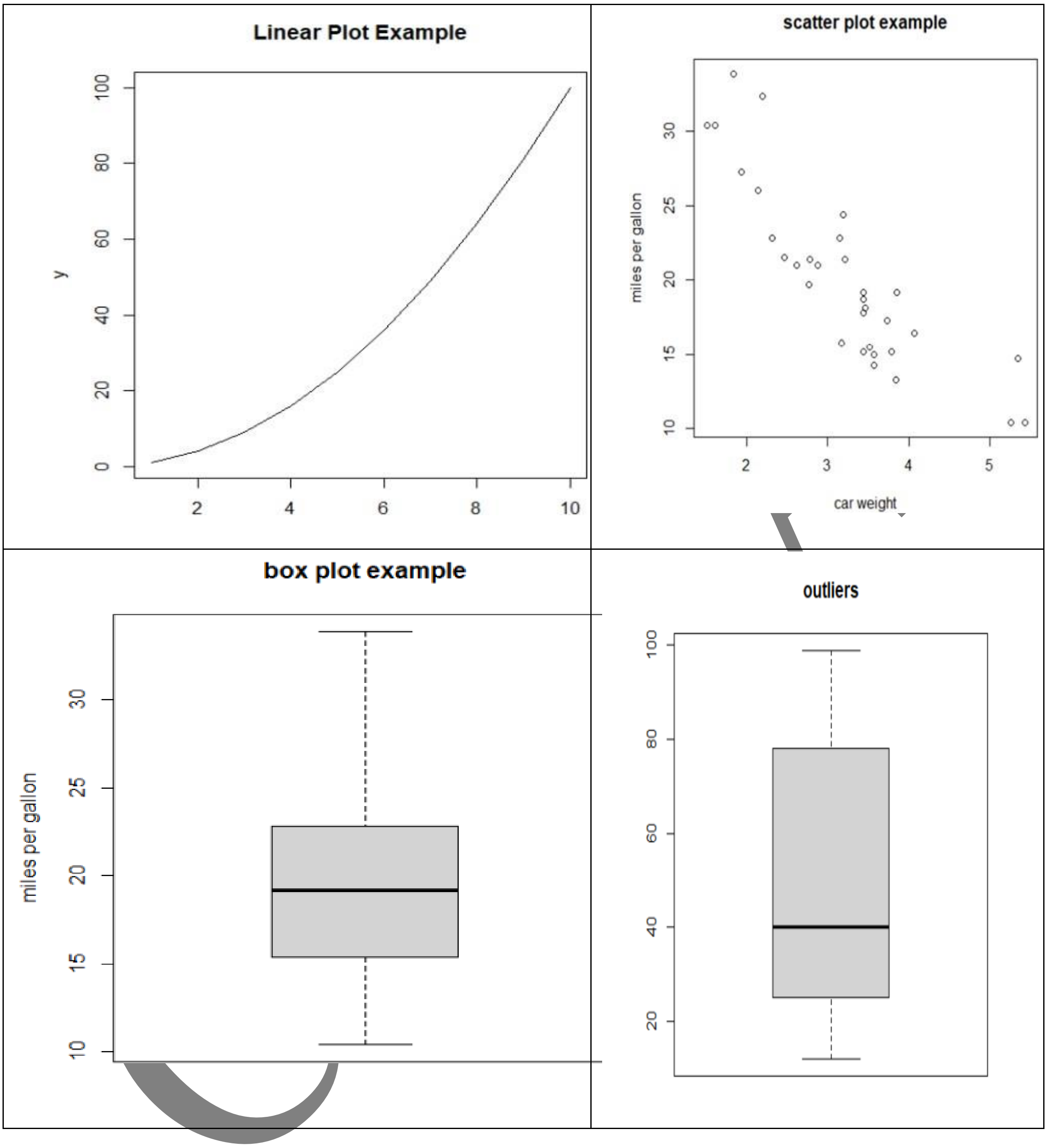
c. To plot the histogram, bar chart and pie chart on sample data.

Dataset used: mtcars

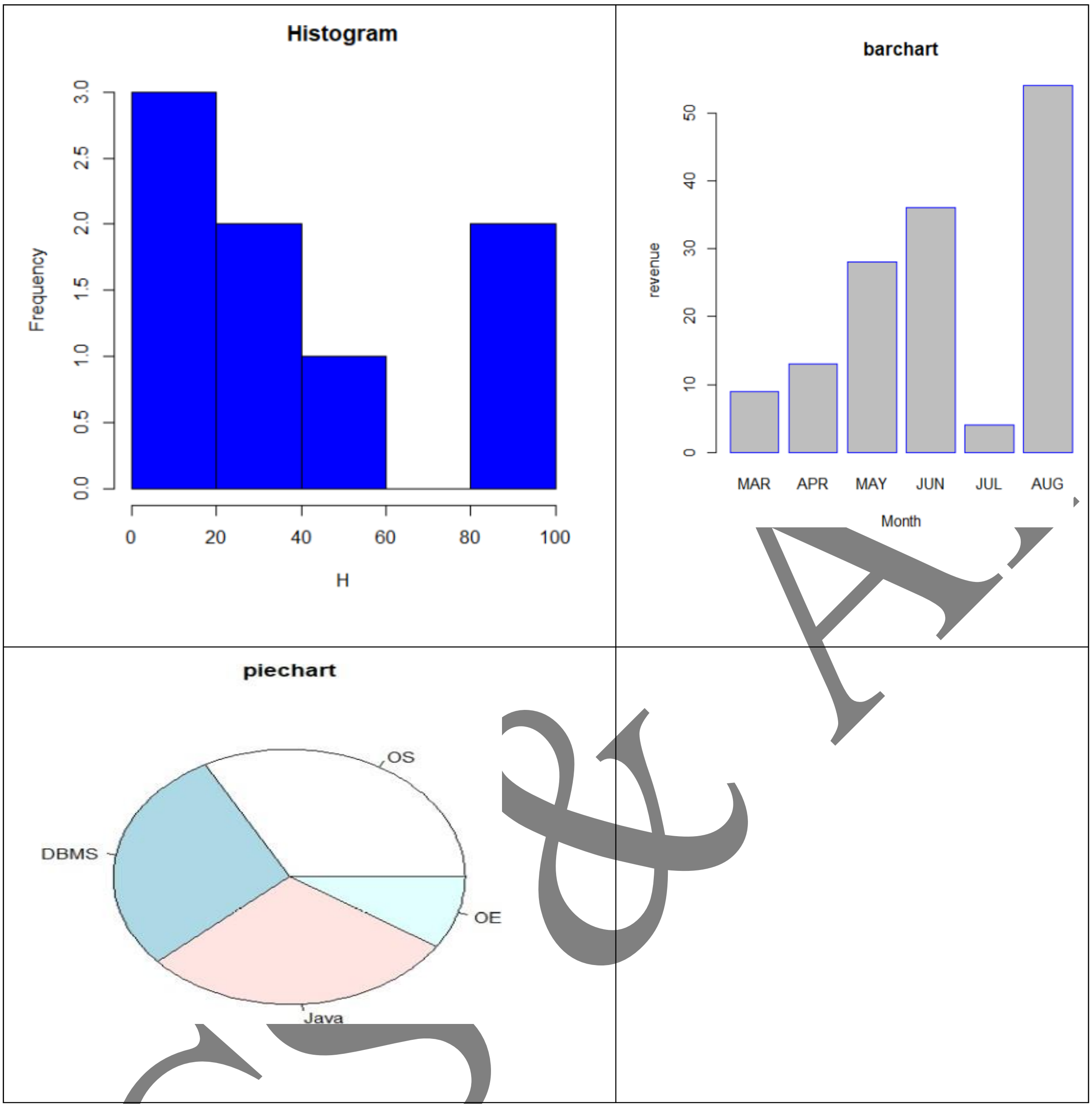
Program:

```
#Linear plot
x=1:10
y=x^2
plot(x,y,type="l",main=”Linear Plot Example”)
#installing package
install.packages("ggplot2")
#scatter plot
data("mtcars")
plot(
mtcars$wt,mtcars$mpg,
main = "scatter plot example",
xlab = "car weight",
ylab="miles per gallon",
)
#box plot
data("mtcars")
boxplot(
mtcars$mpg,
main = "box plot example",
ylab="miles per gallon"
)
#outliers
v<-c(50,25,30,12,78,99)
boxplot(v,main="outliers")
#Histogram
H<-c(9,13,28,36,4,54,99,98)
hist(H,main="Histogram",col="blue")
#Barchart
h<-c(9,13,28,36,4,54)
m<-c("MAR","APR","MAY","JUN","JUL","AUG")
barplot(h,names.arg=m,xlab="Month",ylab="revenue",main="barchart",border ="blue")
#pie chart
h<-c(90,78,80,25)
m<-c("OS","DBMS","Java","OE")
pie(h,m,main = "piechart")
```

