Practical No. 5

Aim: - Introduction to R programming and Data acquisition.
Objective: - To understand the concept of R Programming. and evaluate different data mining techniques like classification, prediction, clustering and association rule mining in R

R Commands:-

```
myString <- "Hello, World!"
print ( myString)
```
```
[1] "Hello, World!"
```

```
#setwd() - sets working directory.
setwd("D:/58")
getwd() ##getwd() - gets current working directory.
```
```
[1] "D:/58"
```

```
dir() ## dir() - lists the contents of current working directory.
```
```
[1] "data_apriori.csv"  "missing_col1.csv"  "missing_col1.xlsx" "na_data.csv"       "PRACT5.R"
>
```

```
ls() ##ls() - lists names of objects in R environment
```
```
 [1] "accuracy"            "clusterCut"          "clusters"            "df"
 [5] "df2"                 "fit"                 "index"               "iris"
 [9] "iris_norm"           "iris_target_category" "iris_test"          "iris_test_category"
[13] "iris_train"          "irisCluster"         "lmmodel1"            "model"
[17] "my_data"             "myString"            "nor"                 "pr"
[21] "preds"               "preds_new"           "ran"                 "tab"
[25] "test"                "TestData"            "train"               "TrainData"
[29] "xTest"               "xTrain"              "yTest"               "yTrain"
```

```
#Creating and assigning to a variable:
x<-1
class(x)
```
```
[1] "numeric"
```

```
#Printing a variable:
#auto-printing
print(x) #explicit printing
```
```
[1] 1
```

```
x<-'c'
is.character(x) #check if character
```
```
[1] TRUE
```

```
is.integer(x) #check if integer
```
```
[1] FALSE
```

```
y<-'2.14'
as.integer(y)
```
```
[1] 2
```

```
###Creating Vector: contains objects of same class.
y<-vector("logical",length=10)
length(x)
```
```
[1] 3
```

```
y<-c(4,5,6)
5*x
```
```
[1]  56.5 137.5 169.0
```

```
###Creating Matrix: Two-dimensional array having elements of same class.
m<-matrix(c(11,12,13,55,60,65,66,72,78), nrow=3,ncol=3)
m
```
```
     [,1] [,2] [,3]
[1,]   11   55   66
[2,]   12   60   72
[3,]   13   65   78
>
```

```
dim(m)
```
```
[1] 3 3
```

```
attributes(m)
```
```
$dim
[1] 3 3
```

```
matrix(c(11,12,13,55,60,65,66,72,78),nrow=3,ncol=3,byrow = TRUE)
m
```
```
     [,1] [,2] [,3]
[1,]   11   12   13
[2,]   55   60   65
[3,]   66   72   78
> m
     [,1] [,2] [,3]
[1,]   11   55   66
[2,]   12   60   72
[3,]   13   65   78
>
```

```
x<-c(1,2,3) y<-c(11,12,13)
cbind(x,y)
```
```
          x y
[1,] 11.3 4
[2,] 27.5 5
[3,] 33.8 6
>
```

```
rbind(x,y)
```
```
   [,1] [,2] [,3]
x 11.3 27.5 33.8
y  4.0  5.0  6.0
>
```

```
p<-3*m
p
```

```
     [,1] [,2] [,3]
[1,]   33  165  198
[2,]   36  180  216
[3,]   39  195  234
> 
```

n<-
matrix(c(4,5,6,14,15,16,24,25,26),nrow=3,ncol=3)
q<- m+n
q

```
     [,1] [,2] [,3]
[1,]   15   69   90
[2,]   17   75   97
[3,]   19   81  104
> 
```

o<-matrix(c(4,5,6,14,15,16),nrow=3,ncol=2)
o

```
     [,1] [,2]
[1,]    4   14
[2,]    5   15
[3,]    6   16
> 
```

r<-m%*% o
r

```
     [,1] [,2]
[1,]  715 2035
[2,]  780 2220
[3,]  845 2405
> 
```

mdash<-t(m)
mdash

```
     [,1] [,2] [,3]
[1,]   11   12   13
[2,]   55   60   65
[3,]   66   72   78
> 
```

s<-matrix(c(4,5,6,14,15,16,24,25,26),
nrow=3,ncol=3,byrow=TRUE)
s_det<-det(s)
s_det

```
[1] 1.110223e-14
```

###List: A special type of vector containing elements of different classes. ####Elements of list can be accessed by giving element index or name in [[]].

x<-list(1,"p",TRUE,2+4i)
x

```
[[1]]
[1] 1

[[2]]
[1] "p"

[[3]]
[1] TRUE

[[4]]
[1] 2+4i
```

###Factor: Represents categorical data. Can be ordered or unordered.
status<-
c("low","high","medium","high","low")
x<-factor(status,
ordered=TRUE,levels=c("low","medium","high"))
x

```
[1] low    high   medium high   low
Levels: low < medium < high
> 
```

###Data frame: Used to store tabular data. Can contain different classes.
student_id<-c(1,2,3)
student_names<-
c("Ram","Shyam","Laxman")
position<-
 c("First","Second","Third")
data<-
data.frame(student_id,student_names,position)
data

```
  student_id student_names position
1          1           Ram    First
2          2         Shyam   Second
3          3        Laxman    Third
> 
```

data$student_id

```
[1] 1 2 3
```

nrow(data)

```
[1] 3
```

ncol(data)

```
[1] 3
```

names(data)

```
[1] "student_id"    "student_names" "position"
```

smoke <-
matrix(c(51,43,22,92,28,21,68,22,9),ncol=3,byrow=TRUE)
colnames(smoke) <-
c("High","Low","Middle")
rownames(smoke) <-

c("current","former","never")
smoke <- as.table(smoke)
smoke

```
         High Low Middle
current    51  43      22
former     92  28      21
never      68  22       9
> |
```

