CHEMBUR TROMABY EDUCATION SOCIETY

N. G. ACHARYA & D. K. MARATHE COLLEGE OF ARTS, SCIENCE & COMMERCE

N. G. Acharya Marg, Chembur, Mumbai – 400 071.

NAAC ACCREDITED COLLEGE

M.Sc (Computer Science /Information Technology)

Certificate

Certified that the work entered in this journal was done in the Computer laboratory by the student Mr. / Miss Dalvi Siddhesh Dinesh Roll No. A1-5 Of Class M.Sc IT & A Semester II During the Year 2021-2022 in a Satisfactory manner.

Lecture Incharge	*External Examiner*	*Head of Dept*

Index

PRACTICAL NO 1

AIM - Perform K-means clustering on the IRIS dataset. Plot WSS to determine the optimum number of clusters to use

STEPS:

1- newiris=iris

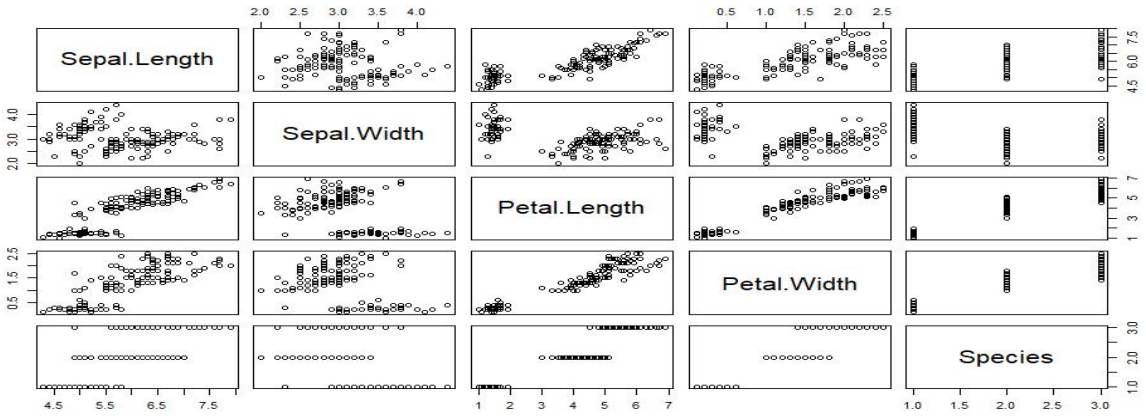
head(newiris)

pairs(newiris)

newiris$Species = NULL

newiris[1:5,]

```
> newiris=iris
> head(newiris)
  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
> pairs(newiris)
> newiris$Species = NULL
> newiris[1:5,]
  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
> |
```



2- # Apply kmeans to newiris, and store the clustering result in kc.

The cluster number is set to 3.

kc=kmeans(newiris,3,10)

kc

Check Cluster variables

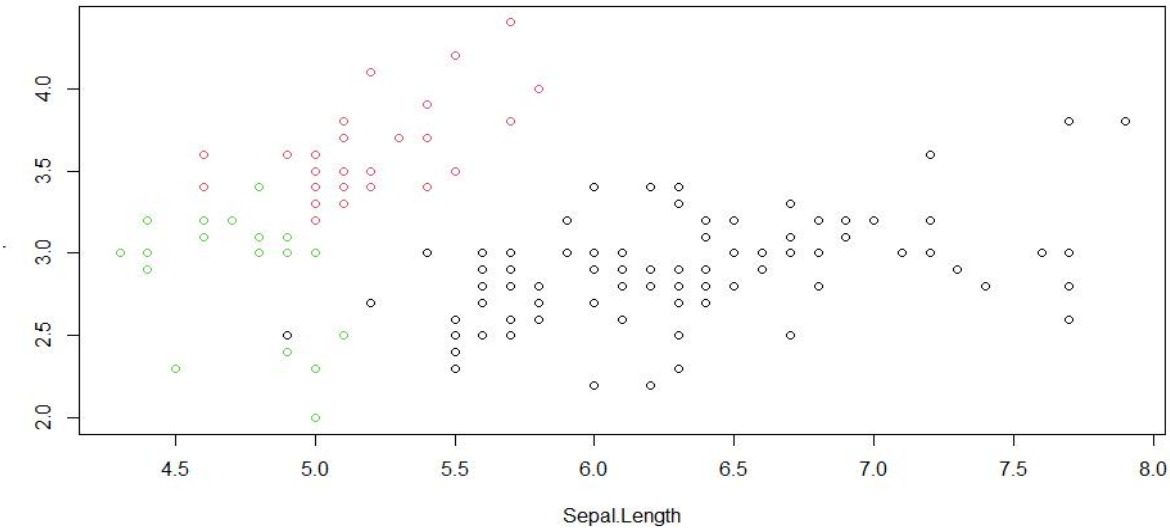
kc$cluster

table(iris$Species, kc$cluster)

Plot the clusters and their centres

plot(newiris[c("Sepal.Length", "Sepal.Width")], col=kc$cluster)

```
> # 2.  Apply kmeans to newiris, and store the clustering result in kc.
> #     The cluster number is set to 3.
> kc=kmeans(newiris,3,10)
> # Check cluster variables
> kc$cluster
  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3
 [52] 3 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 3
[103] 2 2 2 2 3 2 2 2 2 2 2 3 3 2 2 2 2 3 2 3 2 3 2 2 3 3 2 2 2 2 2 3 2 2 2 2 2 3 2 2 2 3 2 2 3
> table(iris$Species, kc$cluster)

            1  2  3
  setosa     50  0  0
  versicolor  0  2 48
  virginica   0 36 14
> # 4.  Plot the clusters and their centres
> plot(newiris[c("Sepal.Length", "Sepal.Width")], col=kc$cluster)
>
```



3- # Mark Cluster centers

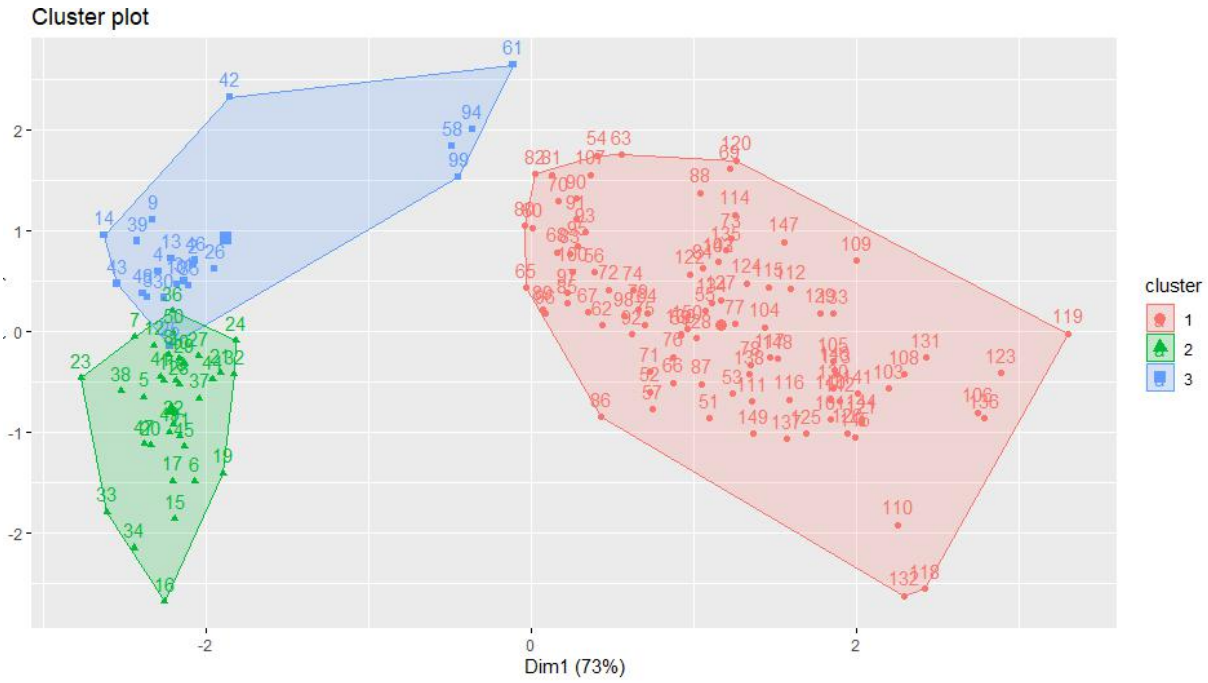
points(kc$centers[,c("Sepal.Length", "Sepal.Width")], col=1:3, pch=8, cex=3)

library('factoextra')

library('tidyverse')

library('ggplot2')

fviz_cluster(kc,data=newiris)



4- k2=kmeans(newiris,2,25)

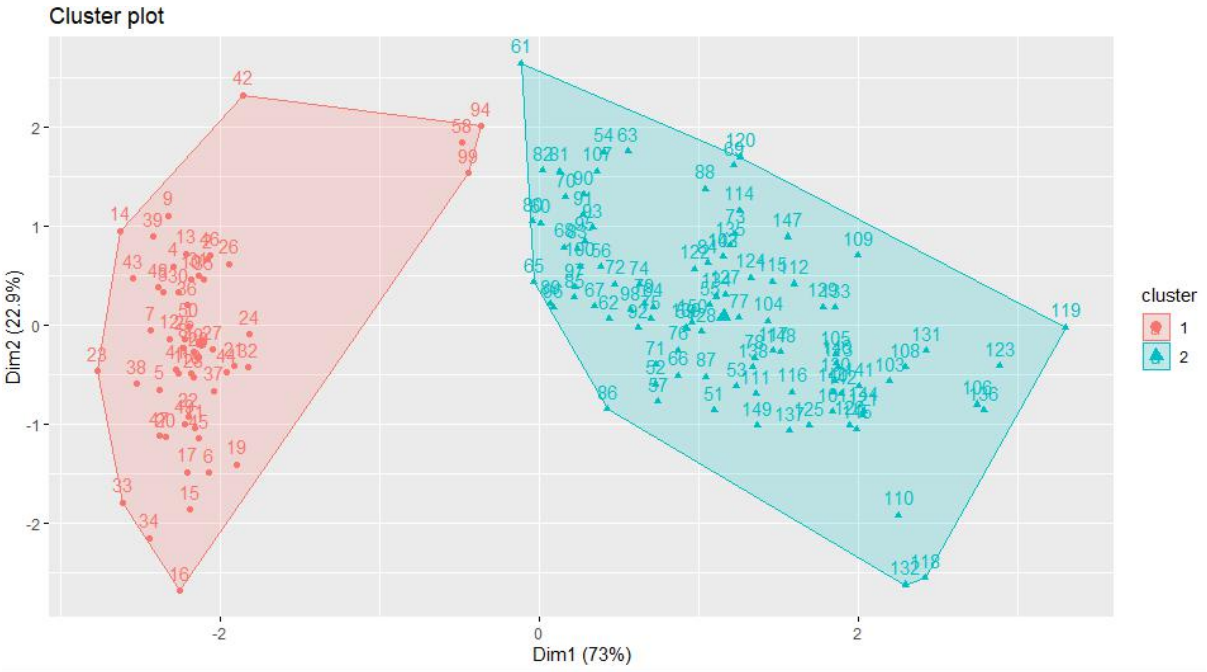
k3=kmeans(newiris,3,25)

k4=kmeans(newiris,4,25)

k5=kmeans(newiris,5,25)

ggplot(aes(Sepal.Length, Sepal.Width, color = factor(cluster))) + geom_text()