Output:



Linear Plot Example
y
0
20
40
60
80
100
2
4
6
8
10
scatter plot example
miles per gallon
10
15
20
25
30
2
3
4
5
car weight
box plot example
miles per gallon
10
15
20
25
30
outliers
20
40
60
80
100


Histogram
Frequency
0.0
0.5
1.0
1.5
2.0
2.5
3.0
0
20
40
60
80
100
H
barchart
revenue
0
10
20
30
40
50
MAR
APR
MAY
JUN
JUL
AUG
Month
piechart
OS
DBMS
OE
Java

Experiment – 5

Correlation and Covariance

Correlation and Covariance are terms used in statistics to measure relationships between two random variables. Both of these terms measure linear dependency between a pair ofrandom variables or bivariate data.

Correlation in R Programming Language –

cor() function in R programming measures the correlation coefficient value. Correlationis a relationship term in statistics that uses the covariance method to measure how strong the vectors are related. Mathematically,

$$\mathrm{Corr}(x,y) = \frac{\sum(x_i-\bar{x})(y_i-\bar{y})}{\sqrt{\sum(x_i-\bar{x})^2 \sum(y_i-\bar{y})^2}}$$

where,

x *represents the x data vector*

y represents the y data vector

Syntax: *cor(x, y, method)*

where,

- ***x*** *and* ***y*** *represents the data vectors*
- ***method*** *defines the type of method to be used to compute covariance.*

Covariance in R Programming Language –

In R programming, covariance can be measured using **cov()** function. Covariance is a statistical term used to measures the direction of the linear relationship between the data vectors. Mathematically,

$$\mathrm{Cov}(x,y) = \frac{\Sigma(x_i-\bar{x})(y_i-\bar{y})}{N}$$

Syntax: *cov(x, y, method)*
where,

- ***x*** *and* ***y*** *represents the data vectors*
- ***method*** *defines the type of method to be used to compute covariance.*
- ***N*** *represents total observations*

Package – CORRPLOT()

R package **corrplot** provides a visual exploratory tool on correlation matrix that supports automatic variable reordering to help detect hidden patterns among variables.

corrplot is very easy to use and provides a rich array of plotting options invisualization method, graphic layout, color, legend, text labels, etc. It also provides p-values and confidence intervals to help users determine the statistical significance of the correlations.

corrplot() - The mostly using parameters include method, type, order, diag, and etc.

Correlation matrix –

A correlation matrix is a table of correlation coefficients for a set of variables used to determine if a relationship exists between the variables. The coefficient indicates both the strength of the relationship as well as the direction.

Syntax: *cor (x, use = , method =   )*
Parameters:

- ***x:*** *It is a numeric matrix or a data frame.*
- ***use:*** *Deals with missing data.*
- ***method:*** *Deals with a type of relationship*

Dataset - iris

Description –

The data set contains 3 classes of 50 instances each, where each class refers to a type of iris plant. One class is linearly separable from the other 2; the latter are NOT linearly separable from each other.

Attributes -

1. sepal length in cm
2. sepal width in cm
3. petal length in cm
4. petal width in cm
5. species:
-- Iris Setosa
-- Iris Versicolour
-- Iris Virginica

Variance (ANOVA) –

Analysis of variance (ANOVA) is an analysis tool used in statistics that splits an observed aggregate variability found inside a data set into two parts: systematic factors and random factors. The systematic factors have a statistical influence on the given data set, while the random factors do not.

5. Aim: To Calculate Correlation and Covariance

a. To find the correlation matrix.
b. To plot the correlation plot on the dataset and visualize giving an overview of relationships among data on iris data.
c. To analysis of covariance: variance (ANOVA), if data have categorical variables on iris data.

Dataset used: iris

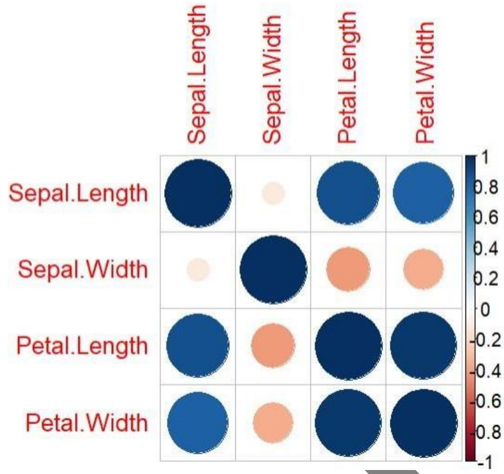
Program:

```
install.packages('corrplot')
x<-rnorm(2)
x
y<-rnorm(2)
y
mat<-cbind(x,y)
mat
cor(mat)
cov(mat)
data(iris)
iris
mydata<-iris[,c(1,2,3,4)]
mydata
str(mydata)
d1<-cor(mydata)
d1
library(corrplot)
corrplot(d1,method="circle")
color<-c('red','green','blue','black')
pairs(mydata,col=color,bg=color,pch=21)
cov(iris$Petal.Length,iris$Petal.Width
```