Reading and writing data from csv
dataT <- read.table("na_data.csv", sep =",", header = T)
dataT

```
   X1       Rick  X623.3 X01.01.2012          IT
1   2        Dan  515.20  23/09/2013 Operations
2   3  Michelle   611.00  15/11/2014          IT
3   4       Ryan  729.00  11/05/2014          HR
4  NA       Gary  843.25  27/03/2015     Finance
5   6      Nina       NA  21/05/2013          IT
6   7    Simon    632.80  30/07/2013 Operations
7   8       Guru  722.50  17/06/2014    Finance
8   9       John       NA  21/05/2012
9  10       Rock  600.80  30/07/2013          HR
10 11      Brad  1032.80  30/07/2013 Operations
11 12       Ryan  729.00  11/05/2014          HR
> |
```

dim(dataT)

```
[1] 11  5
```

head(dataT, 2)

```
  X1       Rick X623.3 X01.01.2012          IT
1  2        Dan  515.2  23/09/2013 Operations
2  3  Michelle   611.0  15/11/2014          IT
> |
```

tail(dataT, 2)

```
   X1   Rick X623.3 X01.01.2012          IT
10 11 Brad   1032.8  30/07/2013 Operations
11 12  Ryan   729.0  11/05/2014          HR
> |
```

z <- data.frame(a = 5, b = 10, c = pi)
write.csv(z,file="data.csv")

	A	B	C	D
		a	b	c
1		5	10	3.141593

Reading and writing data from Excel using XLConnect

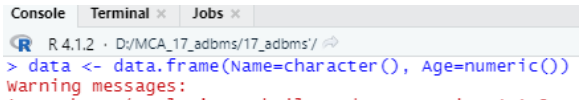
dataX <- XLConnect::readWorksheetFromFile("employee_info.xls",sheet=1)
dataX

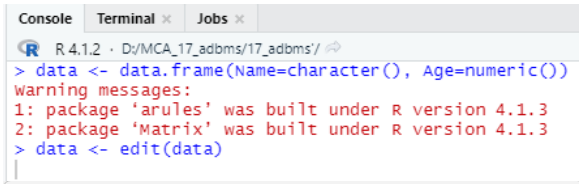
```
   X1       Rick  X623.3 X2012.01.01.00.00.00          IT
1   2        Dan  515.20            23/09/2013 Operations
2   3  Michelle   611.00            15/11/2014          IT
3   4       Ryan  729.00  2014-11-05 00:00:00          HR
4  NA       Gary  843.25            27/03/2015     Finance
5   6      Nina       NA            21/05/2013          IT
6   7    Simon    632.80            30/07/2013 Operations
7   8       Guru  722.50            17/06/2014    Finance
8   9       John       NA            21/05/2012       <NA>
9  10       Rock  600.80            30/07/2013          HR
10 11      Brad  1032.80            30/07/2013 Operations
11 12       Ryan  729.00  2014-11-05 00:00:00          HR
> |
```

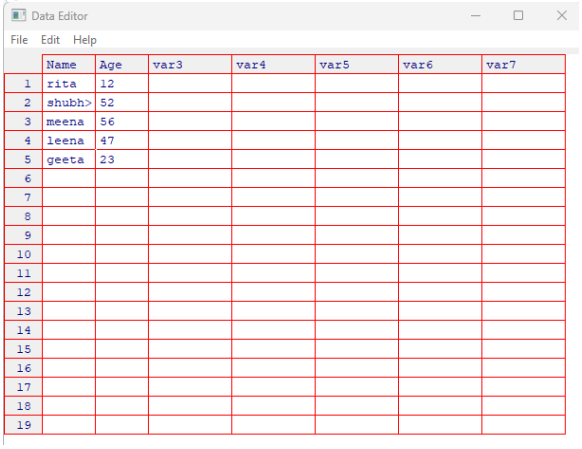
dataY<- dataX[1:2,]
dataY

```
  X1       Rick X623.3 X2012.01.01.00.00.00          IT
1  2        Dan  515.2            23/09/2013 Operations
2  3  Michelle   611.0            15/11/2014          IT
> |
```

Reading and writing data from Excel using readXL and writeXL
data <-data.frame(Name=character(), Age=numeric())







	Name	Age	var3	var4	var5	var6	var7
1	rita	12					
2	shubh>	52					
3	meena	56					
4	leena	47					
5	geeta	23					
6							
7							
8							
9							
10							
11							
12							
13							
14							
15							
16							
17							
18							
19							

Practical No. 6

Aim: - Implementation of Data preprocessing techniques like,

1. Naming and Renaming variables, adding a new variable.

2. Dealing with missing data.

3. Dealing with categorical data.

4. Data reduction using sub setting.

R Commands:-

#my_data

```
> setwd("D:/46_fymca")
> getwd()
[1] "D:/46_fymca"
> data<-mtcars
> head(data,5)
                   mpg cyl disp  hp drat    wt  qsec vs am gear carb
Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
> data1<-data[1:6,1:5]
> data1
                   mpg cyl disp  hp drat
Mazda RX4         21.0   6  160 110 3.90
Mazda RX4 Wag     21.0   6  160 110 3.90
Datsun 710        22.8   4  108  93 3.85
Hornet 4 Drive    21.4   6  258 110 3.08
Hornet Sportabout 18.7   8  360 175 3.15
Valiant           18.1   6  225 105 2.76
```