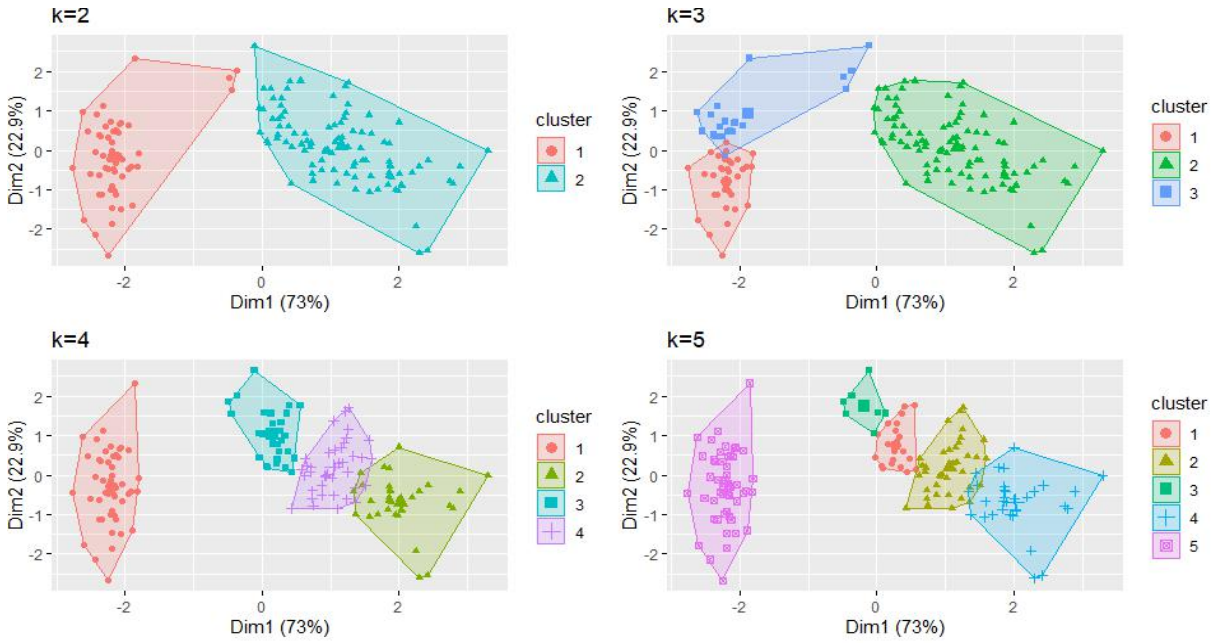
fviz_cluster(k2,data=newiris)



5- p1 = fviz_cluster(k2,data=newiris, geom='point') + ggtitle('k=2')

p2 = fviz_cluster(k3,data=newiris, geom='point') + ggtitle('k=3')

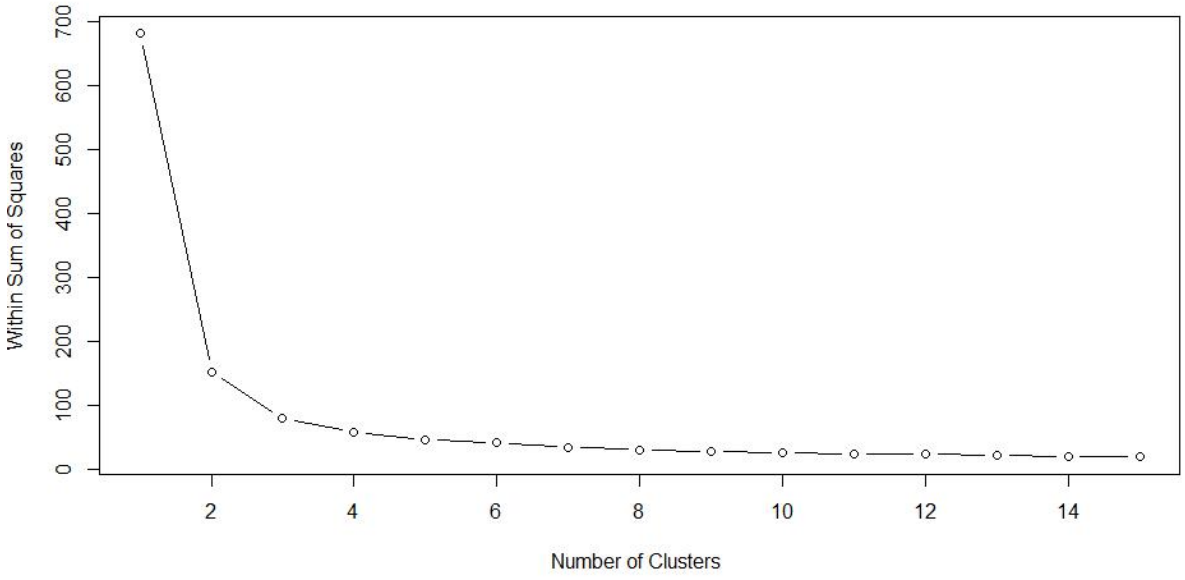
```
p3 = fviz_cluster(k4,data=newiris, geom='point') + ggtitle('k=4')
p4 = fviz_cluster(k5,data=newiris, geom='point') + ggtitle('k=5')
library('gridExtra')
grid.arrange(p1,p2,p3,p4, nrow=2)
```



6- #optimum number of clusters - wss

```
set.seed(125)
#compute and plot wss for k=1 to k=15
wss <- numeric(15)
wss
for (k in 1:15)
  wss[k]=sum(kmeans(newiris,centers=k, nstart=25 ) $withinss )
wss
# each WSS plotted against the respective number of centroids 1 to 15.
plot(1 : 15, wss, type="b", xlab="Number of Clusters " ,
ylab="Within Sum of Squares" )
# The WSS is greatly reduced when k increases from one to two.
# Another substantial reduction in WSS occurs at k = 3.
# However, the improvement in WSS is fairly linear fork > 3.
# Therefore, the k-means analysis will be conducted for k = 3.
# The process of identifying the appropriate value of k is
# referred to as finding the "elbow" of the WSS curve.
```

```
> # 2.  Apply kmeans to newiris, and store the clustering result in kc.
> #     The cluster number is set to 3.
> kc=kmeans(newiris,3,10)
> # Check Cluster variables
> kc$cluster
  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3
 [52] 3 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 3
[103] 2 2 2 2 3 2 2 2 2 2 2 3 3 2 2 2 2 3 2 3 2 3 2 2 3 3 2 2 2 2 2 3 2 2 2 2 3 2 2 2 3 2 2 3
> table(iris$Species, kc$cluster)

            1  2  3
  setosa     50  0  0
  versicolor  0  2 48
  virginica   0 36 14
> # 4.  Plot the clusters and their centres
> plot(newiris[c("Sepal.Length", "Sepal.Width")], col=kc$cluster)
>
```



CONCLUSION

The practical to perform K-means clustering on the IRIS dataset was successfully executed The WSS plot heled to determine the optimum number of clusters to be three (03)

PRACTICAL NO 2

AIM- Perform K-means clustering on the Student Grades dataset.

Plot WSS to determine the optimum number of clusters to use

STEPS

1- # Load libraries from installed pacakges

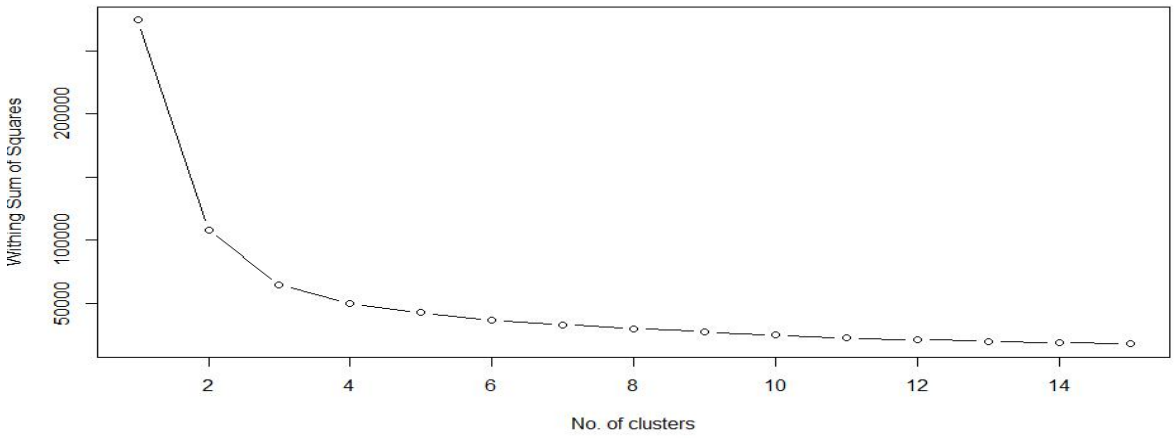
```
library (plyr)
library(ggplot2)
library(cluster)
library(lattice)
library(graphics)
library(grid)
library(gridExtra)
#How to set path of an input file in R - set Working Directory
# getwd in r - current working directory
getwd()
# To set working directory use- setwd()
setwd("C:/Users/PRIYANKA/Desktop/Sem 2/BDA Practicals")
grade_input = read.csv('C:/Users/PRIYANKA/Desktop/Sem 2/BDA Practicals/grades_km_input.csv')
str(grade_input)
kmdata = as.matrix(grade_input[,2:4])
str(kmdata)
kmdata[1:5,]
# To find appropriate number of clusters using elbow method- wss
wss = 15
for (k in 1:15)
  wss[k]=sum(kmeans(kmdata,centers=k,nstart=25)$withinss)
wss
plot(
  1:15,
  wss,
  type="b",
```

xlab = "No. of clusters",

ylab = "Withing Sum of Squares")

```
> str(grade_input)
'data.frame':   620 obs. of  4 variables:
 $ Student: int  1 2 3 4 5 6 7 8 9 10 ...
 $ English: int  99 99 98 95 95 96 100 95 98 99 ...
 $ Math   : int  96 96 97 100 96 97 96 98 96 99 ...
 $ Science: int  97 97 97 95 96 96 97 98 96 95 ...
> kmdata = as.matrix(grade_input[,2:4])
> str(kmdata)
 int [1:620, 1:3] 99 99 98 95 95 96 100 95 98 99 ...
 - attr(*, "dimnames")=List of 2
  ..$ : NULL
  ..$ : chr [1:3] "English" "Math" "Science"
> kmdata[1:5,]
     English Math Science
[1,]      99   96      97
[2,]      99   96      97
[3,]      98   97      97
[4,]      95  100      95
[5,]      95   96      96
>
```

```
# To find appropriate number of clusters using elbow method- wss
wss = 15
for (k in 1:15)
  wss[k]=sum(kmeans(kmdata,centers=k,nstart=25)$withinss)
wss
[1] 274067.06 108378.71  64483.06  50057.00  42931.51  36895.31  33311.87  30332.70  27700.44  25071.38
11]  22702.60  21377.40  20171.85  19175.14  18126.08
plot(
  1:15,
  wss,
  type="b",
  xlab = "No. of clusters",
  ylab = "Withing Sum of Squares")
```



2- # Peform KMeans clustering with 3 clusters

km = kmeans(kmdata,3, nstart=25)

km

#visualize the data and assigned clusters

#prepare the student data and clustering results for plotting

df=grade_input[,2:4]

str(df)

km$cluster

df$cluster = factor(km$cluster)

str(df$cluster)