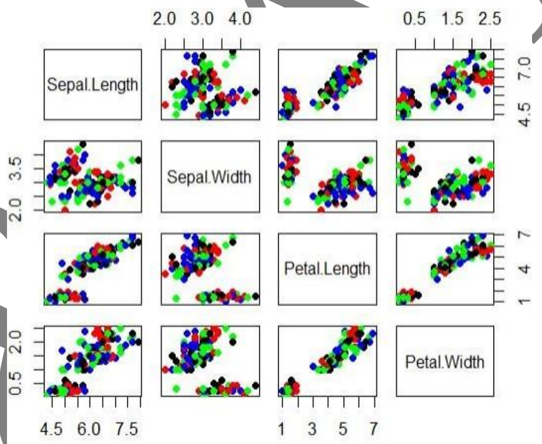
Output:

```
> #install.packages('corrplot')
> x<-rnorm(2)
> x
[1] 0.6471771 1.4214339
> y<-rnorm(2)
> y
[1] 0.61047662 0.06848282
> mat<-cbind(x,y)
> mat
             x          y
[1,] 0.6471771 0.61047662
[2,] 1.4214339 0.06848282
> cor(mat)
   x  y
x  1 -1
y -1  1
> cov(mat)
           x          y
x  0.2997368 -0.2098212
y -0.2098212  0.1468786
> data(iris)
> iris
    Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
1            5.1         3.5          1.4         0.2     setosa
2            4.9         3.0          1.4         0.2     setosa
3            4.7         3.2          1.3         0.2     setosa
4            4.6         3.1          1.5         0.2     setosa
5            5.0         3.6          1.4         0.2     setosa
6            5.4         3.9          1.7         0.4     setosa
7            4.6         3.4          1.4         0.3     setosa
```

```
> mydata<-iris[,c(1,2,3,4)]
> mydata
   Sepal.Length Sepal.Width Petal.Length Petal.Width
1           5.1         3.5          1.4         0.2
2           4.9         3.0          1.4         0.2
3           4.7         3.2          1.3         0.2
4           4.6         3.1          1.5         0.2
5           5.0         3.6          1.4         0.2
6           5.4         3.9          1.7         0.4
7           4.6         3.4          1.4         0.3
8           5.0         3.4          1.5         0.2
9           4.4         2.9          1.4         0.2
10          4.9         3.1          1.5         0.1
11          5.4         3.7          1.5         0.2
12          4.8         3.4          1.6         0.2
13          4.8         3.0          1.4         0.1
14          4.3         3.0          1.1         0.1
15          5.8         4.0          1.2         0.2
16          5.7         4.4          1.5         0.4
17          5.4         3.9          1.3         0.4
18          5.1         3.5          1.4         0.3
19          5.7         3.8          1.7         0.3
20          5.1         3.8          1.5         0.3
21          5.4         3.4          1.7         0.2
22          5.1         3.7          1.5         0.4
23          4.6         3.6          1.0         0.2
24          5.1         3.3          1.7         0.5
25          4.8         3.4          1.9         0.2
26          5.0         3.0          1.6         0.2
```




```
> str(mydata)
'data.frame':	150 obs. of  4 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 ...
> d1<-cor(mydata)
> d1
             Sepal.Length Sepal.Width Petal.Length Petal.Width
Sepal.Length    1.0000000  -0.1175698    0.8717538   0.8179411
Sepal.Width    -0.1175698   1.0000000   -0.4284401  -0.3661259
Petal.Length    0.8717538  -0.4284401    1.0000000   0.9628654
Petal.Width     0.8179411  -0.3661259    0.9628654   1.0000000
> library(corrplot)
> corrplot(d1,method="circle")
> color<-c('red','green','blue','black')
> pairs(mydata,col=color,bg=color,pch=21)
> cov(iris$Petal.Length,iris$Petal.Width)
[1] 1.295609
> 
```



Experiment – 6

Regression Model

Dataset – crashdata.csv

Description – This dataset has 80 observations of 6 variables.

Attributes –

1. ManHI
2. ManBI
3. IntI
4. HVACi
5. Safety
6. CarType

Dataset – crashdataset.csv
Description – This dataset has 20 observations of 6 variables.

Attributes –

1. ManHI
2. ManBI
3. IntI
4. HVACi
5. Safety
6. CarType

GLM – 'glm' is used to fit generalised linear models, specified by giving a symbolic description of the linear predictor and a description of the error distribution.

Syntax - glm (formula, family, data)

Parameters –

- Family types includes binomial, Poisson, Gaussian, gamma, quasi.
- Data: refers to the dataset being used

Package used – caret

Caret stands for classification and regression training and is arguably the biggest project in R.

One of the most powerful and popular packages is the caret library, which follows a consistent syntax for data preparation, model building, and model evaluation, making it easy for data science practitioners.

6. Aim: To evaluate the performance of Regression Model

a. Import data from web storage. Name the dataset and perform Logistic
b. Regression to find out relation between variables in the model. Also
c. check the model is fit or not [require (foreign), require(MASS)]

Datasets used are

crashdata.csv
crashdataset.csv

Program:

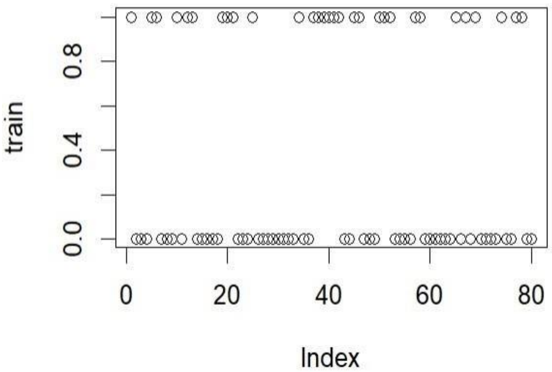
```
#logistic regression
mydata <- read.csv('crashdata.csv')
mytestdata <- read.csv('crashtestdata.csv')
mydata
mytestdata
str(mydata)
summary(mydata)
mydata[6] <- as.factor(mydata$CarType)
mydata
fit <- glm(formula=mydata$CarType~.,family='binomial', data=mydata)
fit
summary(fit)
train <- predict(fit, type='response')
plot(train)
tapply(train, mydata$CarType, mean)
pred <- predict(fit,newdata = mytestdata, type='response')
plot(pred)
mytestdata[pred<=0.5,'Predict'] <- 'Hatchback'
mytestdata[pred>0.5,'Predict'] <- 'SUV'
mytestdata
#install.packages("caret") run on console
library(caret)
confusionMatrix(table(mytestdata[,7],mytestdata[,6]),positive='Hatchback')
```