Renaming columns with dplyr::rename()

```
> require(dplyr)
> install.packages("dplyr")
Error in install.packages : Updating loaded packages
> install.packages("dplyr")
WARNING: Rtools is required to build R packages but is not currently installed. Please
 download and install the appropriate version of Rtools before proceeding:

https://cran.rstudio.com/bin/windows/Rtools/
Warning in install.packages :
  package 'dplyr' is in use and will not be installed
> data1=rename(data1,horse_power=hp)
> data1
                   mpg cyl disp horse_power drat
Mazda RX4         21.0   6  160         110 3.90
Mazda RX4 Wag     21.0   6  160         110 3.90
Datsun 710        22.8   4  108          93 3.85
Hornet 4 Drive    21.4   6  258         110 3.08
Hornet Sportabout 18.7   8  360         175 3.15
Valiant           18.1   6  225         105 2.76
```

Adding new variable

```
> data1$new_hp1 <- data1$horse_power*0.5
> colnames(data1)
[1] "mpg"        "cyl"        "disp"       "horse_power" "drat"       "new_hp1"
> data1
                   mpg cyl disp horse_power drat new_hp1
Mazda RX4         21.0   6  160         110 3.90    55.0
Mazda RX4 Wag     21.0   6  160         110 3.90    55.0
Datsun 710        22.8   4  108          93 3.85    46.5
Hornet 4 Drive    21.4   6  258         110 3.08    55.0
Hornet Sportabout 18.7   8  360         175 3.15    87.5
Valiant           18.1   6  225         105 2.76    52.5
```

#naming variable

Create missing_col1.csv file with following data.

1	Rick	623.3	01/01/2012	IT
2	Dan	515.2	23/09/2013	Operations
3	Michelle	611	15/11/2014	IT
4	Ryan	729	11/05/2014	HR
	Gary	843.25	27/03/2015	Finance
6	Nina		21/05/2013	IT
7	Simon	632.8	30/07/2013	Operations
8	Guru	722.5	17/06/2014	Finance
9	John		21/05/2012	
10	Rock	600.8	30/07/2013	HR
11	Brad	1032.8	30/07/2013	Operations
12	Ryan	729	11/05/2014	HR

#Reading with read.table() assumes no headers by default. First few lines :

```
> data2 = read.table(file= "D:/46_fymca/missing_col1.csv", sep = ",")
> data2=read.table(file="D:/46_fymca/na_data.csv",sep = ",")
> data2
   V1         V2      V3         V4          V5
1   1       Rick  623.30 01/01/2012          IT
2   2        Dan  515.20 23/09/2013  Operations
3   3  Michelle   611.00 15/11/2014          IT
4   4       Ryan  729.00 11/05/2014          HR
5  NA       Gary  843.25 27/03/2015     Finance
6   6      Nina       NA 21/05/2013          IT
7   7    Simon    632.80 30/07/2013  Operations
8   8       Guru  722.50 17/06/2014    Finance
9   9      John       NA 21/05/2012
10 10       Rock  600.80 30/07/2013          HR
11 11      Brad  1032.80 30/07/2013  Operations
12 12       Ryan  729.00 11/05/2014          HR
```

#V1, V2, V3.. are given as default names (titles) by R

```
> data2=read.csv(file="D:/46_fymca/na_data.csv",col.names=c("sn
o","Name","Salary","DateofJoining","Department"))
> data2
   sno       Name  Salary DateofJoining Department
1    2        Dan  515.20    23/09/2013 Operations
2    3  Michelle   611.00    15/11/2014         IT
3    4       Ryan  729.00    11/05/2014         HR
4   NA       Gary  843.25    27/03/2015    Finance
5    6      Nina       NA    21/05/2013         IT
6    7    Simon    632.80    30/07/2013 Operations
7    8       Guru  722.50    17/06/2014    Finance
8    9      John       NA    21/05/2012
9   10       Rock  600.80    30/07/2013         HR
10  11      Brad  1032.80    30/07/2013 Operations
11  12       Ryan  729.00    11/05/2014         HR
```

Error Detection and Correction NA: Not Available - Known as missing values

Works as a place holder for something that is 'missing'

Most basic operations (addition, subtraction, multiplication, etc.) in R deal with it without crashing and return NA if one of the inputs is NA

is.na(VALUE) is used to check if the input value is NA or not. Returns a TRUE/FALSE vector Whereas in case of Excel like utilities for numeric computations it's assumed to be 0.

Operation with NA

```
NA+4
[1]NA
```

# Create a vector V with 1 NA value
v <- c(1,2,NA,3)

Median with and without NA (remove NA)

```
> median(v)
[1] NA
```

On removing NAs

```
> median(v,na.rm = T)
[1] 2
```

Apply is.na() to vector
>is.n(V)

[1] FALSE FALSE TRUE FALSE

Removing the NA values by using logical indexing

```
> naVals <- is.na(v)
```

Get values that are not NA
V[!naVals]
[1] 1 2 3

Subsetting with complete cases - values that are not NA

V[complete.cases(V)]

[1] 1 2 3

#Create na_data.csv file with following data.