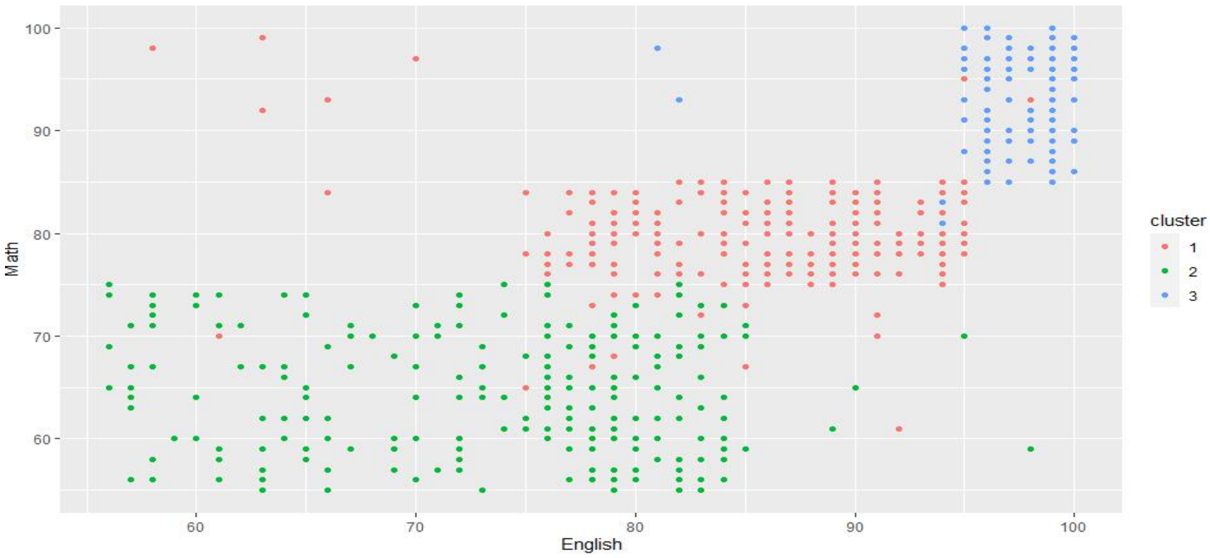
centers = as.data.frame(km$centers)

centers

ggplot(data=df, aes(x=English, y=Math, color=cluster))+

geom_point() + theme(legend.position="right")

```
> km = kmeans(kmdata,3, nstart=25)
> km
K-means clustering with 3 clusters of sizes 244, 218, 158

Cluster means:
   English     Math  Science
1 85.84426 79.68033 81.50820
2 73.22018 64.62844 65.84862
3 97.21519 93.37342 94.86076

Clustering vector:
  [1] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
 [52] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
[103] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 1 1
[154] 1 1 1 1 1 1 1 1 1 1 1 1 1 3 1 1 1 1 1 1 1 1 1 1 3 3 1 1 3 1 1 1 3 1 1 1 1 1 1 3 1 1 1 1 1 1 1 1 1 1 1
[205] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[256] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[307] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[358] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 1 2 1 2 1 1 1 2 2 2 2 1 1 2 2 2 2 2 2 2 2 2 2 2 2
[409] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2
[460] 2 2 2 2 2 2 1 2 2 2 1 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
[511] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
```



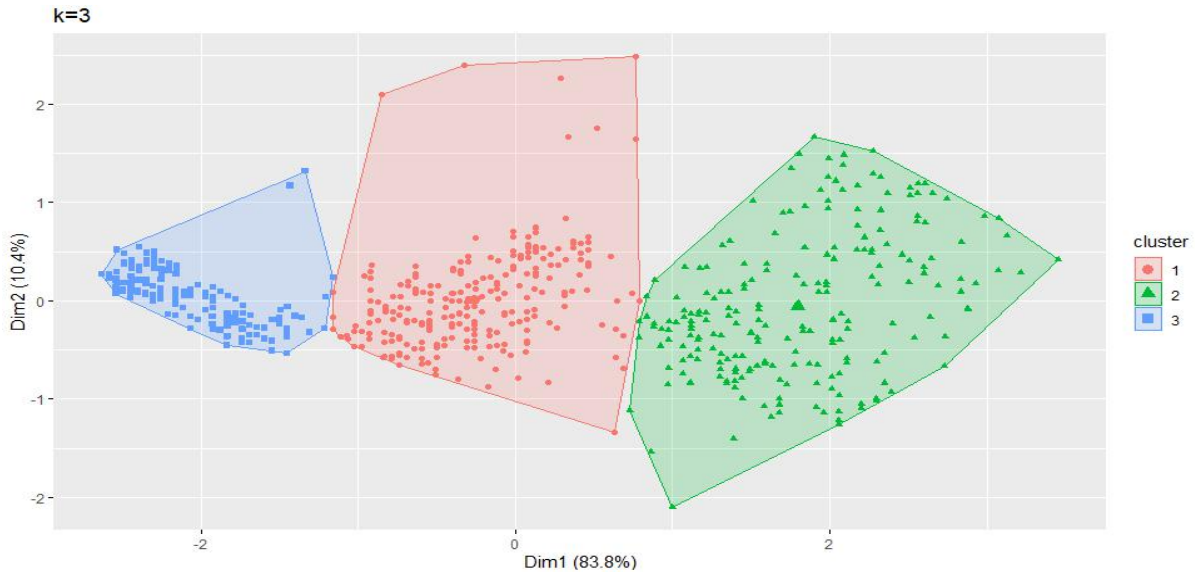
3- library('factoextra')

library('tidyverse')

library('ggplot2')

fviz_cluster(km,data=kmdata, geom='point') + ggtitle('k=3')

```
[358] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 1 2 1 2 1 1 1 2 2 2 2 1 1 2 2 2 2 2 2 2 2 2 2
[409] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2
[460] 2 2 2 2 2 2 1 2 2 2 1 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
[511] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
[562] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 2 2 1 1 1 1 3 3 1 1
[613] 1 2 2 1 2 1 1 1
> df$cluster = factor(km$cluster)
> str(df$cluster)
 Factor w/ 3 levels "1","2","3": 3 3 3 3 3 3 3 3 3 3 ...
> centers = as.data.frame(km$centers)
> centers
   English     Math  Science
1 85.84426 79.68033 81.50820
2 73.22018 64.62844 65.84862
3 97.21519 93.37342 94.86076
> ggplot(data=df, aes(x=English, y=Math, color=cluster ))+
+   geom_point() + theme(legend.position="right")
> library('factoextra')
> library('tidyverse')
> library('ggplot2')
> fviz_cluster(km,data=kmdata, geom='point') + ggtitle('k=3')
```



4- g1= ggplot(data=df, aes(x=English, y=Math, color=cluster)) +

geom_point() + theme(legend.position="right") +

geom_point(data=centers,

aes(x=English,y=Math, color=as.factor(c(1,2,3))),

size=10, alpha=.3,show.legend = FALSE)

g2 =ggplot(data=df, aes(x=English, y=Science, color=cluster)) +

geom_point () +

geom_point(data=centers,

aes(x=English,y=Science, color=as.factor(c(1,2,3))),

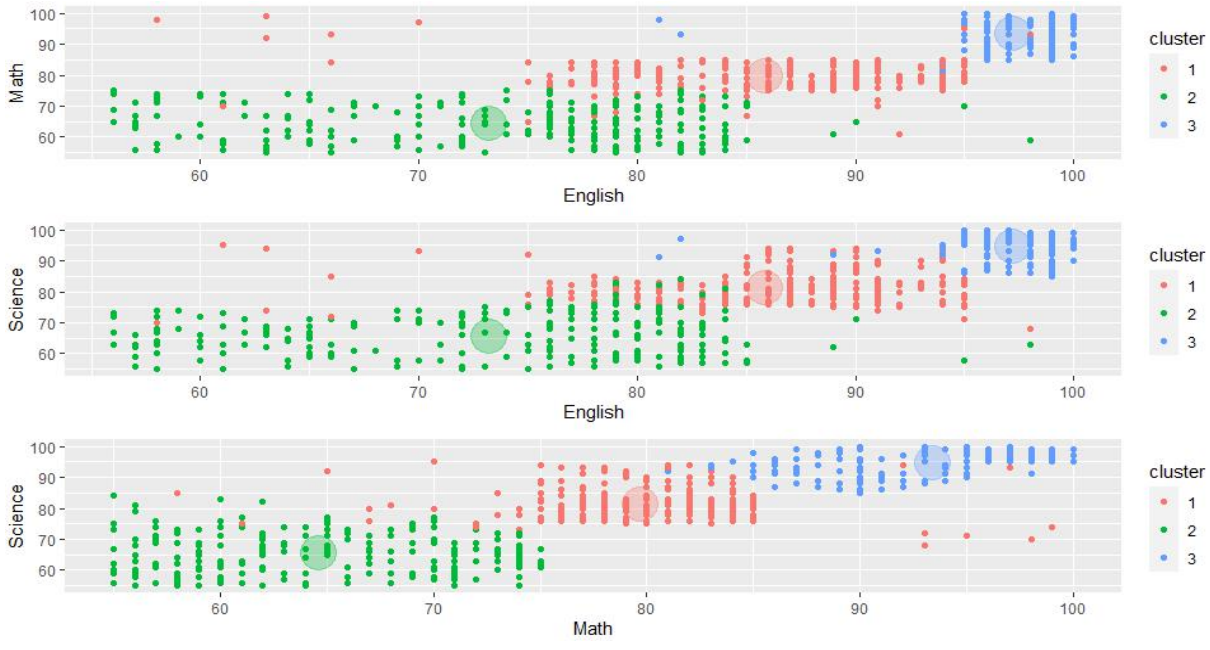
size=10, alpha=.3,show.legend = FALSE)

g3 = ggplot(data=df, aes(x=Math, y=Science, color=cluster)) +

geom_point () +

```
 geom_point(data=centers,
            aes(x=Math,y=Science, color=as.factor(c(1,2,3))),
            size=10, alpha=0.3,show.legend = FALSE)


tmp = ggplot_gtable(ggplot_build(g1))
library('gridExtra')
grid.arrange(g1,g2,g3, nrow=3)
```

```
> library('ggplot2')
> fviz_cluster(km,data=kmdata, geom='point') + ggtitle('k=3')
> g1= ggplot(data=df, aes(x=English, y=Math, color=cluster )) +
+   geom_point() + theme(legend.position="right") +
+   geom_point(data=centers,
+              aes(x=English,y=Math, color=as.factor(c(1,2,3))),
+              size=10, alpha=.3,show.legend = FALSE)
>
> g2 =ggplot(data=df, aes(x=English, y=Science, color=cluster )) +
+   geom_point () +
+   geom_point(data=centers,
+              aes(x=English,y=Science, color=as.factor(c(1,2,3))),
+              size=10, alpha=.3,show.legend = FALSE)
> g3 = ggplot(data=df, aes(x=Math, y=Science, color=cluster )) +
+   geom_point () +
+   geom_point(data=centers,
+              aes(x=Math,y=Science, color=as.factor(c(1,2,3))),
+              size=10, alpha=0.3,show.legend = FALSE)
> tmp = ggplot_gtable(ggplot_build(g1))
> library('gridExtra')
> grid.arrange(g1,g2,g3, nrow=3)
```



CONCLUSION

The practical to Perform K-means clustering on the Student Grades dataset was successfully executed. The WSS plot heled to determine the optimum number of clusters to be three (03)

PRACTICAL NO 3

AIM - Perform Hierarchical clustering on the Utilities dataset.

Plot its dendogram & Silhoutte Plot

STEPS:

1- #Clustering in the utlities dataset

utilities = read.csv("C:/Users/PRIYANKA/Desktop/Sem 2/BDA Practicals/utilities.csv")