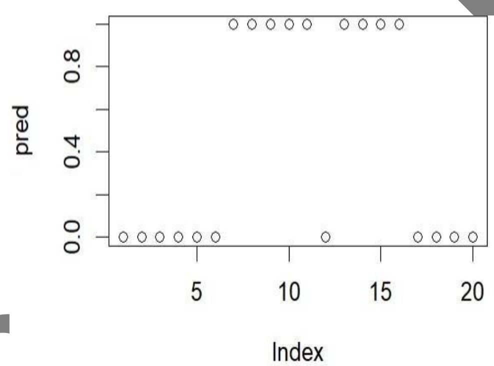

Output:

```
> #logistic regression
> mydata <- read.csv('crashdata.csv')
> mytestdata <- read.csv('crashtestdata.csv')
> mydata
   ManHI ManBI  IntI HVACi Safety   CarType
1  -5.27 -1.30  2.86 -4.85   4.04       SUV
2  -4.82 -5.38  9.72 -0.97  -4.57 Hatchback
3   9.57 -7.50 -7.61  1.33  -5.10 Hatchback
4   2.84 -2.85  0.92  5.51  -6.64 Hatchback
5   0.00  2.68 -4.15  0.85   5.58       SUV
6   0.40  6.34  0.83  5.03  -8.10       SUV
7   5.94  3.14 -6.65  6.62  -1.32 Hatchback
8   5.78 -1.75 -6.85  0.73   5.50 Hatchback
9   0.86 -4.32  8.10 -8.96   3.10 Hatchback
10  7.36  7.42  0.27 -8.62   3.08       SUV
11 -7.95 -6.44 -4.68  3.86   9.82 Hatchback
12 -1.75 -8.09 -6.06 -9.13  -6.45       SUV
13  9.44 -0.02 -9.38  0.06   6.06       SUV
14 -9.67  9.61  7.85  5.88  -8.14 Hatchback
15  6.81  3.66  4.75  0.89  -2.41 Hatchback
16 -9.93 -6.15  5.77  8.76   3.60 Hatchback
17  3.16 -6.80 -2.94  1.83   8.36 Hatchback
```

```
> mytestdata
   ManHI ManBI  IntI HVACi Safety   CarType
1   1.94  2.21  3.38  1.78  -7.19 Hatchback
2  -0.02 -3.33  0.79 -6.63   7.99       SUV
3  -0.49 -4.48  5.00  8.33  -2.77 Hatchback
4   5.76  1.35  7.92 -0.43   4.29 Hatchback
5   2.51 -8.74  4.53 -1.91   3.95 Hatchback
6  -4.47  8.42 -0.05  5.57   9.62 Hatchback
7  -9.89 -2.25 -5.00 -9.23   9.38       SUV
8  -9.94 -3.23  2.81 -2.98  -1.12       SUV
9  -8.37  4.21 -8.95  6.66   7.34       SUV
10  8.48  0.38 -3.02 -1.92  -7.43       SUV
11  0.79  0.96 -4.03 -2.28   6.20       SUV
12  5.32  2.08  5.55  7.89  -6.80 Hatchback
13 -7.26 -0.11 -5.27 -7.14   1.20       SUV
14  0.69  3.37  3.70 -5.73  -5.86       SUV
15 -5.53 -0.12  1.61  2.31  -8.66       SUV
16  8.29  1.44 -7.26  5.06  -7.00       SUV
17  9.09 -2.26  1.64  2.80  -1.22 Hatchback
18  5.04  4.52  0.28  8.26   4.59 Hatchback
19  4.55 -3.88 -2.02 -1.20  -0.42 Hatchback
20 -5.55  6.02  8.87  5.26  -2.27 Hatchback
> str(mydata)
'data.frame':   80 obs. of  6 variables:
 $ ManHI  : num  -5.27 -4.82 9.57 2.84 0 0.4 5.94 5.78 0.86 7.36 ...
 $ ManBI  : num  -1.3 -5.38 -7.5 -2.85 2.68 6.34 3.14 -1.75 -4.32 7.42 ...
 $ IntI   : num  2.86 9.72 -7.61 0.92 -4.15 0.83 -6.65 -6.85 8.1 0.27 ...
 $ HVACi  : num  -4.85 -0.97 1.33 5.51 0.85 5.03 6.62 0.73 -8.96 -8.62 ...
 $ Safety : num  4.04 -4.57 -5.1 -6.64 5.58 -8.1 -1.32 5.5 3.1 3.08 ...
 $ CarType: chr  "SUV" "Hatchback" "Hatchback" "Hatchback" ...
> fit

Call:  glm(formula = mydata$CarType ~ ., family = "binomial", data = mydata)

Coefficients:
(Intercept)        ManHI        ManBI         IntI        HVACi       Safety
     -22.76       -13.48        36.02       -44.90       -58.50       -27.36

Degrees of Freedom: 79 Total (i.e. Null);  74 Residual
Null Deviance:      105.9
Residual Deviance: 5.359e-08    AIC: 12
> summary(fit)

Call:
glm(formula = mydata$CarType ~ ., family = "binomial", data = mydata)

Deviance Residuals:
       Min          1Q      Median          3Q         Max
-1.316e-04  -2.100e-08  -2.100e-08   2.100e-08   1.266e-04

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)    -22.76   12007.54  -0.002    0.998
ManHI          -13.48    3077.29  -0.004    0.997
ManBI           36.02    7221.18   0.005    0.996
IntI           -44.90    8853.08  -0.005    0.996
HVACi          -58.50   11461.92  -0.005    0.996
Safety         -27.36    5396.42  -0.005    0.996
> summary(mydata)
     ManHI              ManBI               IntI              HVACi
 Min.   :-9.9300   Min.   :-9.9400   Min.   :-9.9900   Min.   :-9.8200
 1st Qu.:-5.1950   1st Qu.:-5.7050   1st Qu.:-5.5725   1st Qu.:-5.6750
 Median : 0.6350   Median :-1.8150   Median :-0.4150   Median : 0.8700
 Mean   :-0.0935   Mean   :-0.9277   Mean   :-0.1349   Mean   : 0.1197
 3rd Qu.: 5.0500   3rd Qu.: 3.4175   3rd Qu.: 4.9775   3rd Qu.: 5.0625
 Max.   : 9.5700   Max.   : 9.6100   Max.   : 9.7200   Max.   : 9.8900
     Safety            CarType
 Min.   :-9.8000   Length:80
 1st Qu.:-4.6775   Class :character
 Median : 0.8300   Mode  :character
 Mean   : 0.5437
 3rd Qu.: 4.6225
 Max.   : 9.9900
> mydata[6] <- as.factor(mydata$CarType)
> mydata
   ManHI ManBI  IntI HVACi Safety   CarType
1  -5.27 -1.30  2.86 -4.85   4.04       SUV
2  -4.82 -5.38  9.72 -0.97  -4.57 Hatchback
3   9.57 -7.50 -7.61  1.33  -5.10 Hatchback
4   2.84 -2.85  0.92  5.51  -6.64 Hatchback
5   0.00  2.68 -4.15  0.85   5.58       SUV
6   0.40  6.34  0.83  5.03  -8.10       SUV
7   5.94  3.14 -6.65  6.62  -1.32 Hatchback
8   5.78 -1.75 -6.85  0.73   5.50 Hatchback
9   0.86 -4.32  8.10 -8.96   3.10 Hatchback
10  7.36  7.42  0.27 -8.62   3.08       SUV
11 -7.95 -6.44 -4.68  3.86   9.82 Hatchback
12 -1.75 -8.09 -6.06 -9.13  -6.45       SUV
```

```
(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1.0585e+02  on 79  degrees of freedom
Residual deviance: 5.3590e-08  on 74  degrees of freedom
AIC: 12

Number of Fisher Scoring iterations: 25

> train <- predict(fit, type='response')
> plot(train)
> tapply(train, mydata$CarType, mean)
   Hatchback          SUV
2.851316e-10 1.000000e+00
> pred <- predict(fit,newdata = mytestdata, type='response')
> plot(pred)
> mytestdata[pred<=0.5,'Predict'] <- 'Hatchback'
> mytestdata[pred>0.5,'Predict'] <- 'SUV'
```






```
> mytestdata
   ManHI ManBI  IntI HVACi Safety   CarType   Predict
1   1.94  2.21  3.38  1.78  -7.19 Hatchback Hatchback
2  -0.02 -3.33  0.79 -6.63   7.99       SUV Hatchback
3  -0.49 -4.48  5.00  8.33  -2.77 Hatchback Hatchback
4   5.76  1.35  7.92 -0.43   4.29 Hatchback Hatchback
5   2.51 -8.74  4.53 -1.91   3.95 Hatchback Hatchback
6  -4.47  8.42 -0.05  5.57   9.62 Hatchback Hatchback
7  -9.89 -2.25 -5.00 -9.23   9.38       SUV       SUV
8  -9.94 -3.23  2.81 -2.98  -1.12       SUV       SUV
9  -8.37  4.21 -8.95  6.66   7.34       SUV       SUV
10  8.48  0.38 -3.02 -1.92  -7.43       SUV       SUV
11  0.79  0.96 -4.03 -2.28   6.20       SUV       SUV
12  5.32  2.08  5.55  7.89  -6.80 Hatchback Hatchback
13 -7.26 -0.11 -5.27 -7.14   1.20       SUV       SUV
14  0.69  3.37  3.70 -5.73  -5.86       SUV       SUV
15 -5.53 -0.12  1.61  2.31  -8.66       SUV       SUV
16  8.29  1.44 -7.26  5.06  -7.00       SUV       SUV
17  9.09 -2.26  1.64  2.80  -1.22 Hatchback Hatchback
18  5.04  4.52  0.28  8.26   4.59 Hatchback Hatchback
19  4.55 -3.88 -2.02 -1.20  -0.42 Hatchback Hatchback
20 -5.55  6.02  8.87  5.26  -2.27 Hatchback Hatchback
> #install.packages("caret") run on console
> library(caret)
```

```
> confusionMatrix(table(mytestdata[,7],mytestdata[,6]),positive='Hatchback')
Confusion Matrix and Statistics


           Hatchback SUV
  Hatchback        10   1
  SUV               0   9

               Accuracy : 0.95
                 95% CI : (0.7513, 0.9987)
    No Information Rate : 0.5
    P-Value [Acc > NIR] : 2.003e-05

                  Kappa : 0.9

 Mcnemar's Test P-Value : 1

            Sensitivity : 1.0000
            Specificity : 0.9000
         Pos Pred Value : 0.9091
         Neg Pred Value : 1.0000
             Prevalence : 0.5000
         Detection Rate : 0.5000
   Detection Prevalence : 0.5500
      Balanced Accuracy : 0.9500

       'Positive' Class : Hatchback
```