1	Rick	623.3	01/01/2012	IT
2	Dan	515.2	23/09/2013	Operations
3	Michelle	611	15/11/2014	IT
4	Ryan	729	11/05/2014	HR
	Gary	843.25	27/03/2015	Finance
6	Nina		21/05/2013	IT
7	Simon	632.8	30/07/2013	Operations
8	Guru	722.5	17/06/2014	Finance
9	John		21/05/2012	
10	Rock	600.8	30/07/2013	HR
11	Brad	1032.8	30/07/2013	Operations
12	Ryan	729	11/05/2014	HR

Subsetting a data frame with complete cases # Complete Data of Prime Ministers. Notice NAs

```
> dataC <- read.csv(file ="D:/46_fymca/na_data.csv", na.strings = "")
> dataC
   X1       Rick  X623.3 X01.01.2012         IT
1   2        Dan  515.20  23/09/2013 Operations
2   3  Michelle   611.00  15/11/2014         IT
3   4       Ryan  729.00  11/05/2014         HR
4  NA       Gary  843.25  27/03/2015    Finance
5   6      Nina       NA  21/05/2013         IT
6   7    Simon    632.80  30/07/2013 Operations
7   8       Guru  722.50  17/06/2014    Finance
8   9      John       NA  21/05/2012       <NA>
9  10       Rock  600.80  30/07/2013         HR
10 11      Brad  1032.80  30/07/2013 Operations
11 12       Ryan  729.00  11/05/2014         HR
```

Subset only the rows without NA

```
> dataCompleteCases <- dataC[complete.cases(dataC),]
> dataCompleteCases
   X1       Rick X623.3 X01.01.2012         IT
1   2        Dan  515.2  23/09/2013 Operations
2   3  Michelle   611.0  15/11/2014         IT
3   4       Ryan  729.0  11/05/2014         HR
6   7    Simon    632.8  30/07/2013 Operations
7   8       Guru  722.5  17/06/2014    Finance
9  10       Rock  600.8  30/07/2013         HR
10 11      Brad  1032.8  30/07/2013 Operations
11 12       Ryan  729.0  11/05/2014         HR
```

Imputation
The process of estimating or deriving missing values There are various methods for imputation
– Imputation of the mean
– Imputation of the median
– Imputation using linear regression models
• Package Hmisc implments many imputation methods, few examples :

```
> install.packages("Hmisc")
WARNING: Rtools is required to build R packages but is not currently installed. Please down
load and install the appropriate version of Rtools before proceeding:

https://cran.rstudio.com/bin/windows/Rtools/
Installing package into 'C:/Users/ADMIN/Documents/R/win-library/4.1'
(as 'lib' is unspecified)
trying URL 'https://cran.rstudio.com/bin/windows/contrib/4.1/Hmisc_4.7-2.zip'
Content type 'application/zip' length 3302837 bytes (3.1 MB)
downloaded 3.1 MB

package 'Hmisc' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
        C:\Users\ADMIN\AppData\Local\Temp\RtmpCiYL4J\downloaded_packages
> library(Hmisc)
Loading required package: lattice
Loading required package: survival
Loading required package: Formula
Loading required package: ggplot2

Attaching package: 'Hmisc'

The following objects are masked from 'package:base':

    format.pval, units

Warning messages:
1: package 'Hmisc' was built under R version 4.1.3
2: package 'ggplot2' was built under R version 4.1.3
```

create a vector

```
> x=c(1,2,3,NA,4,4,NA)
```

mean imputation - from package, mention name of function to be used

```
> x <- impute(x,fun = mean)
> x
   1    2    3    4    5    6    7
 1.0  2.0  3.0 2.8*  4.0  4.0 2.8*
```

#median imputation

```
> x <- impute(x,fun = median)
> x
   1    2    3    4    5    6    7
 1.0  2.0  3.0 2.8*  4.0  4.0 2.8*
```

** Categorical Data **

Factors are variables in R which take on a limited number of different values; such variables are often referred to as categorical variables.

#Convert Character into Factor(categorical data)

Create gender vector

```
> gender_vector <- c("Male", "Female", "Female", "Male", "Male")
> class(gender_vector)
[1] "character"
```

Convert gender_vector to a factor

```
> factor_gender_vector <- factor(gender_vector)
> class(factor_gender_vector)
[1] "factor"
```

Create Ordinal categorical vector

```
> day_vector <- c('evening', 'morning', 'afternoon', 'midday', 'midnight', 'evening')
```

Convert `day_vector` to a factor with ordered level

```
> factor_day <- factor(day_vector, order = TRUE, levels =c('morning', 'midday', 'afternoo
n','evening','midnight'))
```

Print the new variable

```
> factor_day
[1] evening   morning   afternoon midday    midnight  evening
Levels: morning < midday < afternoon < evening < midnight
```

Convert Numeric to Factor

```
> age <- c(40, 49, 48, 40, 67, 52, 53)
> salary <- c(103200, 106200, 150200, 10606, 10390, 14070, 10220)
> gender <- c("male","male", "transgender", "female", "male", "female", "transgender")
```

Creating vectors

Creating data frame named employee

```
> employee<-data.frame(age, salary, gender)
> employee
  age salary      gender
1  40 103200        male
2  49 106200        male
3  48 150200 transgender
4  40  10606      female
5  67  10390        male
6  52  14070      female
7  53  10220 transgender
```

Creating a factor corresponding to age with labels

```
> wfact = cut(employee$age, 3,labels=c('Young', 'Medium', 'Aged'))
> table(wfact)
wfact
 Young Medium   Aged
     4      2      1
>
```

Practical 7

Aim: - Implementation and analysis of Linear regression through graphical methods

Theory: - Linear regression analysis is used to predict the value of a variable based on the value of another variable. The variable you want to predict is called the dependent variable. The variable you are using to predict the other variable's value is called the independent variable.