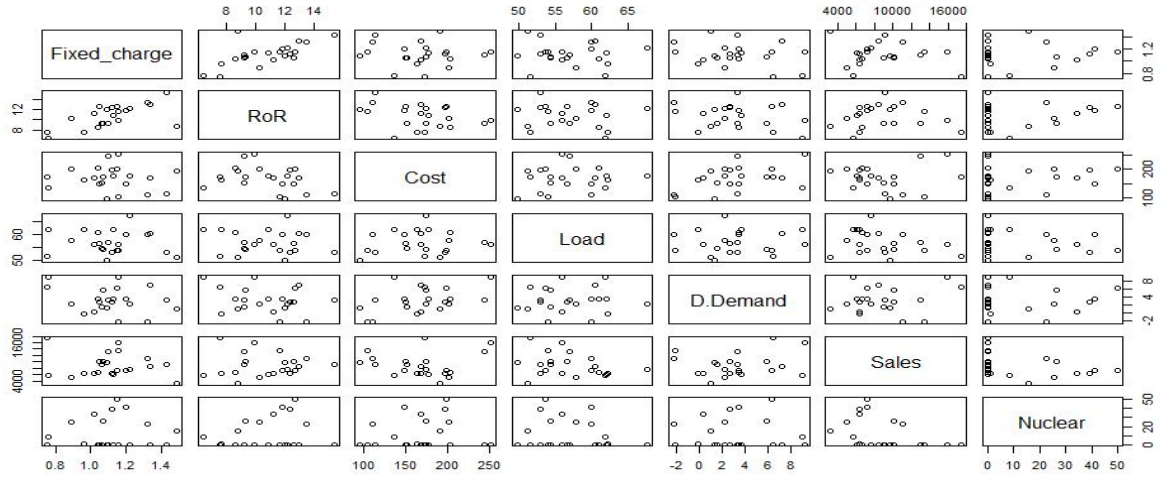
str(utilities)

utilities

all possible plots

pairs(utilities[,2:8])

```
 str(utilities)
data.frame':   22 obs. of  9 variables:
$ Company     : chr  "Arizona " "Boston " "Central " "Commonwealth" ...
$ Fixed_charge: num  1.06 0.89 1.43 1.02 1.49 1.32 1.22 1.1 1.34 1.12 ...
$ RoR         : num  9.2 10.3 15.4 11.2 8.8 13.5 12.2 9.2 13 12.4 ...
$ Cost        : int  151 202 113 168 192 111 175 245 168 197 ...
$ Load        : num  54.4 57.9 53 56 51.2 60 67.6 57 60.4 53 ...
$ D.Demand    : num  1.6 2.2 3.4 0.3 1 -2.2 2.2 3.3 7.2 2.7 ...
$ Sales       : int  9077 5088 9212 6423 3300 11127 7642 13082 8406 6455 ...
$ Nuclear     : num  0 25.3 0 34.3 15.6 22.5 0 0 0 39.2 ...
$ Fuel_Cost   : num  0.628 1.555 1.058 0.7 2.044 ...
 utilities
       Company Fixed_charge  RoR Cost Load D.Demand Sales Nuclear Fuel_Cost
      Arizona          1.06  9.2  151 54.4      1.6  9077     0.0     0.628
       Boston          0.89 10.3  202 57.9      2.2  5088    25.3     1.555
      Central          1.43 15.4  113 53.0      3.4  9212     0.0     1.058
  Commonwealth         1.02 11.2  168 56.0      0.3  6423    34.3     0.700
     Con Ed NY         1.49  8.8  192 51.2      1.0  3300    15.6     2.044
      Florida          1.32 13.5  111 60.0     -2.2 11127    22.5     1.241
     Hawaiian          1.22 12.2  175 67.6      2.2  7642     0.0     1.652
        Idaho          1.10  9.2  245 57.0      3.3 13082     0.0     0.309
```



2- # scatterplot of Fuel_cost vs Sales

plot(Fuel_Cost ~ Sales,utilities)

with(utilities,text(Fuel_Cost ~ Sales,labels=Company, pos=1,cex=0.7))

utilities

Normalization - Avg = 0 & std = 1

z=utilities[,2:9]

z

Calculate mean and Standard dev for entire data set columnwise

m=apply(z,2,mean)

s=apply(z,2,sd)

m

s

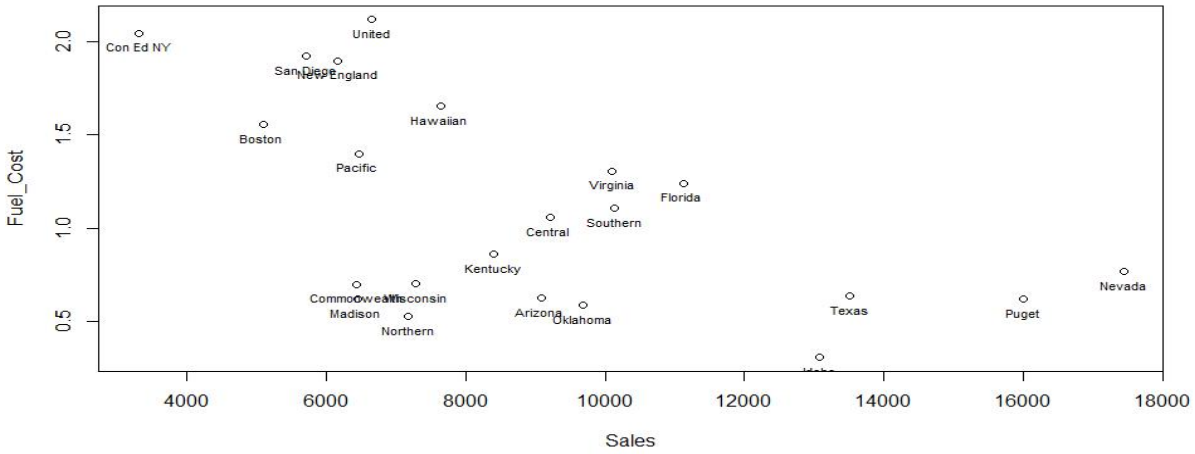
z=scale(z,m,s)

#calculating euclidean distance

distance = dist(z)

print(distance,digits=3)

```
> # all possible plots
> pairs(utilities[,2:8])
> # scatterplot of Fuel_cost vs Sales
> plot(Fuel_Cost ~ Sales,utilities)
> with(utilities,text(Fuel_Cost ~ Sales,labels=Company, pos=1,cex=0.7))
> utilities
         Company Fixed_charge  RoR Cost Load D.Demand Sales Nuclear Fuel_Cost
1        Arizona          1.06  9.2  151 54.4      1.6  9077     0.0     0.628
2         Boston          0.89 10.3  202 57.9      2.2  5088    25.3     1.555
3        Central          1.43 15.4  113 53.0      3.4  9212     0.0     1.058
4   Commonwealth          1.02 11.2  168 56.0      0.3  6423    34.3     0.700
5      Con Ed NY          1.49  8.8  192 51.2      1.0  3300    15.6     2.044
6        Florida          1.32 13.5  111 60.0     -2.2 11127    22.5     1.241
7        Hawaiian         1.22 12.2  175 67.6      2.2  7642     0.0     1.652
8           Idaho         1.10  9.2  245 57.0      3.3 13082     0.0     0.309
9        Kentucky         1.34 13.0  168 60.4      7.2  8406     0.0     0.862
10       Madison          1.12 12.4  197 53.0      2.7  6455    39.2     0.623
11         Nevada         0.75  7.5  173 51.5      6.5 17441     0.0     0.768
> # calculate mean and standard dev for entire data set columnwise
> m=apply(z,2,mean)
> s=apply(z,2,sd)
> m
Fixed_charge          RoR         Cost         Load     D.Demand        Sales      Nuclear    Fuel_Cost
    1.114091    10.736364   168.181818    56.977273     3.240909  8914.045455    12.000000     1.102727
> s
Fixed_charge          RoR         Cost         Load     D.Demand        Sales      Nuclear    Fuel_Cost
   0.1845112    2.2440494   41.1913495    4.4611478    3.1182503 3549.9840305   16.7919198    0.5560981
> z=scale(z,m,s)
>
> #calculating euclidean distance
> distance = dist(z)
> print(distance,digits=3)
     1    2    3    4    5    6    7    8    9   10   11   12   13   14   15   16   17   18   19   20   21
2  3.10
3  3.68 4.92
4  2.46 2.16 4.11
5  4.12 3.85 4.47 4.13
6  3.61 4.22 2.99 3.20 4.60
7  3.90 3.45 4.22 3.97 4.60 3.35
8  2.74 3.89 4.99 3.69 5.16 4.91 4.36
```



3- # Hierarchical agglomerative clustering

Cluster dendogram with complete linkage

hc.c=hclust(distance)

plot(hc.c)

plot(hc.c,labels = utilities$Company)

plot(hc.c,hang=-1)

Cluster dendogram with average linkage

hc.a=hclust(distance, method = "average")

plot(hc.a)

plot(hc.a,labels = utilities$Company)

plot(hc.a,hang=-1)

cluster membership

member.c=cutree(hc.c,3)

member.c

member.a=cutree(hc.a,3)

member.a

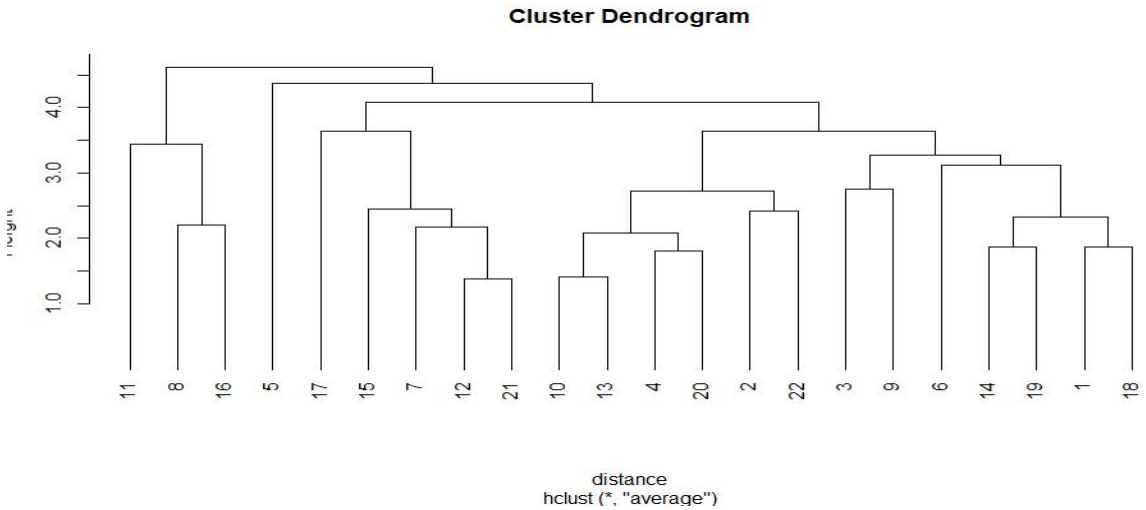
table(member.c, member.a)

cluster means

aggregate(z, list(member.c), mean)

aggregate(utilities[,-c(1,1)], list(member.c), mean)

```
> table(member.c, member.a)
        member.a
member.c  1  2  3
       1 13  1  0
       2  5  0  0
       3  0  0  3
>
> # cluster means
> aggregate(z, list(member.c), mean)
  Group.1 Fixed_charge        RoR        Cost       Load   D.Demand      Sales    Nuclear  Fuel_Cost
1       1    0.3068832  0.4326015 -0.31481203 -0.3743722 -0.2605107 -0.1575387  0.3692252 -0.2389329
2       2   -0.4991075 -0.7113763  0.07812761  1.3365904  0.1343994 -0.6728046 -0.6050529  1.2484717
3       3   -0.6002757 -0.8331800  1.33891013 -0.4805802  0.9917178  1.8565214 -0.7146294 -0.9657660
> aggregate(utilities[,-c(1,1)], list(member.c), mean)
  Group.1 Fixed_charge       RoR     Cost     Load D.Demand     Sales Nuclear Fuel_Cost
1       1     1.170714 11.707143 155.2143 55.30714 2.428571  8354.786   18.20 0.9698571
2       2     1.022000  9.140000 171.4000 62.94000 3.660000  6525.600    1.84 1.7970000
3       3     1.003333  8.866667 223.3333 54.83333 6.333333 15504.667    0.00 0.5656667
>
```