Experiment – 7

Classification Model

Packets for classification:

1. Caret package –

Caret stands for classification and regression training and is arguably the biggest project in R.

One of the most powerful and popular packages is the caret library, which follows a consistent syntax for data preparation, model building, and model evaluation, making it easy for data science practitioners.

2. Class package –

A **class** is just a blueprint or a sketch of methods or attributes. It represents the set of properties or methods that are common to all objects of one type.

Dataset – Servicetraindata.csv

Description – This data set contains 315 observations of 6 variables.

Attributes –

1. OilQual
2. EnginePerf
3. NormMileage
4. TypeWear
5. HVACwear
6. Service

Dataset – Servicetestdata.csv
Description – This dataset contains 135 observations of 6 variables.
Attributes –

1. OilQual
2. EnginePerf
3. NormMileage
4. TypeWear
5. HVACwear
6. Service

Predictknn –

Predictions are calculated for each test case by aggregating the responses of the k-nearest neighbours among the training cases. k may be specified to be any positive integer less than the number of training cases, but is generally between 1 and 10.

7. **Aim:** To find the performance of Classification Model

a. To install relevant packages for classification.

b. To choose a classifier for classification problems.

c. To evaluate the performance of the classifier.

Datasets used are servicetraindata.csv and servicetestdata.csv

Program:

```
# install.packages("caret") run command on console
# install.packages("class") run command on console

mytraindata <- read.csv('servicetraindata.csv')
mytestdata <- read.csv('servicetestdata.csv')
mytraindata
mytestdata
str(mytraindata)
str(mytestdata)
summary(mytraindata)
summary(mytestdata)
mytraindata[6] <- as.factor(mytraindata$Service)
summary(mytraindata)
mytestdata[6] <- as.factor(mytestdata$Service)
summary(mytestdata)
library(class)
predictknn <- knn(train=mytraindata[,-6],
test=mytestdata[,-6],
cl=mytraindata$Service,
k = 3)
predictknn
library(caret)
confusionMatrix(data=predictknn,mytestdata$Service)
```