R Commands:

setwd("D:/53 FYMCA/prac7")
#getwd()
[1] "D:/53 FYMCA/prac7"
my_data <-mtcars
names(my_data)

```
 [1] "mpg"  "cyl"  "disp" "hp"
```

dim(my_data)
---------- RANDOMIZE----------
my_data <- my_data[sample(nrow(my_data),),]
head(my_data)

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Honda Civic	30.4	4	75.7	52	4.93	1.615	18.52	1	1	4	2
Lotus Europa	30.4	4	95.1	113	3.77	1.513	16.90	1	1	5	2
Merc 280	19.2	6	167.6	123	3.92	3.440	18.30	1	0	4	4
Merc 230	22.8	4	140.8	95	3.92	3.150	22.90	1	0	4	2
Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3	1
AMC Javelin	15.2	8	304.0	150	3.15	3.435	17.30	0	0	3	2

TrainData <- my_data[1:20,]
TestData <- my_data[21:32,]
----------Linear Model----------
fit = lm(mpg ~ hp, data=mtcars)
summary(fit)

```
Call:
lm(formula = mpg ~ hp, data = mtcars)

Residuals:
    Min      1Q  Median      3Q     Max
-5.7121 -2.1122 -0.8854  1.5819  8.2360

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 30.09886    1.63392  18.421  < 2e-16 ***
hp          -0.06823    0.01012  -6.742 1.79e-07 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 3.863 on 30 degrees of freedom
Multiple R-squared:  0.6024,    Adjusted R-squared:  0.5892
F-statistic: 45.46 on 1 and 30 DF,  p-value: 1.788e-07
```

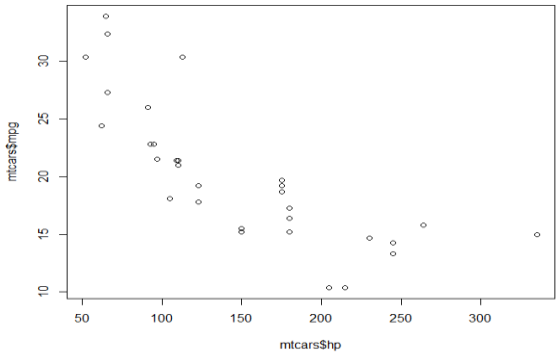
```
                    preds TestData.mpg
Fiat X1-9        25.59579         27.3
Lincoln Continental 15.42978         10.4
Hornet Sportabout   18.15891         18.7
Toyota Corona       23.48072         21.5
Chrysler Imperial   14.40636         14.7
Mazda RX4           22.59375         21.0
```

preds <- predict(fit, newdata = TestData)
df1 <-data.frame(preds,TestData$mpg)
head(df1)

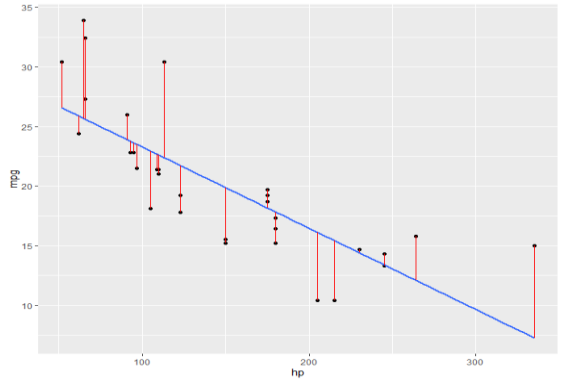
#Correlation :- cor(preds,TestData$mpg)

```
[1] 0.8784059
```

plot(mtcars$hp, mtcars$mpg)

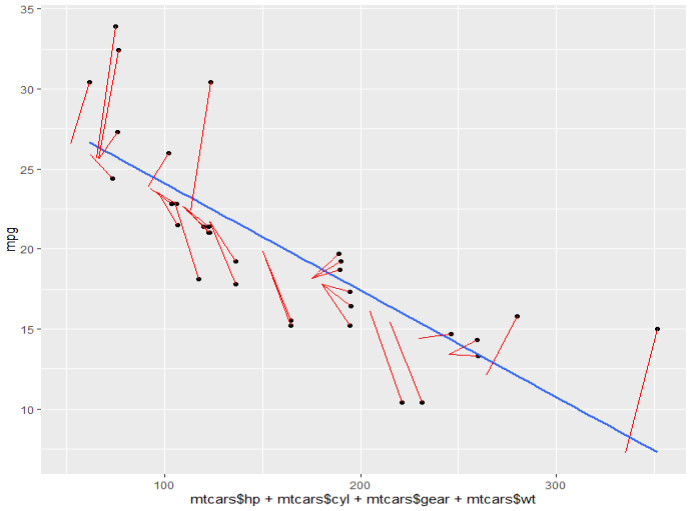


ggplot(fit, aes(hp, mpg)) + geom_point() +
stat_smooth(method = lm, se = FALSE) +
geom_segment(aes(xend = hp, yend = .fitted), color = "red", size = 0.3)



------Better MODEL----------
lmmodel1 <- lm(mpg ~ hp+cyl+gear+wt, data =TrainData)
summary(lmmodel1)

```
Call:
lm(formula = mpg ~ hp + cyl + gear + wt, data = TrainData)

Residuals:
    Min      1Q  Median      3Q     Max
-1.9431 -1.1795 -0.4437  0.4191  4.1244

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 50.719998   6.657175   7.619 1.56e-06
hp          -0.001659   0.015130  -0.110 0.914155
cyl         -1.684467   0.760197  -2.216 0.042588
gear        -1.310123   1.008465  -1.299 0.213515
wt          -4.667897   1.082578  -4.312 0.000617

(Intercept) ***
hp
cyl         *
gear
wt          ***
---
Signif. codes:
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.976 on 15 degrees of freedom
Multiple R-squared:  0.9138,    Adjusted R-squared:  0.8908
F-statistic: 39.73 on 4 and 15 DF,  p-value: 8.187e-08
```



preds_new <- predict(lmmodel1, newdata = TestData)
df2 <-data.frame(preds_new,TestData$mpg)
head(df2)

```
                  preds_new TestData.mpg
Fiat X1-9         29.599785          27.3
Lincoln Continental  7.638601          10.4
Hornet Sportabout 16.966057          18.7
Toyota Corona     28.384503          21.5
Chrysler Imperial  7.982484          14.7
Mazda RX4         22.960359          21.0
```