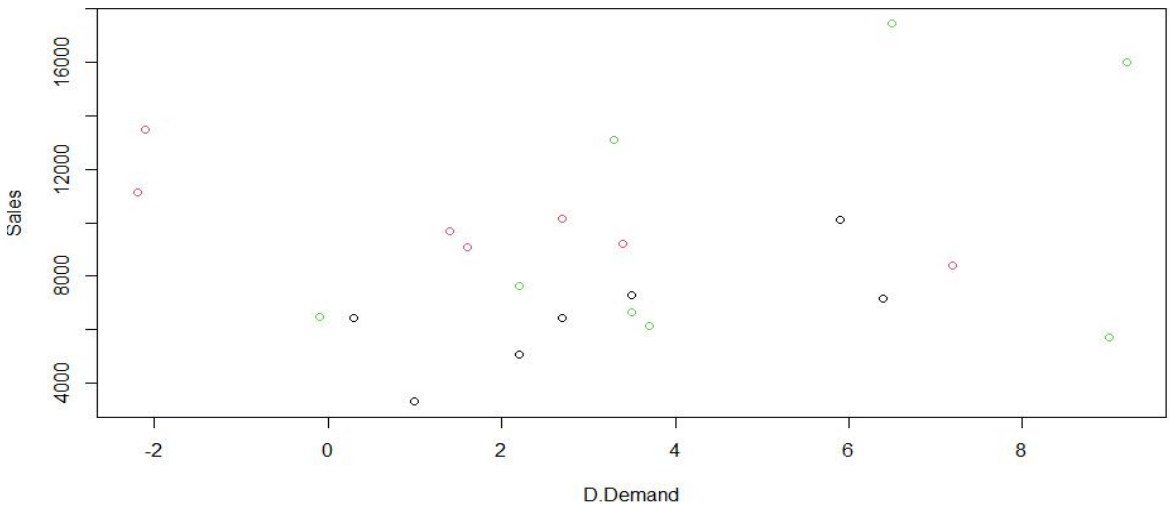
4- # Silhoutte Plot

```
library(cluster)
plot(silhouette(cutree(hc.c,3),distance))
#Scree Plot
wss <- numeric(20)
for (k in 1:20)
  wss[k]=sum(kmeans(z,centers=k, nstart=25 ) $withinss )
wss
plot(1 : 20, wss, type="b", xlab="Number of Clusters " ,
    ylab="Within Sum of Squares" )
kc = kmeans(z,3)
kc
member.a
member.c
kc$cluster
```

plot(Sales ~ D.Demand, utilities)

plot(Sales ~ D.Demand, utilities, col =kc$cluster)

```
Clustering vector:
 [1] 1 3 1 3 3 1 2 2 1 3 2 2 3 1 2 2 2 1 1 3 2 3

Within cluster sum of squares by cluster:
[1] 26.50777 58.25276 27.69741
 (between_SS / total_SS =  33.1 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss" "betweenss"    "size"
[8] "iter"         "ifault"
> member.a
 [1] 1 1 1 1 2 1 1 3 1 1 3 1 1 1 1 3 1 1 1 1 1 1
> member.c
 [1] 1 1 1 1 1 1 2 3 1 1 3 2 1 1 2 3 2 1 1 1 2 1
> kc$cluster
 [1] 1 3 1 3 3 1 2 2 1 3 2 2 3 1 2 2 2 1 1 3 2 3
>
> plot(Sales ~ D.Demand, utilities)
> plot(Sales ~ D.Demand, utilities, col =kc$cluster)
```



CONCLUSION

The Practical to Perform Hierarchical clustering on the Utilities dataset and Plot of dendogram & Silhoutte Plot was successfully executed.

PRACTICAL NO 4

AIM- Perform Association mining on the Groceries dataset.

Generate frequent itemset of size 1, 2, 3 & 4.

Generate & Plot rules with support=0.001, confidence=0.6

STEPS:

1- library ('arules')

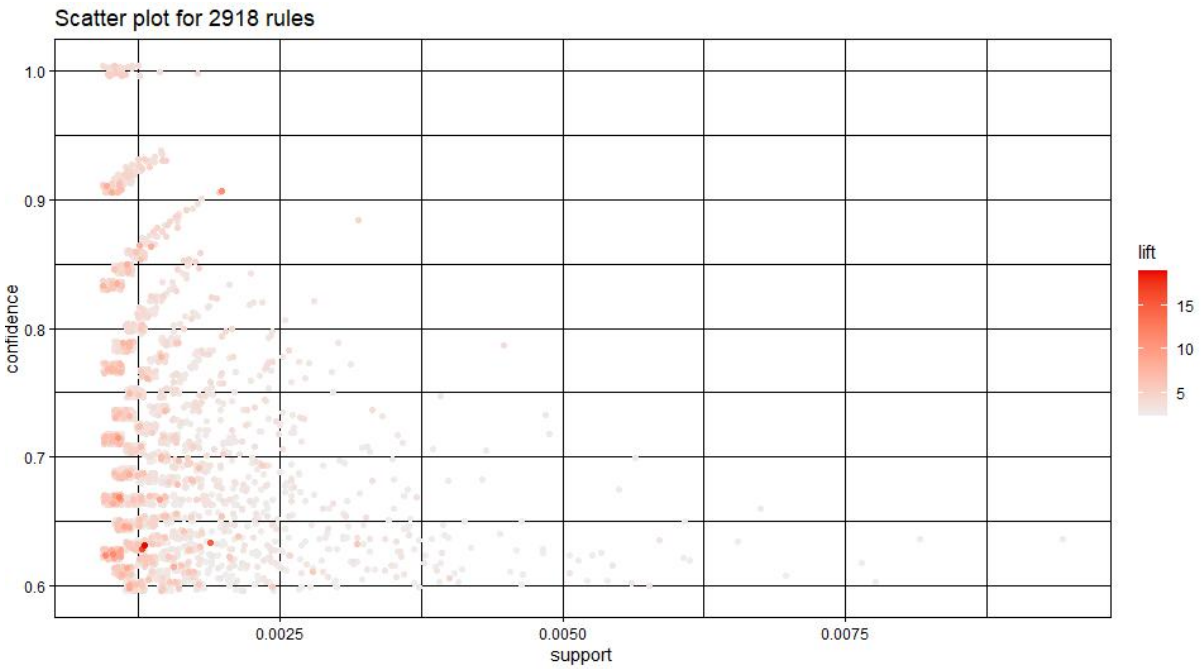
```
library( 'arulesViz' )
data("Groceries")
View(Groceries)
summary(Groceries)
class(Groceries)
Groceries@itemInfo[1:20,]
Groceries@itemsetInfo[1:20,]
g=Groceries@itemsetInfo[1:20,]
apply(Groceries@data[,1:30],2,
    function(r) paste(Groceries@itemInfo[r,"labels"], collapse = ", "))
# Generating frequent itemset of size 1
itemsets = apriori(Groceries)
itemsets = apriori(Groceries, parameter =
                list(minlen=1,
                    maxlen=1,
                    support=0.02,
                    target="frequent itemsets"))

inspect(itemsets)
summary(itemsets)
# The top 10 most frequent 1-itemsets
inspect(head(sort(itemsets, by = "support"),10))
# Generating frequent itemset of size 2
itemsets = apriori(Groceries, parameter =
```

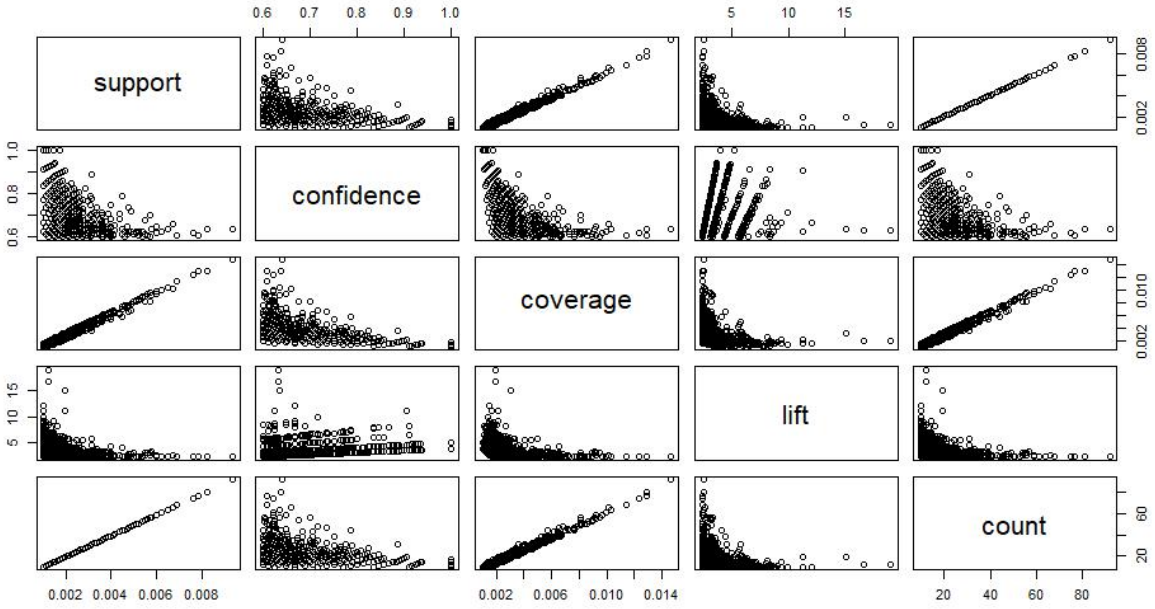
```
                list(minlen=2,
                    maxlen=2,
                    support=0.02,
                    target="frequent itemsets"))

inspect(itemsets)
summary(itemsets)
# The top 10 most frequent 2-itemsets
inspect(head(sort(itemsets, by = "support"),10))
# Generating frequent itemset of size 3
itemsets = apriori(Groceries, parameter =
                list(minlen=3,
                    maxlen=3,
                    support=0.02,
                    target="frequent itemsets"))

inspect(itemsets)
summary(itemsets)
# The top 10 most frequent 3-itemsets
inspect(head(sort(itemsets, by = "support"),10))

# Generating frequent itemset of size 4
itemsets = apriori(Groceries, parameter =
                list(minlen=4,
                    maxlen=4,
                    support=0.02,
                    target="frequent itemsets"))

inspect(itemsets)
summary(itemsets)
#Rule Generation
```

```
rules = apriori(Groceries,
                parameter=list(support=0.001,
                               confidence=0.6,
                               target="rules"))
inspect(rules)
summary(rules)
inspect(head(sort(rules, by = "support"),10))
# Rule Visualization
plot(rules)
plot(rules@quality)
```

Groceries	S4 [9835 x 169] (arules::transactic	S4 object of class transactions
data	S4 [169 x 9835] (Matrix::ngCMatr	S4 object of class ngCMatrix
itemInfo	list [169 x 3] (S3: data.frame)	A data.frame with 169 rows and 3 columns
itemsetInfo	list [0 x 0] (S3: data.frame)	A data.frame with 0 rows and 0 columns

```
rule length distribution (lhs + rhs):sizes
   2    3    4    5    6
   3  490 1765  626   34

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  2.000   4.000   4.000   4.068   4.000   6.000

summary of quality measures:
    support           confidence        coverage             lift            count
 Min.   :0.001017   Min.   :0.6000   Min.   :0.001017   Min.   : 2.348   Min.   :10.00
 1st Qu.:0.001118   1st Qu.:0.6316   1st Qu.:0.001525   1st Qu.: 2.668   1st Qu.:11.00
 Median :0.001220   Median :0.6818   Median :0.001830   Median : 3.168   Median :12.00
 Mean   :0.001480   Mean   :0.7028   Mean   :0.002157   Mean   : 3.450   Mean   :14.55
 3rd Qu.:0.001525   3rd Qu.:0.7500   3rd Qu.:0.002339   3rd Qu.: 3.692   3rd Qu.:15.00
 Max.   :0.009354   Max.   :1.0000   Max.   :0.014642   Max.   :18.996   Max.   :92.00

mining info:
      data ntransactions support confidence
 Groceries          9835   0.001        0.6
                                                                                    call
 apriori(data = Groceries, parameter = list(support = 0.001, confidence = 0.6, target = "rules"))
> inspect(head(sort(rules, by = "support"),10))
```





2- inspect(head(sort(rules, by = "lift"),10))

confidentRules = rules[quality(rules)$confidence>0.9]

confidentRules

###

plot(confidentRules,

method="matrix",

measure = c("lift", "confidence"),

control=list(reorder=TRUE)

)