Output:

```
> #install.packages("caret") run command on console
> #install.packages("class") run command on console
> mytraindata = read.csv('servicetraindata.csv')
> mytestdata = read.csv('servicetestdata.csv')
> mytraindata
      OilQual EnginePerf NormMileage  TyreWear    HVACwear Service
1  103.388821 103.504032  103.051485 106.172658 105.6868429      No
2   26.765516  26.188265   31.259536  29.186162  31.3112751     Yes
3   62.413282  63.681061   59.720883  64.704031  58.6205175     Yes
4   45.533338  49.906615   48.777581  48.113851  47.9520717      No
5  104.388821 103.284032  103.051485 105.822658 106.5368429      No
6    4.987185   4.941003    3.588986   8.363161   1.2802622      No
7  104.468821 103.744032  102.291485 106.172658 107.5368429      No
8  104.388821 103.134032  103.301485 106.172658 105.5368429      No
9    4.987185   5.851003    9.218986   8.373161   2.2802622      No
10  45.603338  49.666615   49.777581  51.263851  48.8720717      No
11  45.533338  46.666615   48.627581  47.273851  47.9520717      No
12   5.517185   6.651003    3.588986   8.373161   1.2802622      No
13  26.765516  29.418265   31.339536  29.036162  33.2812751     Yes
14 104.388821 103.744032  103.051485 108.092658 106.1568429      No
15  59.283282  64.641061   59.720883  63.304031  62.6205175     Yes
16  49.533338  49.666615   49.687581  48.263851  48.0020717      No
17  45.703338  46.786615   49.777581  48.263851  47.9520717      No
18   4.987185   2.891003    6.588986  11.783161   1.5502622      No
19  46.473338  49.666615   49.777581  45.913851  47.9520717      No
20  28.765516  27.418265   31.259536  28.596162  31.1112751     Yes
21  46.073338  48.346615   49.777581  48.263851  45.9520717      No
22 103.418821 102.634032  103.051485 105.172658 105.5368429      No
23 105.258821 103.744032  101.541485 105.172658 105.5368429      No
24  45.533338  48.806615   46.777581  48.233851  47.9520717      No
```

```
25   46.303338  49.666615   48.227581  45.263851  47.9520717        No
26    4.987185   3.021003    5.588986   8.373161   0.4602622        No
27    4.987185   4.891003    4.358986   8.373161   1.6602622        No
28   28.795516  27.418265   31.699536  29.036162  31.3112751       Yes
29  104.698821 103.744032  103.051485 106.222658 105.5368429        No
30   45.533338  49.666615   46.777581  47.513851  47.6620717        No
31  103.938821 103.744032  104.161485 106.172658 105.5368429        No
32    5.987185   4.891003    8.578986  10.543161   1.2802622        No
33  103.668821 103.744032  103.051485 106.172658 105.8168429        No
34  106.428821 103.744032  103.311485 106.172658 105.5368429        No
35   64.413282  62.741061   59.720883  62.474031  63.7205175       Yes
36   62.413282  62.741061   64.340883  64.324031  63.6205175       Yes
37   45.533338  48.916615   49.777581  48.263851  48.9720717        No
38   26.765516  28.608265   31.259536  30.036162  31.3612751       Yes
39  104.388821 103.294032  104.051485 106.272658 105.5368429        No
40   62.413282  62.701061   59.720883  63.544031  63.6205175       Yes
41  104.388821 103.634032  103.051485 105.902658 104.5368429        No
42    3.127185   1.891003    5.658986   8.373161   1.2802622        No
43  104.388821 104.514032  104.821485 106.172658 105.5368429        No
44  106.108821 103.744032  102.851485 106.172658 105.5368429        No
45   60.073282  62.741061   61.590883  63.304031  65.6205175       Yes
46    3.987185   6.981003    6.588986  10.353161   1.2802622        No
47   44.043338  47.666615   49.777581  47.863851  47.9520717        No
48  105.028821 103.744032  102.661485 106.172658 105.5368429        No
49   45.533338  49.646615   49.777581  48.273851  48.9520717        No
50  104.278821 102.744032  103.051485 106.172658 105.7868429        No
51   65.413282  63.701061   59.720883  63.304031  60.9705175       Yes
52  104.388821 104.744032  101.861485 106.172658 107.0568429        No
53   45.003338  49.096615   49.777581  48.263851  47.9520717        No
54   61.473282  62.741061   60.910883  63.304031  63.6205175       Yes
```

```
> mytestdata
      OilQual EnginePerf NormMileage   TyreWear     HVACwear Service
1   45.773338  49.936615   49.777581  48.263851  50.95207173      No
2    4.987185   7.891003    6.588986   9.493161   3.24026216      No
3    4.987185   4.891003    7.308986   8.373161   2.78026216      No
4  106.388821 104.454032  103.051485 106.282658 105.53684290      No
5  104.388821 103.744032  103.051485 106.132658 105.77684290      No
6    4.987185   4.891003    5.618986   8.373161   1.76026216      No
7   45.533338  50.666615   48.167581  50.633851  47.95207173      No
8   27.765516  29.138265   31.259536  31.226162  31.31127506     Yes
9   26.765516  28.418265   30.809536  29.266162  31.31127506     Yes
10 104.388821 103.744032  105.051485 106.212658 104.24684290      No
11   4.987185   5.891003    7.228986   8.373161   1.08026216      No
12 104.388821 103.434032  104.051485 106.062658 105.53684290      No
13   4.987185   4.391003    6.588986   9.253161  -1.71973784      No
14 104.338821 103.744032  103.591485 106.172658 105.53684290      No
15   7.987185   4.661003    6.588986   8.373161   2.25026216      No
16   4.987185   5.171003    6.588986  10.373161   0.08026216      No
17  63.413282  62.451061   60.520883  63.304031  62.62051752     Yes
18   6.897185   4.891003    6.588986   7.373161   3.55026216      No
19 103.648821 103.744032  102.051485 106.662658 105.53684290      No
20  58.963282  62.741061   61.570883  63.304031  62.62051752     Yes
21  26.745516  27.418265   31.259536  31.036162  31.26127506     Yes
22 104.388821 103.584032  103.371485 106.172658 104.53684290      No
23  63.083282  64.741061   59.720883  64.914031  62.62051752     Yes
24   4.877185   4.461003    6.588986   8.373161   2.28026216      No
25   4.987185   2.891003    4.808986   6.143161   1.28026216      No
26 104.328821 103.744032  103.051485 106.172658 105.53684290      No
27   4.987185   2.891003    6.588986  10.833161   0.10026216      No
28   2.597185   3.541003    6.588986   8.373161   1.28026216      No
```

```
> summary(mytestdata)
    OilQual          EnginePerf        NormMileage         TyreWear          HVACwear      Service
 Min.   :  2.597   Min.   :  1.891   Min.   :  3.589   Min.   :  6.143   Min.   : -1.72   No :99
 1st Qu.: 26.696   1st Qu.: 27.418   1st Qu.: 31.260   1st Qu.: 28.901   1st Qu.: 31.31   Yes:36
 Median : 61.023   Median : 61.501   Median : 59.351   Median : 61.304   Median : 62.62
 Mean   : 58.629   Mean   : 59.077   Mean   : 59.118   Mean   : 60.864   Mean   : 58.99
 3rd Qu.:104.229   3rd Qu.:103.744   3rd Qu.:103.051   3rd Qu.:106.173   3rd Qu.:105.33
 Max.   :106.389   Max.   :105.744   Max.   :105.051   Max.   :108.173   Max.   :105.83
> mytraindata[6] = as.factor(mytraindata$Service)
> summary(mytraindata)
    OilQual           EnginePerf        NormMileage         TyreWear          HVACwear
 Min.   :  0.9872   Min.   :  1.891   Min.   :  3.359   Min.   :  6.213   Min.   : -1.72
 1st Qu.: 26.7655   1st Qu.: 27.418   1st Qu.: 31.260   1st Qu.: 29.036   1st Qu.: 31.34
 Median : 59.6633   Median : 59.741   Median : 57.221   Median : 60.304   Median : 60.62
 Mean   : 59.6493   Mean   : 60.306   Mean   : 60.297   Mean   : 61.759   Mean   : 60.39
 3rd Qu.:104.3888   3rd Qu.:103.744   3rd Qu.:103.051   3rd Qu.:106.173   3rd Qu.:105.54
 Max.   :106.4288   Max.   :105.744   Max.   :105.051   Max.   :108.173   Max.   :107.54
 Service
 No :232
 Yes: 83
```

```
> mytestdata[6] = as.factor(mytestdata$Service)
> summary(mytestdata)
    OilQual          EnginePerf        NormMileage         TyreWear          HVACwear       Service
 Min.   :  2.597   Min.   :  1.891   Min.   :  3.589   Min.   :  6.143   Min.   : -1.72   No :99
 1st Qu.: 26.696   1st Qu.: 27.418   1st Qu.: 31.260   1st Qu.: 28.901   1st Qu.: 31.31   Yes:36
 Median : 61.023   Median : 61.501   Median : 59.351   Median : 61.304   Median : 62.62
 Mean   : 58.629   Mean   : 59.077   Mean   : 59.118   Mean   : 60.864   Mean   : 58.99
 3rd Qu.:104.229   3rd Qu.:103.744   3rd Qu.:103.051   3rd Qu.:106.173   3rd Qu.:105.33
 Max.   :106.389   Max.   :105.744   Max.   :105.051   Max.   :108.173   Max.   :105.83
> library(class)
> predictknn = knn(train=mytraindata[,-6],
+ test=mytestdata[,-6],
+ cl = mytraindata$Service, k=3)
> predictknn
  [1] No  No  No  No  No  No  No  Yes Yes No  No  No  No  No  No  No  Yes No  No  Yes Yes No  Yes
 [24] No  No  No  No  No  No  No  No  No  No  No  No  Yes No  No  No  No  No  Yes No  Yes No  No
 [47] No  Yes Yes No  Yes No  Yes No  No  No  No  No  No  No  No  No  No  No  Yes Yes Yes No  Yes
 [70] No  No  Yes No  No  No  No  No  No  Yes Yes Yes Yes No  Yes No  No  Yes Yes Yes No  No  No
 [93] Yes No  Yes No  No  No  No  No  No  No  No  No  No  Yes No  No  No  No  No  Yes No  Yes No
[116] Yes Yes No  Yes No  No  No  No  No  Yes No  No  No  No  No  No  Yes No  No
Levels: No Yes
> library(caret)
> str(mytraindata)
'data.frame':	315 obs. of  6 variables:
 $ OilQual    : num  103.4 26.8 62.4 45.5 104.4 ...
 $ EnginePerf : num  103.5 26.2 63.7 49.9 103.3 ...
 $ NormMileage: num  103.1 31.3 59.7 48.8 103.1 ...
 $ TyreWear   : num  106.2 29.2 64.7 48.1 105.8 ...
 $ HVACwear   : num  105.7 31.3 58.6 48 106.5 ...
 $ Service    : Factor w/ 2 levels "No","Yes": 1 2 2 1 1 1 1 1 1 1 ...
> str(mytestdata)
'data.frame':	135 obs. of  6 variables:
 $ OilQual    : num  45.77 4.99 4.99 106.39 104.39 ...
 $ EnginePerf : num  49.94 7.89 4.89 104.45 103.74 ...
 $ NormMileage: num  49.78 6.59 7.31 103.05 103.05 ...
 $ TyreWear   : num  48.26 9.49 8.37 106.28 106.13 ...
 $ HVACwear   : num  50.95 3.24 2.78 105.54 105.78 ...
 $ Service    : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 2 2 1 ...
> summary(mytraindata)
    OilQual           EnginePerf        NormMileage         TyreWear          HVACwear
 Min.   :  0.9872   Min.   :  1.891   Min.   :  3.359   Min.   :  6.213   Min.   : -1.72
 1st Qu.: 26.7655   1st Qu.: 27.418   1st Qu.: 31.260   1st Qu.: 29.036   1st Qu.: 31.34
 Median : 59.6633   Median : 59.741   Median : 57.221   Median : 60.304   Median : 60.62
 Mean   : 59.6493   Mean   : 60.306   Mean   : 60.297   Mean   : 61.759   Mean   : 60.39
 3rd Qu.:104.3888   3rd Qu.:103.744   3rd Qu.:103.051   3rd Qu.:106.173   3rd Qu.:105.54
 Max.   :106.4288   Max.   :105.744   Max.   :105.051   Max.   :108.173   Max.   :107.54
 Service
 No :232
 Yes: 83

> confusionMatrix(data=predictknn, mytestdata$Service)
Confusion Matrix and Statistics

          Reference
Prediction No Yes
       No  99   0
       Yes  0  36

               Accuracy : 1
                 95% CI : (0.973, 1)
    No Information Rate : 0.7333
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 1

 Mcnemar's Test P-Value : NA

            Sensitivity : 1.0000
            Specificity : 1.0000
         Pos Pred Value : 1.0000
         Neg Pred Value : 1.0000
             Prevalence : 0.7333
         Detection Rate : 0.7333
   Detection Prevalence : 0.7333
      Balanced Accuracy : 1.0000

       'Positive' Class : No
```