----------Corelation---------
cor(preds_new,TestData$mpg)

```
 [1] 0.8891933
```

ggplot(fit,
aes(mtcars$hp+mtcars$cyl+mtcars$gear+mtcars$wt, mpg)) + geom_point() +
stat_smooth(method = lm, se = FALSE) +
geom_segment(aes(xend = hp, yend = .fitted), color = "red", size = 0.3)

Practical 8

Aim: - Implementation and analysis of Classification algorithms like
1. Naive Bayesian,
2. K-Nearest Neighbor.

Theory:-

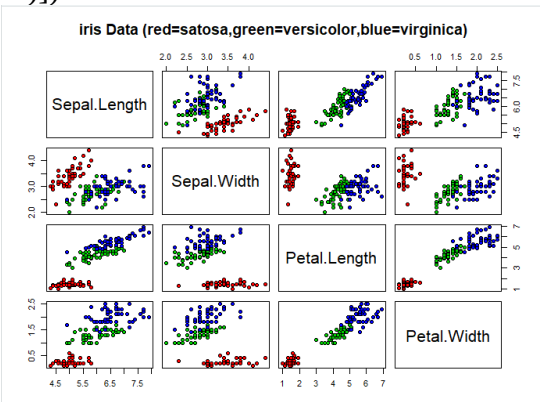
Naive Bayes
• Based on the Bayes theorem
• Predicts based on probabilities from training data
P(B|A) = P(A|B) P(B)/P(A) Gives posterior probability of 'B' given 'A' using prior probability of 'B' prior probability of 'A' and conditional probability of 'A' given 'B'
•Takes two step approach
– Calculates the posterior probability of the Class given the input - for every class
– Assigns the class with higher posterior probability
• More suited when dimensionality of input is high the - widely used for document classification
• Also good for the multiclass classifications
• Works well with less datasets also, but the assumption that predictor variables are independent should hold ##NaiveBayes .

R Commands:-

```
setwd("C:/Users/ADMIN/Desktop/ADBMS_R")
install.packages("caret")
library("caret")
install.packages("e1071")
library(e1071)
install.packages("klaR")
library("klaR")
library(ggplot2)

data(iris)
head(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
>
```

```
unique(iris$Species)
pairs(iris[1:4], main="Iris Data (red=setosa,green=versicolor,blue=virginica)", pch=21,
bg=c("red","green3","blue")[unclass(iris$Species)])
```




```
index = sample(nrow(iris), floor(nrow(iris) * 0.7))
train = iris[index,] test = iris[-index,]
xTrain = train[,-5]
yTrain = train$Species
xTest = test[,-5]
yTest =test$Species
model = train(xTrain,yTrain,'nb',trControl=trainControl(method='cv',number=10))
model
```

```
> model
Naive Bayes

105 samples
  4 predictor
  3 classes: 'setosa', 'versicolor', 'virginica'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 96, 95, 94, 93, 94, 94, ...
Resampling results across tuning parameters:

  usekernel  Accuracy   Kappa
  FALSE      0.9336364  0.9003204
   TRUE      0.9436364  0.9155895

Tuning parameter 'fL' was held constant at a value of 0
Tuning parameter 'adjust' was held
 constant at a value of 1
Accuracy was used to select the optimal model using the largest value.
The final values used for the model were fL = 0, usekernel = TRUE and adjust = 1.
>
```

```
prop.table(table(predict(model$finalModel,xTest)$class,yTest))
```

```
> prop.table(table(predict(model$finalModel,xTest)$class,yTest))
            yTest
                 setosa versicolor  virginica
  setosa     0.31111111 0.00000000 0.00000000
  versicolor 0.00000000 0.26666667 0.02222222
  virginica  0.00000000 0.00000000 0.40000000
>
```

K-Nearest Neighbor

```
df <- data(iris)
head(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
>
```

```
ran <-sample(1:nrow(iris), 0.9 * nrow(iris))
nor <-function(x) { (x - min(x))/(max(x)-
min(x)) }
iris_norm <-
as.data.frame(lapply(iris[,c(1,2,3,4)], nor))
summary(iris_norm)
```

```
> summary(iris_norm)
  Sepal.Length     Sepal.Width      Petal.Length     Petal.Width
 Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000
 1st Qu.:0.2222   1st Qu.:0.3333   1st Qu.:0.1017   1st Qu.:0.0833
 Median :0.4167   Median :0.4167   Median :0.5678   Median :0.5000
 Mean   :0.4287   Mean   :0.4406   Mean   :0.4675   Mean   :0.4580
 3rd Qu.:0.5833   3rd Qu.:0.5417   3rd Qu.:0.6949   3rd Qu.:0.7083
 Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000
>
```

```
iris_train <-iris_norm[ran,]
iris_test <-iris_norm[-ran,]
iris_target_category <- iris[ran,5]
iris_test_category <- iris[-ran,5]
library(class)
pr <-
knn(iris_train,iris_test,cl=iris_target_category,
k=13)
tab <-table(pr,iris_test_category)
accuracy <-
function(x){sum(diag(x)/(sum(rowSums(x))))
* 100}
accuracy(tab)
```

```
> accuracy(tab)
[1] 93.33333
>
```

Practical No: 9

Aim: - Implementation and analysis of Apriori Algorithm using Market Basket Analysis.