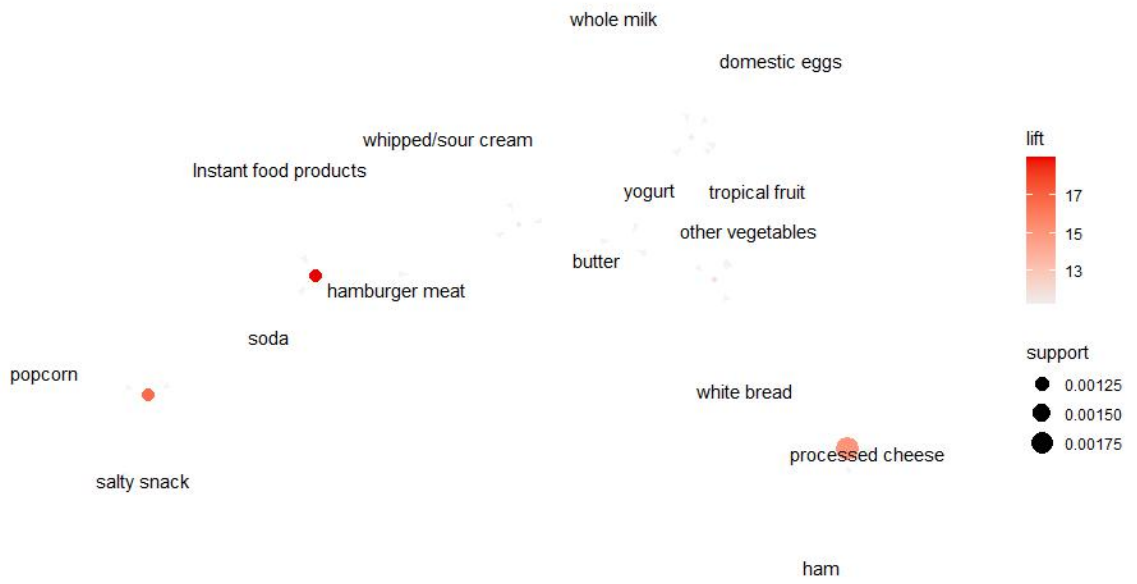
highLiftRules = head(sort (rules, by= "lift ") , 5)

highLiftRules = head(sort(rules, by = "lift"))

highLiftRules

plot(highLiftRules, method = "graph", control = list(type="items"))

```
> highLiftRules = head(sort(rules, by = "lift"))
> highLiftRules
set of 6 rules
> plot(highLiftRules, method = "graph", control = list(type="items"))
Warning: Unknown control parameters: type
Available control parameters (with default values):
layout   =  stress
circular         =  FALSE
ggraphdots       =  NULL
edges    =  <environment>
nodes    =  <environment>
nodetext         =  <environment>
colors   =  c("#EE0000FF", "#EEEEEEFF")
engine   =  ggplot2
max      =  100
verbose  =  FALSE
>
```



CONCLUSION

The Practical to Perform Association mining on the Groceries dataset was successfully executed.

PRACTICAL NO 5

AIM - Create a decision tree for the Bank-Sample dataset. Predict the outcome for following job="retired", marital="married", education="secondary", default="no", housing="yes", loan="no", contact = "cellular", duration = 598, poutcome="unknown"

STEPS:

1-library("rpart")

library("rpart.plot")

Read the data

setwd("C:\Users\PRIYANKA\Desktop\Sem 2\BDA Practicals")

banktrain <- read.table("bank-sample.csv",header=TRUE,sep=",")

View(banktrain)

drop a few columns to simplify the tree

drops<-c("age", "balance", "day", "campaign", "pdays", "previous", "month")

banktrain <- banktrain [,!(names(banktrain) %in% drops)]

View(banktrain)

summary(banktrain)

Make a simple decision tree by only keeping the categorical variables

```
fit <- rpart(subscribed ~ job + marital + education + default + housing + loan + contact + poutcome,
        method="class",
        data=banktrain,
        control=rpart.control(minsplit=1),
        parms=list(split='information'))
```

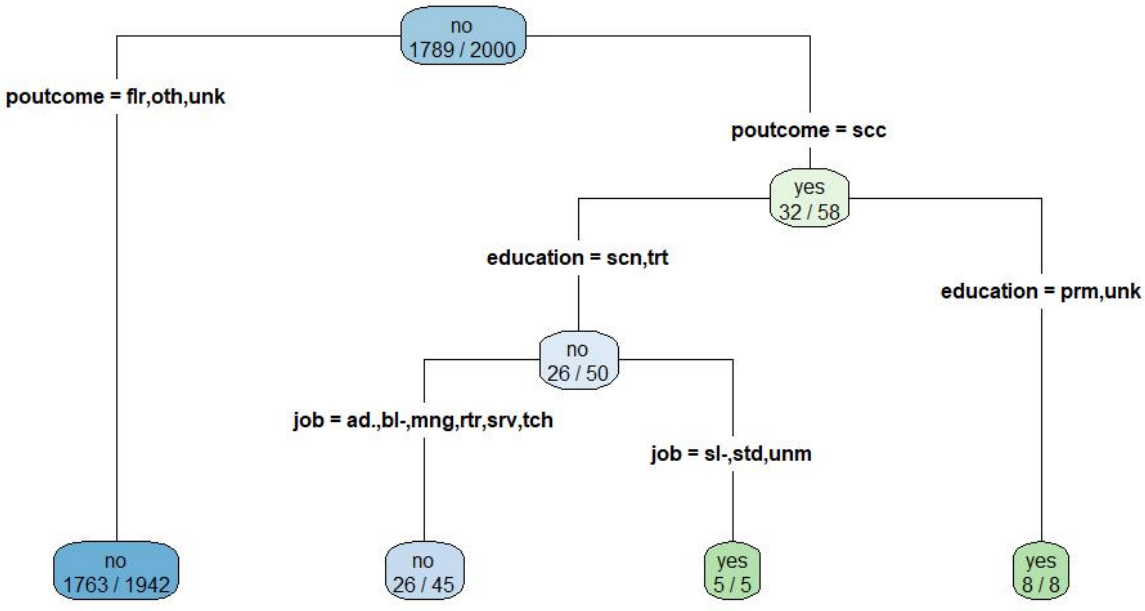
summary(fit)

Plot the tree

rpart.plot(fit, type=4, extra=2, clip.right.labs=FALSE, varlen=0, faclen=3)

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign
1	31	management	single	tertiary	no	0	yes	no	cellular	15	apr	185	
2	45	entrepreneur	married	tertiary	no	1752	yes	yes	cellular	20	nov	56	
3	46	services	divorced	secondary	no	4329	no	no	cellular	21	nov	534	
4	35	management	married	tertiary	no	1108	yes	no	cellular	17	nov	52	
5	39	management	married	secondary	no	1410	yes	no	unknown	23	may	55	
6	31	management	single	tertiary	no	499	yes	no	unknown	9	jun	122	
7	34	entrepreneur	married	tertiary	no	0	yes	no	cellular	6	may	611	
8	39	admin.	married	secondary	no	26233	no	no	cellular	2	oct	462	
9	38	blue-collar	married	secondary	no	8444	yes	no	cellular	17	apr	39	
10	50	management	married	tertiary	yes	72	no	no	cellular	22	aug	171	
11	58	blue-collar	married	secondary	no	1075	yes	no	cellular	18	may	42	
12	36	management	divorced	secondary	no	47	no	no	unknown	19	jun	159	
13	35	blue-collar	married	secondary	no	635	yes	no	cellular	13	may	265	
14	72	retired	married	secondary	no	17739	no	no	cellular	11	sep	135	
15	31	management	single	tertiary	no	999	yes	no	cellular	13	apr	187	
16	57	retired	married	secondary	yes	32	yes	no	cellular	15	may	476	
17	35	unemployed	married	secondary	no	315	yes	no	telephone	19	nov	70	

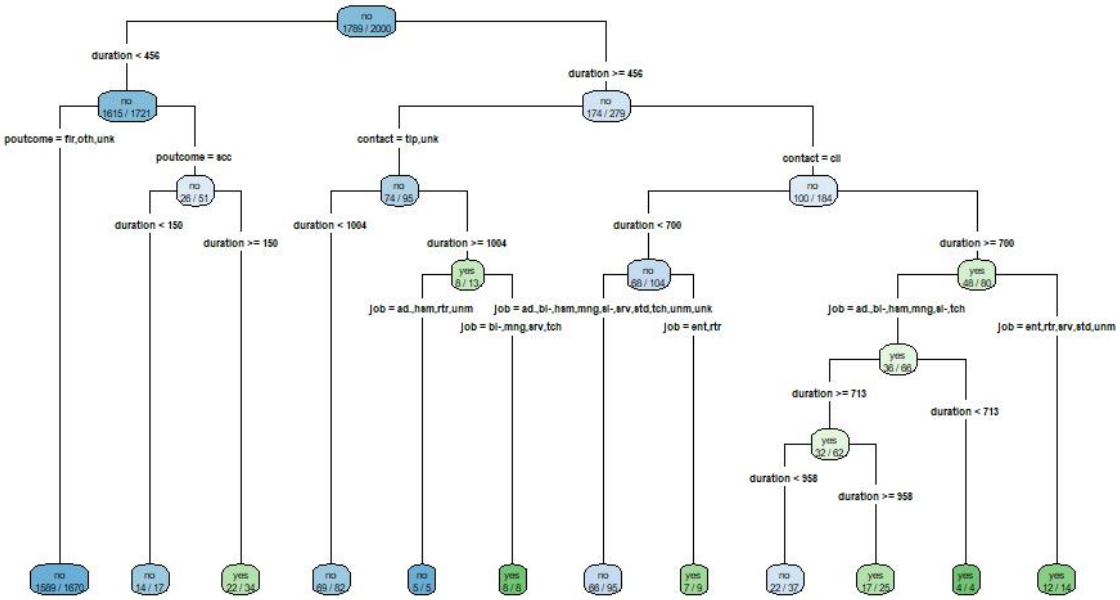
```
      housing splits as  RL,          improve=1.3488020, (0 missing)
      loan    splits as  LR,          improve=0.7450307, (0 missing)
      marital splits as  LLR,         improve=0.3260181, (0 missing)

Node number 7: 8 observations
  predicted class=yes  expected loss=0  P(node) =0.004
    class counts:     0     8
   probabilities: 0.000 1.000

Node number 12: 45 observations
  predicted class=no   expected loss=0.4222222  P(node) =0.0225
    class counts:    26    19
   probabilities: 0.578 0.422

Node number 13: 5 observations
  predicted class=yes  expected loss=0  P(node) =0.0025
    class counts:     0     5
   probabilities: 0.000 1.000

> # Plot the tree
> rpart.plot(fit, type=4, extra=2, clip.right.labs=FALSE, varlen=0, faclen=3)
>
```



2- # include a numeric variable "duration" into the model

```
fit <- rpart(subscribed ~ job + marital + education + default + housing + loan + contact + duration +
poutcome,
        method="class",
        data=banktrain,
        control=rpart.control(minsplit=1),
        parms=list(split='information'))
summary(fit)
# Plot the tree
rpart.plot(fit, type=4, extra=2, clip.right.labs=FALSE, varlen=0, faclen=3)
# Predict
newdata <- data.frame(job="retired",
                      marital="married",
                      education="secondary",
                      default="no",
                      housing="yes",
                      loan="no",
                      contact = "cellular",
                      duration = 598,
                      poutcome="unknown")
newdata
predict(fit,newdata=newdata,type=c("class"))
```

```
> # Plot the tree
> rpart.plot(fit, type=4, extra=2, clip.right.labs=FALSE, varlen=0, faclen=3)
>
> # Predict
> newdata <- data.frame(job="retired",
+                       marital="married",
+                       education="secondary",
+                       default="no",
+                       housing="yes",
+                       loan="no",
+                       contact = "cellular",
+                       duration = 598,
+                       poutcome="unknown")
> newdata
      job marital education default housing loan  contact duration poutcome
1 retired married secondary      no     yes   no cellular      598  unknown
> predict(fit,newdata=newdata,type=c("class"))
  1
yes
Levels: no yes
>
```



CONCLUSION

The Practical to Create a decision tree for the Bank-Sample dataset was successfully executed. The outcome for job="retired", marital="married", education="secondary", default="no", housing="yes", loan="no", contact = "cellular", duration = 598, poutcome="unknown" was successfully predicted.