Experiment – 8

Clustering Model

8a -

K-Means Clustering in R Programming language K-Means is an iterative hard clustering technique that uses an unsupervised learning algorithm. In this, total numbers of clusters are pre-defined by the user and based on the similarity of each data point, the data points are clustered. This algorithm also finds out the centroid of the cluster.

Algorithm -

- Specify number of clusters (K)
- Randomly assign each data point to a cluster
- Calculate cluster centroids
- Re-allocate each data point to their nearest cluster centroid.
- Re-figure cluster centroid.

8b -

1. We will use the built in read.csv(...) function call, which reads the data in as a data frame, and assign the data frame to a variable (using <-) so that it is stored in R's memory. Then we will explore some of the basic arguments that can be supplied to the function.

2. The default for read.csv(...) is to set the header argument to TRUE. This means that the first row of values in the .csv is set as header information (column names). If your data set does not have a header, set the header argument to FALSE

3. To see the internal structure, we can use another function, str(). In this case, the data frame's internal structure includes the format of each column.

Library – factoextra

" factoextra " is an R package making easy to extract and visualize the output of exploratory multivariate data analyses.

- It produces a ggplot2-based elegant data visualization with less typing.
- It contains also many functions facilitating clustering analysis and visualization.

8 . Aim: To evaluate the performance of Clustering Model

a. Clustering algorithms for unsupervised classification.

b. Plot the cluster data using R visualizations.

Datasets used : tripdetails.csv

Program:

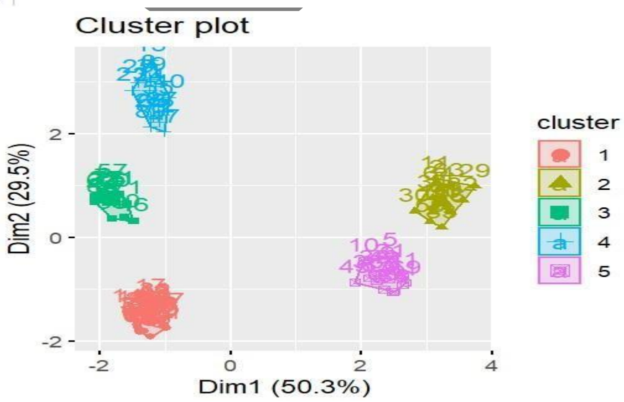
```
mydata<-read.csv('tripdetails.csv')
mydata
str(mydata)
summary(mydata)
myclusters<-kmeans(mydata[-1],5)
myclusters
library(factoextra)
fviz_cluster(myclusters,da=mydata,goem="point")
```

Output:

```
> mydata = read.csv("tripdetails.csv")
> mydata
   TripID TripLength MaxSpeed MostFreqSpeed TripDuration Brakes IdlingTime Honking
1       1         21       51            14           93    307         27     112
2       2        148      130           106          156    226          5     114
3       3         18       38            16          100    351         26     107
4       4         22       43            48           36     17          4       5
5       5        183      108            90          171     88          5      29
6       6         18       43            13           64    136         25      21
7       7         20       37            15           85    121         26      23
8       8         21       38            14           69    114         25      20
9       9        181       99           108          155     86          5      25
10     10        174      100            92          133    106          5      34
11     11        177      130            85          152    210          5     128
12     12         17       67            41           30     33          4      17
13     13         19       42            14          102    429         27      97
14     14         18       39            39           37     20          4       5
15     15         17       39            16           87    115         25      26
16     16        193      122           101          150    183          6      94
17     17         17       61            43           26     40          5      23
18     18         20       35            43           42     15          4       5
19     19         21       48            15           88    384         27      98
20     20         21       39            15           92    131         24      23
21     21        181      111           100          147     88          6      33
22     22         20       35            15           88    128         26      22
23     23         22       35            14           79    343         28     100
24     24         20       43            16           91    361         26      96
25     25         20       41            14           87    405         27      84
26     26         21       37            16           94    120         27      20
27     27         21       44            45           42     17          5       5
```