Theory: -

Apriori algorithm is used for finding frequent itemsets in a dataset for boolean association rule. Name of the algorithm is Apriori because it uses prior knowledge of frequent itemset properties. We apply an iterative approach or level-wise search where k-frequent itemsets are used to find k+1 itemsets.

R Commands:

```
setwd("D:/13_Fymca")
getwd()
mba_data<-read.csv("data_apriori.csv")
trans<-
split(mba_data$Products,mba_data$Customer_
Id,"transactions")
head(trans)
```

```
> head(trans)
$`1`
[1] "bread"  "butter" "eggs"   "milk"

$`2`
[1] "beer"   "bread"  "cheese" "chips"  "mayo"   "soda"

$`3`
[1] "bread"   "butter"  "eggs"    "milk"    "oranges"

$`4`
[1] "bread"  "butter" "eggs"   "milk"   "soda"

$`5`
[1] "buns"    "chips"   "beer"    "mustard" "pickels" "soda"

$`6`
[1] "bread"     "butter"    "chocolate" "eggs"      "milk"
```

```
install.packages("arules")
library(arules)
rules=apriori(trans,parameter =
list(support=0.5,confidence=0.9,maxlen=3,min
len=2))
inspect(rules)
```

```
> inspect(rules)
    lhs             rhs      support   confidence coverage  lift     count
[1]  {eggs}         => {milk}  0.6000000 1          0.6000000 1.666667 9
[2]  {milk}         => {eggs}  0.6000000 1          0.6000000 1.666667 9
[3]  {butter}       => {bread} 0.6000000 1          0.6000000 1.250000 9
[4]  {butter, eggs} => {milk}  0.5333333 1          0.5333333 1.666667 8
[5]  {butter, milk} => {eggs}  0.5333333 1          0.5333333 1.666667 8
[6]  {bread, eggs}  => {milk}  0.5333333 1          0.5333333 1.666667 8
[7]  {bread, milk}  => {eggs}  0.5333333 1          0.5333333 1.666667 8
[8]  {butter, eggs} => {bread} 0.5333333 1          0.5333333 1.250000 8
[9]  {bread, eggs}  => {butter} 0.5333333 1          0.5333333 1.666667 8
[10] {butter, milk} => {bread} 0.5333333 1          0.5333333 1.250000 8
[11] {bread, milk}  => {butter} 0.5333333 1          0.5333333 1.666667 8
```

Practical No: 10

Aim: - Implementation and analysis of clustering algorithms like K-Means , Agglomerative.

Theory:

K-Means Clustering is an Unsupervised Learning algorithm, which groups the unlabeled dataset into different clusters. Here K defines the number of pre-defined clusters that need to be created in the process, as if K=2, there will be two clusters, and for K=3, there will be three clusters, and so on.

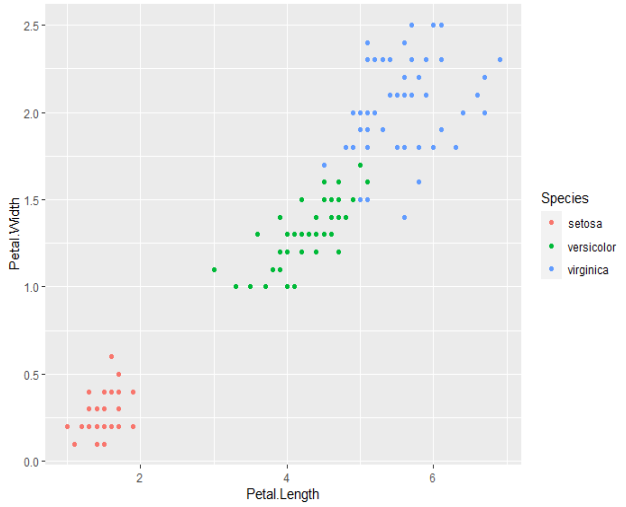
R Commands:-

head(iris)

install.packages("ggplot2",dependencies = TRUE)

library(ggplot2)

ggplot(iris, aes(Petal.Length, Petal.Width, color = Species)) + geom_point()



set.seed(20)

irisCluster <- kmeans(iris[, 3:4], 3, nstart = 20)

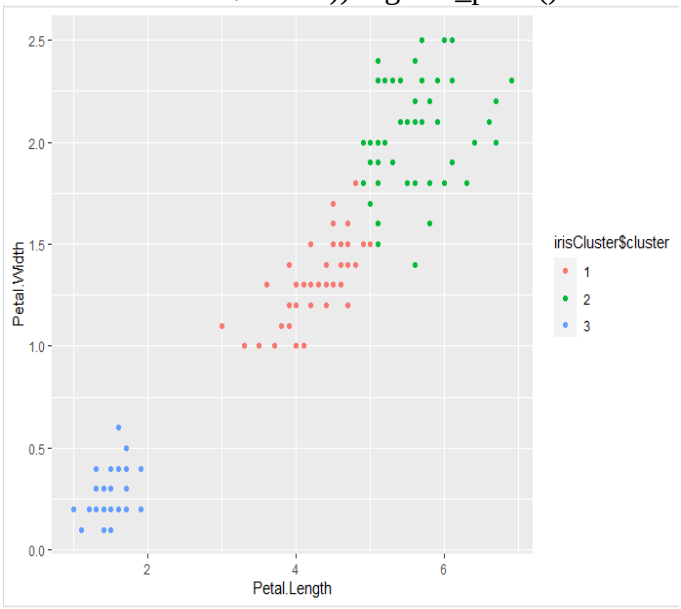
irisCluster

```
> irisCluster
K-means clustering with 3 clusters of sizes 52, 48, 50

Cluster means:
  Petal.Length Petal.Width
1     4.269231    1.342308
2     5.595833    2.037500
3     1.462000    0.246000

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
 [49] 3 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 2 1 1 1 1 1 1 1 1
 [97] 1 1 1 1 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2
[145] 2 2 2 2 2 2

Within cluster sum of squares by cluster:
[1] 13.05769 16.29167  2.02200
 (between_SS / total_SS =  94.3 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss" "betweenss"
[7] "size"         "iter"         "ifault"
>
```

```
> table(irisCluster$cluster, iris$Species)

    setosa versicolor virginica
  1      0         48         4
  2      0          2        46
  3     50          0         0
>
```