PRACTICAL NO 6

AIM - Create a decision tree for the DTdata dataset. Predict the outcome for following Outlook="rainy", Temperature="mild", Humidity="high", Wind=FALSE

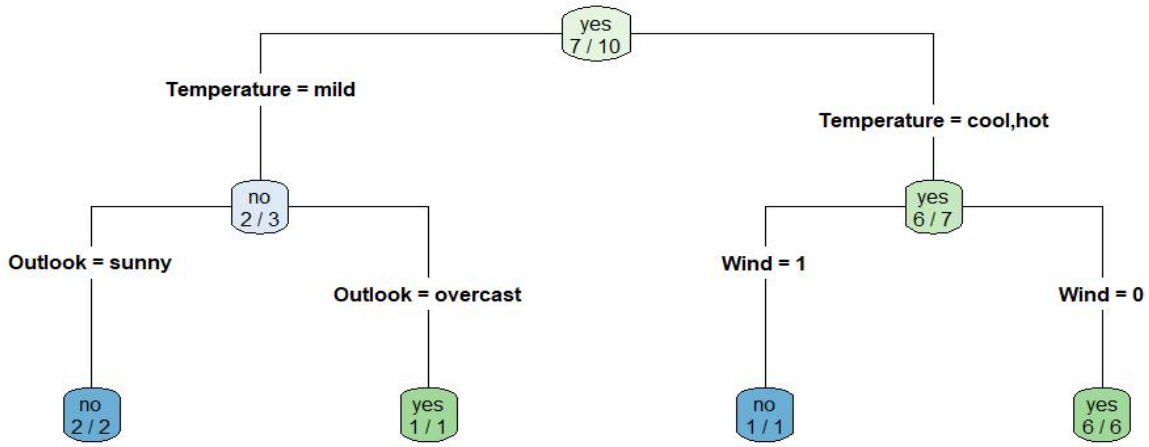
STEPS:

1- #Decision Trees in R

```
library("rpart") # load libraries

library("rpart.plot")

# Read the data

play_decision <- read.csv("C:/Users/PRIYANKA/Desktop/Sem 2/BDA Practicals/DTdata.csv",
header=TRUE, sep=",")

play_decision


summary(play_decision)

# build the decision tree

fit <- rpart(Play ~ Outlook + Temperature + Humidity + Wind,

        method="class",

        data=play_decision,

        control=rpart.control(minsplit=1),

        parms=list(split='information'))

summary(fit)


?rpart.plot

rpart.plot(fit, type=4, extra=1)


rpart.plot(fit, type=4, extra=2, clip.right.labs=FALSE,

      varlen=0, faclen=0)


newdata <- data.frame(Outlook="rainy", Temperature="mild",

            Humidity="high", Wind=FALSE)

newdata


predict(fit,newdata=newdata,type="prob")
```

predict(fit,newdata=newdata,type="class")

```
> play_decision
   Play  Outlook Temperature Humidity  Wind
1   yes    rainy        cool   normal FALSE
2    no    rainy        cool   normal  TRUE
3   yes overcast         hot     high FALSE
4    no    sunny        mild     high FALSE
5   yes    rainy        cool   normal FALSE
6   yes    sunny        cool   normal FALSE
7   yes    rainy        cool   normal FALSE
8   yes    sunny         hot   normal FALSE
9   yes overcast        mild     high  TRUE
10   no    sunny        mild     high  TRUE
> summary(play_decision)
    Play              Outlook           Temperature          Humidity            Wind
 Length:10          Length:10          Length:10          Length:10          Mode :logical
 Class :character   Class :character   Class :character   Class :character   FALSE:7
 Mode  :character   Mode  :character   Mode  :character   Mode  :character   TRUE :3
> # build the decision tree
> fit <- rpart(Play ~ Outlook + Temperature + Humidity + Wind,
+              method="class",
+              data=play_decision,
+              control=rpart.control(minsplit=1),
+              parms=list(split='information'))

    class counts:     0     6
   probabilities: 0.000 1.000

> ?rpart.plot
> rpart.plot(fit, type=4, extra=1)
> rpart.plot(fit, type=4, extra=2, clip.right.labs=FALSE,
+            varlen=0, faclen=0)
>
> newdata <- data.frame(Outlook="rainy", Temperature="mild",
+                       Humidity="high", Wind=FALSE)
> newdata
  Outlook Temperature Humidity  Wind
1   rainy        mild     high FALSE
>
> predict(fit,newdata=newdata,type="prob")
  no yes
1  1   0
> predict(fit,newdata=newdata,type="class")
 1
no
Levels: no yes
>
```



CONCLUSION

The Practical to Create a decision tree for the DTdata dataset was successfully executed. The outcome for Outlook="rainy", Temperature="mild", Humidity="high", Wind=FALSE was successfully Predicted.

PRACTICAL NO 7

AIM: Perform Simple Linear Regression on the INCOME dataset. Predict the income of a person with Age = 35, Education = 17 with confidence level of 95%

STEPS:

1-# Simple Linear Regression

##

income_input = as.data.frame(read.csv("C:/Users/PRIYANKA/Desktop/Sem 2/BDA Practicals/income.csv"))

View(income_input)

income_input[1:10,]

summary(income_input)

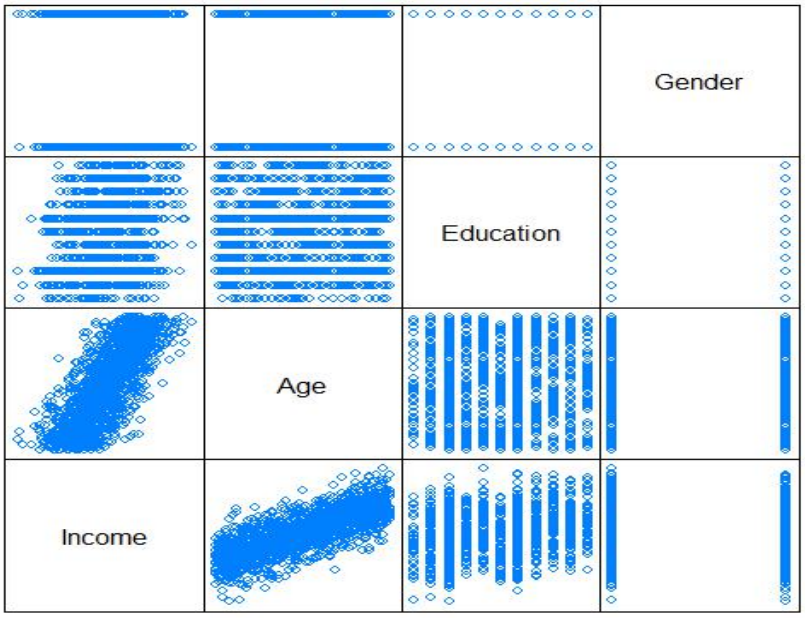
library(lattice)

splom(~income_input[c(2:5)], groups=NULL, data=income_input,

axis.line.tck = 0,

axis.text.alpha = 0)

```
> View(income_input)
> income_input[1:10,]
   ID Income Age Education Gender
1   1    113  69        12      1
2   2     91  52        18      0
3   3    121  65        14      0
4   4     81  58        12      0
5   5     68  31        16      1
6   6     92  51        15      1
7   7     75  53        15      0
8   8     76  56        13      0
9   9     56  42        15      1
10 10     53  33        11      1
>
> summary(income_input)
       ID            Income           Age          Education        Gender
 Min.   :   1.0   Min.   : 14.00   Min.   :18.00   Min.   :10.00   Min.   :0.00
 1st Qu.: 375.8   1st Qu.: 62.00   1st Qu.:30.00   1st Qu.:12.00   1st Qu.:0.00
 Median : 750.5   Median : 76.00   Median :44.00   Median :15.00   Median :0.00
 Mean   : 750.5   Mean   : 75.99   Mean   :43.58   Mean   :14.68   Mean   :0.49
 3rd Qu.:1125.2   3rd Qu.: 91.00   3rd Qu.:57.00   3rd Qu.:16.00   3rd Qu.:1.00
 Max.   :1500.0   Max.   :134.00   Max.   :70.00   Max.   :20.00   Max.   :1.00
>
```



Scatter Plot Matrix

2-pairs(income_input[c(2:5)],)

results <- lm(Income~Age + Education + Gender, income_input)

summary(results)

results2 <- lm(Income ~ Age + Education, income_input)

summary(results2)

##

compute confidence intevals for the model parameters

confint(results2, level = .95)

compute a confidence interval on the expected income of a person

Age <- 41

Education <- 12

new_pt <- data.frame(Age, Education)

new_pt

conf_int_pt <- predict(results2, new_pt, level=.95, interval="confidence")

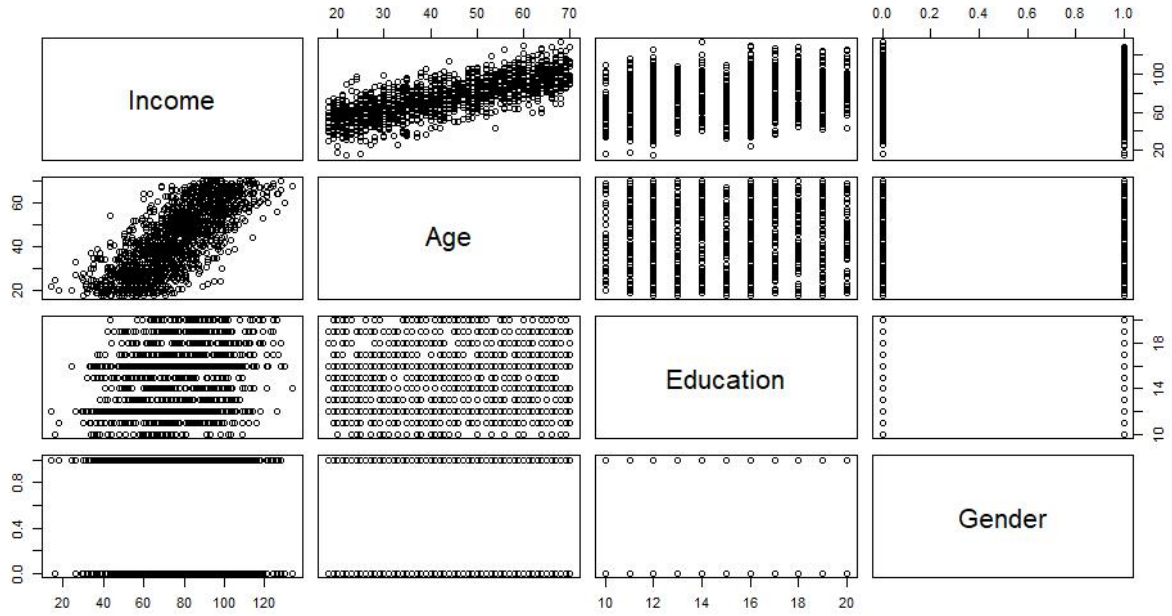
conf_int_pt

compute a prediction interval on the income of the same person

pred_int_pt <- predict(results2, new_pt, level=.95, interval="prediction")

pred_int_pt

```
(Intercept) 2.9777598 10.538690
Age         0.9556771  1.036392
Education   1.5313393  1.985862
>
> # compute a confidence interval on the expected income of a person
> Age <- 41
> Education <- 12
> new_pt <- data.frame(Age, Education)
> new_pt
  Age Education
1  41        12
> conf_int_pt <- predict(results2, new_pt, level=.95, interval="confidence")
> conf_int_pt
       fit      lwr      upr
1 68.69884 67.83102 69.56667
>
> # compute a prediction interval on the income of the same person
> pred_int_pt <- predict(results2, new_pt, level=.95, interval="prediction")
> pred_int_pt
       fit      lwr      upr
1 68.69884 44.98867 92.40902
>
```



CONCLUSION

The Practical to Perform Simple Linear Regression on the INCOME dataset was successfully executed. The income of a person with Age = 35, Education = 17 with confidence level of 95% was also Predicted successfully.