28	28	19	48	16	86	311	24	90
29	29	176	126	118	162	174	6	150
30	30	151	120	105	128	195	5	108
31	31	23	59	43	29	37	5	24
32	32	19	40	13	76	144	26	23
33	33	23	64	39	29	29	5	21
34	34	171	122	106	118	205	5	138
35	35	18	48	36	35	19	4	4
36	36	163	104	95	138	89	5	26
37	37	193	105	90	133	76	5	29
38	38	151	124	81	143	87	5	30
39	39	19	39	46	37	14	5	5
40	40	20	49	13	103	361	23	101
41	41	195	110	98	157	83	5	29
42	42	22	59	42	30	35	5	18
43	43	20	54	42	34	37	5	22
44	44	20	35	13	83	123	27	23
45	45	179	103	87	109	100	5	30
46	46	20	39	17	98	128	18	22
47	47	19	53	14	90	265	23	100
48	48	18	62	34	27	34	5	16
49	49	179	109	85	158	82	6	27
50	50	20	46	40	31	17	5	5
51	51	19	44	15	95	131	28	25
52	52	23	58	40	33	34	5	20
53	53	20	38	13	87	99	24	20
54	54	22	45	48	28	20	4	5
55	55	20	54	14	77	373	25	98
56	56	170	131	87	119	85	5	30
57	57	23	43	15	94	139	32	19

```
> str(mydata)
'data.frame':   91 obs. of  8 variables:
 $ TripID       : int  1 2 3 4 5 6 7 8 9 10 ...
 $ TripLength   : int  21 148 18 22 183 18 20 21 181 174 ...
 $ MaxSpeed     : int  51 130 38 43 108 43 37 38 99 100 ...
 $ MostFreqSpeed: int  14 106 16 48 90 13 15 14 108 92 ...
 $ TripDuration : int  93 156 100 36 171 64 85 69 155 133 ...
 $ Brakes       : int  307 226 351 17 88 136 121 114 86 106 ...
 $ IdlingTime   : int  27 5 26 4 5 25 26 25 5 5 ...
 $ Honking      : int  112 114 107 5 29 21 23 20 25 34 ...
> summary(mydata)
     TripID       TripLength        MaxSpeed       MostFreqSpeed     TripDuration
 Min.   : 1.0   Min.   : 16.00   Min.   : 35.00   Min.   : 12.00   Min.   : 22.00
 1st Qu.:23.5   1st Qu.: 20.00   1st Qu.: 42.00   1st Qu.: 15.50   1st Qu.: 34.50
 Median :46.0   Median : 21.00   Median : 54.00   Median : 42.00   Median : 88.00
 Mean   :46.0   Mean   : 70.77   Mean   : 70.36   Mean   : 50.65   Mean   : 87.37
 3rd Qu.:68.5   3rd Qu.:163.00   3rd Qu.:105.50   3rd Qu.: 89.00   3rd Qu.:133.00
 Max.   :91.0   Max.   :210.00   Max.   :138.00   Max.   :118.00   Max.   :171.00
     Brakes         IdlingTime        Honking
 Min.   : 14.0   Min.   : 4.00   Min.   :  4.00
 1st Qu.: 36.5   1st Qu.: 5.00   1st Qu.: 20.00
 Median :100.0   Median : 5.00   Median : 25.00
 Mean   :135.4   Mean   :11.59   Mean   : 49.92
 3rd Qu.:198.0   3rd Qu.:24.00   3rd Qu.: 97.50
 Max.   :429.0   Max.   :32.00   Max.   :155.00
> myclusters=kmeans(mydata[-1],5)
> myclusters
K-means clustering with 5 clusters of sizes 31, 15, 15, 15, 15

Cluster means:
  TripLength  MaxSpeed MostFreqSpeed TripDuration    Brakes IdlingTime   Honking
1   19.83871  52.80645      41.67742     32.12903  27.16129   4.709677  12.51613
2  172.60000 122.60000     100.13333    142.73333 199.33333   5.000000 127.86667
3   20.06667  38.73333      14.53333     87.33333 127.66667  25.266667  22.00000
4   20.26667  45.06667      14.46667     88.73333 350.13333  25.400000  97.73333
5  175.40000 111.33333      92.00000    144.86667  88.40000   4.933333  29.40000

Clustering vector:
 [1] 4 2 4 1 5 3 3 3 5 5 2 1 4 1 3 2 1 1 4 3 5 3 4 4 4 3 1 4 2 2 1 3 1 2 1 5 5 5 1 4 5 1 1 3 5 3 4
[48] 1 5 1 3 1 3 1 4 5 3 1 2 1 1 1 2 2 4 1 1 1 5 3 3 4 1 1 1 2 2 5 1 1 5 2 2 5 1 1 4 2 4 2 1

Within cluster sum of squares by cluster:
[1]  8837.613 18050.933  4353.200 25986.800 10040.800
 (between_SS / total_SS =  97.1 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss" "betweenss"
[7] "size"         "iter"         "ifault"
> library(factoextra)
> fviz_cluster(myclusters, da=mydata, goem="point")
>
```



Experiment-9

Reading Xml File

Aim: To read an XML file.

XML file:

```xml
<RECORDS>
  <EMPLOYEE>
    <ID>1</ID>
    <NAME>Rick</NAME>
    <SALARY>623.3</SALARY>
    <STARTDATE>1/1/2012</STARTDATE>
    <DEPT>IT</DEPT>
  </EMPLOYEE>

  <EMPLOYEE>
    <ID>2</ID>
    <NAME>Dan</NAME>
    <SALARY>515.2</SALARY>
    <STARTDATE>9/23/2013</STARTDATE>
    <DEPT>Operations</DEPT>
  </EMPLOYEE>

  <EMPLOYEE>
    <ID>3</ID>
    <NAME>Michelle</NAME>
    <SALARY>611</SALARY>
    <STARTDATE>11/15/2014</STARTDATE>
    <DEPT>IT</DEPT>
  </EMPLOYEE>

  <EMPLOYEE>
    <ID>4</ID>
    <NAME>Ryan</NAME>
    <SALARY>729</SALARY>
    <STARTDATE>5/11/2014</STARTDATE>
    <DEPT>HR</DEPT>
  </EMPLOYEE>

  <EMPLOYEE>
    <ID>5</ID>
    <NAME>Gary</NAME>
    <SALARY>843.25</SALARY>
    <STARTDATE>3/27/2015</STARTDATE>
    <DEPT>Finance</DEPT>
  </EMPLOYEE>

  <EMPLOYEE>
    <ID>6</ID>
    <NAME>Nina</NAME>
    <SALARY>578</SALARY>
    <STARTDATE>5/21/2013</STARTDATE>
    <DEPT>IT</DEPT>
  </EMPLOYEE>

  <EMPLOYEE>
    <ID>7</ID>
    <NAME>Simon</NAME>
    <SALARY>632.8</SALARY>
    <STARTDATE>7/30/2013</STARTDATE>
    <DEPT>Operations</DEPT>
  </EMPLOYEE>

  <EMPLOYEE>
    <ID>8</ID>
    <NAME>Guru</NAME>
    <SALARY>722.5</SALARY>
    <STARTDATE>6/17/2014</STARTDATE>
    <DEPT>Finance</DEPT>
  </EMPLOYEE>
```

</RECORDS>

Program:

```
# Load the package required to read XML files.
install.packages("XML")
library("XML")

# Also load the other required package.
library("methods")

# Give the input file name to the function.
result <- xmlParse(file = "D:/emp.xml")

# Print the result.
print(result)
```

Output:

```
> library("XML")
> # Also load the other required package.
> library("methods")
> # Give the input file name to the function.
> result=xmlParse(file = "D:/emp.xml.txt")
> # Print the result.
> print(result)
<?xml version="1.0"?>
<RECORDS>
  <EMPLOYEE>
    <ID>1</ID>
    <NAME>Rick</NAME>
    <SALARY>623.3</SALARY>
    <STARTDATE>1/1/2012</STARTDATE>
    <DEPT>IT</DEPT>
  </EMPLOYEE>
  <EMPLOYEE>
    <ID>2</ID>
    <NAME>Dan</NAME>
    <SALARY>515.2</SALARY>
    <STARTDATE>9/23/2013</STARTDATE>
    <DEPT>Operations</DEPT>
  </EMPLOYEE>
  <EMPLOYEE>
    <ID>3</ID>
    <NAME>Michelle</NAME>
    <SALARY>611</SALARY>
    <STARTDATE>11/15/2014</STARTDATE>
    <DEPT>IT</DEPT>
  </EMPLOYEE>
  <EMPLOYEE>
    <ID>4</ID>
    <NAME>Ryan</NAME>
    <SALARY>729</SALARY>
    <STARTDATE>5/11/2014</STARTDATE>
    <DEPT>HR</DEPT>
  </EMPLOYEE>
  <EMPLOYEE>
    <ID>5</ID>
    <NAME>Gary</NAME>
    <SALARY>843.25</SALARY>
    <STARTDATE>3/27/2015</STARTDATE>
    <DEPT>Finance</DEPT>
  </EMPLOYEE>
  <EMPLOYEE>
    <ID>6</ID>
    <NAME>Nina</NAME>
    <SALARY>578</SALARY>
    <STARTDATE>5/21/2013</STARTDATE>
    <DEPT>IT</DEPT>
  </EMPLOYEE>
  <EMPLOYEE>
    <ID>7</ID>
    <NAME>Simon</NAME>
    <SALARY>632.8</SALARY>
    <STARTDATE>7/30/2013</STARTDATE>
    <DEPT>Operations</DEPT>
  </EMPLOYEE>
```

```
  <EMPLOYEE>
    <ID>8</ID>
    <NAME>Guru</NAME>
    <SALARY>722.5</SALARY>
    <STARTDATE>6/17/2014</STARTDATE>
    <DEPT>Finance</DEPT>
  </EMPLOYEE>
</RECORDS>

> |
```