irisCluster$cluster <- as.factor(irisCluster$cluster)

ggplot(iris, aes(Petal.Length, Petal.Width, color =irisCluster$cluster)) + geom_point()

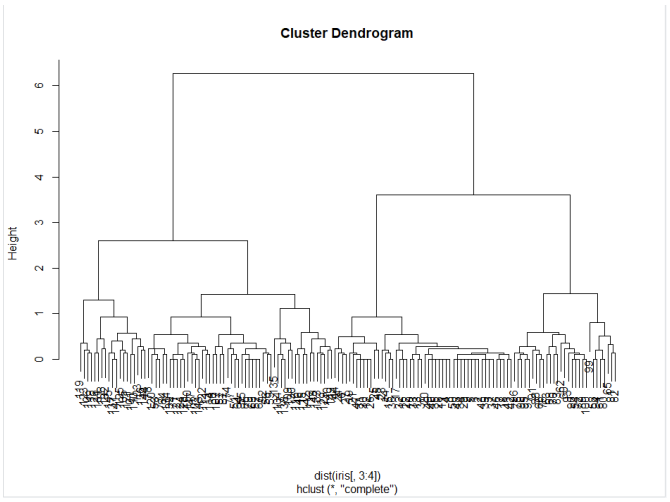


Name: Arif Shaikh **Roll No.39**

#Agglomerative Clustering

Agglomerative Clustering is Also known as bottom-up approach or hierarchical agglomerative clustering . A structure that is more informative than the unstructured set of clusters returned by flat clustering. This clustering algorithm does not require us to prespecify the number of clusters

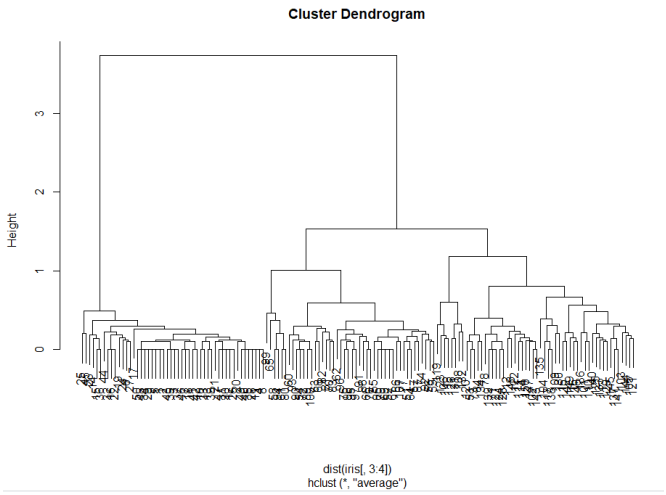
```
head(iris)
clusters <- hclust(dist(iris[, 3:4]))
plot(clusters)\
```

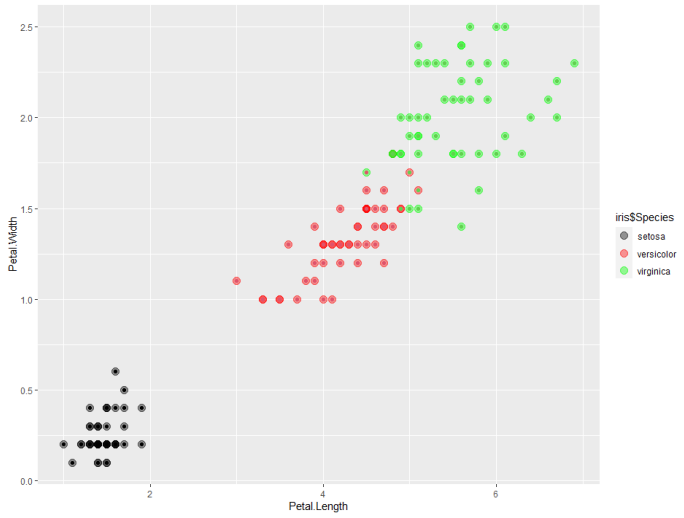



```
clusterCut<- cutree(clusters, 3)
table(clusterCut, iris$Species)
```

```
> table(clusterCut, iris$Species)

clusterCut setosa versicolor virginica
         1     50          0         0
         2      0         21        50
         3      0         29         0
>
```

```
clusters <- hclust(dist(iris[, 3:4]), method =
'average')
plot(clusters)
```






```
clusterCut <-cutree(clusters, 3)
table(clusterCut, iris$Species)
ggplot(iris, aes(Petal.Length, Petal.Width, color
= iris$Species)) +geom_point(alpha = 0.4, size
= 3.5) + geom_point(col = clusterCut) +
scale_color_manual(values = c('black','red',
'green'))
```