PRACTICAL NO 8

AIM: Perform Logistic Regression on the CHURN dataset. Plot the ROC curve and display the AUC

STEPS:

1- **#** Logistic Regression

```
######################################################################
```

churn_input = as.data.frame(read.csv("C:/Users/PRIYANKA/Desktop/Sem 2/BDA Practicals/churn.csv"))

View(churn_input)

head(churn_input)

summary(churn_input)

sum(churn_input$Churned)

```
View(churn_input)
head(churn_input)
 ID Churned Age Married Cust_years Churned_contacts
  1       0  61       1          3                1
  2       0  50       1          3                2
  3       0  47       1          2                0
  4       0  50       1          3                3
  5       0  29       1          1                3
  6       0  43       1          4                3
summary(churn_input)
       ID          Churned           Age           Married         Cust_years     Churned_contacts
Min.   :   1   Min.   :0.0000   Min.   :18.00   Min.   :0.0000   Min.   : 1.000   Min.   :0.000
1st Qu.:2001   1st Qu.:0.0000   1st Qu.:30.00   1st Qu.:0.0000   1st Qu.: 2.000   1st Qu.:1.000
Median :4000   Median :0.0000   Median :41.00   Median :1.0000   Median : 3.000   Median :2.000
Mean   :4000   Mean   :0.2179   Mean   :41.41   Mean   :0.5004   Mean   : 3.164   Mean   :1.718
3rd Qu.:6000   3rd Qu.:0.0000   3rd Qu.:53.00   3rd Qu.:1.0000   3rd Qu.: 4.000   3rd Qu.:2.000
Max.   :8000   Max.   :1.0000   Max.   :65.00   Max.   :1.0000   Max.   :10.000   Max.   :6.000
sum(churn_input$Churned)
1] 1743
```

```
Churn_logistic1 <- glm (Churned~Age + Married + Cust_years + Churned_contacts,
                        data=churn_input, family=binomial(link="logit"))
summary(Churn_logistic1)

Churn_logistic2 <- glm (Churned~Age + Married +  Churned_contacts,
                        data=churn_input, family=binomial(link="logit"))
summary(Churn_logistic2)

Churn_logistic3 <- glm (Churned~Age + Churned_contacts,
                        data=churn_input, family=binomial(link="logit"))
```

summary(Churn_logistic3)

summary(Churn_logistic2)

pchisq(.9 , 1, lower=FALSE)

```
-2.4476  -0.5178  -0.1972  -0.0723   3.3776

Coefficients:
                Estimate Std. Error z value Pr(>|z|)
(Intercept)     3.472062   0.132107  26.282   <2e-16 ***
Age            -0.156635   0.004088 -38.318   <2e-16 ***
Married         0.066430   0.068299   0.973    0.331
Churned_contacts 0.381909   0.027302  13.988   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 8387.3  on 7999  degrees of freedom
Residual deviance: 5358.3  on 7996  degrees of freedom
AIC: 5366.3

Number of Fisher Scoring iterations: 6

> pchisq(.9 , 1, lower=FALSE)
[1] 0.3427817
```

Receiver Operating Characteristic (ROC) Curve

install.packages("ROCR") #install, if necessary

library(ROCR)

pred = predict(Churn_logistic3, type="response")

predObj = prediction(pred, churn_input$Churned)

predObj

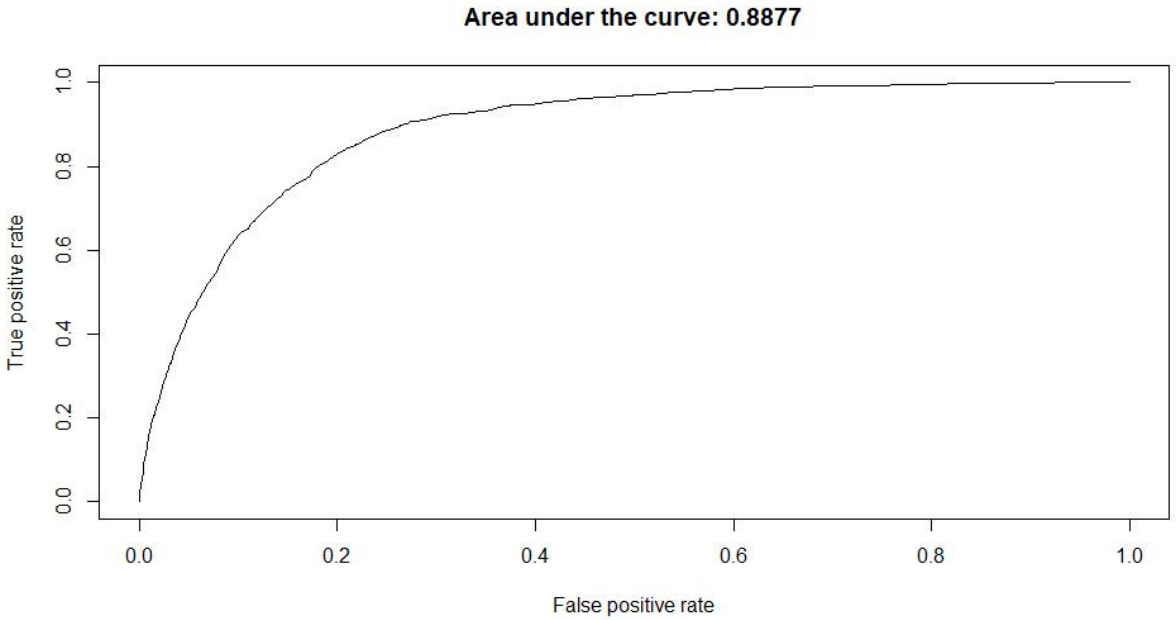
rocObj = performance(predObj, measure="tpr", x.measure="fpr")

aucObj = performance(predObj, measure="auc")

summary(aucObj)

plot(rocObj, main = paste("Area under the curve:", round(aucObj@y.values[[1]] ,4)))

```
> library(ROCR)
> pred = predict(Churn_logistic3, type="response")
> predObj = prediction(pred, churn_input$Churned )
> predObj
A prediction instance
  with 8000 data points
> rocObj = performance(predObj, measure="tpr", x.measure="fpr")
> aucObj = performance(predObj, measure="auc")
> summary(aucObj)
     Length       Class        Mode
          1 performance          S4
> plot(rocObj, main = paste("Area under the curve:", round(aucObj@y.values[[1]] ,4)))
>
```



CONCLUSION

The Practical to Perform Logistic Regression on the CHURN dataset was successfully executed.

PRACTICAL NO 9

AIM - Install Hadoop in Stand-Alone Mode on Ubuntu

STEPS

1 - Installing Java

To get started, we'll update our package list:

sudo apt update

Next, we'll install OpenJDK, the default Java Development Kit on Ubuntu 18.04:

sudo apt install default-jdk

Once the installation is complete, let's check the version.

java -version

This output verifies that OpenJDK has been successfully installed.

```
hadoop@ubuntu:~$ java -version
openjdk version "11.0.11" 2021-04-20
OpenJDK Runtime Environment (build 11.0.11+9-Ubuntu-0ubuntu2.20.04)
OpenJDK 64-Bit Server VM (build 11.0.11+9-Ubuntu-0ubuntu2.20.04, mixed mode, sha
ring)
hadoop@ubuntu:~$
```

Create a new user- hadoop

sudo adduser hadoop

```
udit@ubuntu:~$ sudo adduser hadoop
Adding user `hadoop' ...
Adding new group `hadoop' (1001) ...
Adding new user `hadoop' (1001) with group `hadoop' ...
Creating home directory `/home/hadoop' ...
Copying files from `/etc/skel' ...
New password:
Retype new password:
passwd: password updated successfully
Changing the user information for hadoop
Enter the new value, or press ENTER for the default
        Full Name []: h
        Room Number []: 1
        Work Phone []: 1
        Home Phone []: 1
        Other []: 1
Is the information correct? [Y/n] Y
```

Give all priviliges to hadoop

sudo nano /etc/sudoers

```
udit@ubuntu:~$ sudo nano /etc/sudoers
```

Add the line

user_name ALL=(ALL:ALL) ALL

```
# User privilege specification
root    ALL=(ALL:ALL) ALL
hadoop  ALL=(ALL:ALL) ALL


# Members of the admin group may gain root privileges
%admin ALL=(ALL) ALL

# Allow members of group sudo to execute any command
%sudo   ALL=(ALL:ALL) ALL

# See sudoers(5) for more information on "#include" directives:

#includedir /etc/sudoers.d

File Name to Write: /etc/sudoers
^G Get Help          M-D DOS Format      M-A Append          M-B Backup File
^C Cancel            M-M Mac Format      M-P Prepend         ^T To Files
```

switch user to hadoop

su - hadoop

The prompt should look like this - **hadoop15@ubuntu:/**

```
udit@ubuntu:~$ su - hadoop
Password:
hadoop@ubuntu:~$
```

sudo apt update

```
hadoop@ubuntu:~$ sudo apt update
[sudo] password for hadoop:
Get:1 http://security.ubuntu.com/ubuntu focal-security InRelease [114 kB]
Hit:2 http://us.archive.ubuntu.com/ubuntu focal InRelease
Get:3 http://us.archive.ubuntu.com/ubuntu focal-updates InRelease [114 kB]
Get:4 http://us.archive.ubuntu.com/ubuntu focal-backports InRelease [108 kB]
Fetched 336 kB in 3s (125 kB/s)
Reading package lists... Done
Building dependency tree
Reading state information... Done
381 packages can be upgraded. Run 'apt list --upgradable' to see them.
hadoop@ubuntu:~$
```

It is also required to set up key-based ssh

ssh-keygen -t rsa -P '' -f ~/.ssh/id_rsa

```
hadoop@ubuntu:~$ ssh-keygen -t rsa -P '' -f ~/.ssh/id_rsa
Generating public/private rsa key pair.
Created directory '/home/hadoop/.ssh'.
Your identification has been saved in /home/hadoop/.ssh/id_rsa
Your public key has been saved in /home/hadoop/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:UV4dFDl7Texdh4h5M5az+I2xA76hRKb2syG5NVOM9Eo hadoop@ubuntu
The key's randomart image is:
+---[RSA 3072]----+
|          .o.=+* |
|         oo.O = =|
|        o .+ = ==|
|       .o=o o . =|
|       +E.+o = . |
|      oo.oo = .  |
|     .oo*. o .   |
|       +++.      |
|      . .o       |
+----[SHA256]-----+
hadoop@ubuntu:~$
```

cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys

chmod 0600 ~/.ssh/authorized_keys

```
hadoop@ubuntu:~$ cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
hadoop@ubuntu:~$ chmod 0600 ~/.ssh/authorized_keys
```

verify key based login

ssh localhost

```
hadoop@ubuntu:~$ ssh localhost
The authenticity of host 'localhost (127.0.0.1)' can't be established.
ECDSA key fingerprint is SHA256:8yW0+DBp9qRq5VYzD5SGEVF339b+8fE5KYVmprzHhXA.
Are you sure you want to continue connecting (yes/no/[fingerprint])? y
Please type 'yes', 'no' or the fingerprint: yes
Warning: Permanently added 'localhost' (ECDSA) to the list of known hosts.
```

2 - Installing Hadoop

Visit the Apache Hadoop Releases page to find the most recent stable release.

https://hadoop.apache.org/release/3.2.2.html

we'll install Hadoop 3.2.2.

There are 2 options to download and install hadoop

1. Download and install using one single command

wget https://hadoop.apache.org/release/3.2.2.html/hadoop-3.2.2.tar.gz

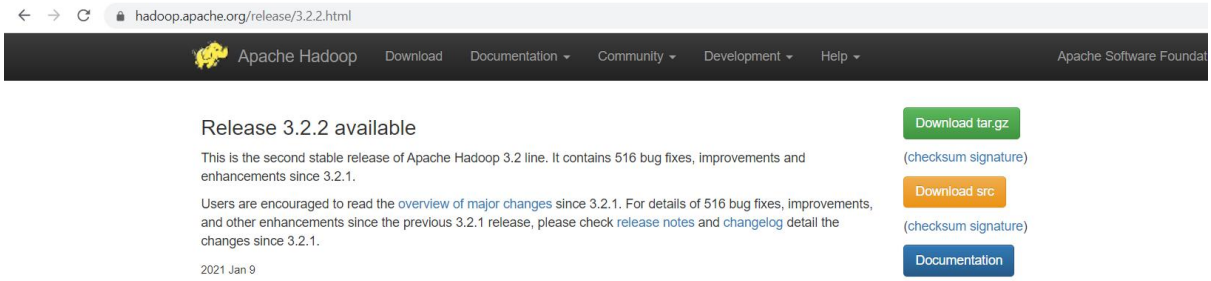
This will download the mentioned version and then install it on your system

2. If the above does not work then go to the webpage of Apache Hadoop, download it manually and then install it

visit website

https://hadoop.apache.org/release/3.2.2.html

Click button that says **"Download tar.gz"** (download the latest version)



copy the existing setup to /home/hadoop/

sudo mv /home/udit/Desktop/Prac/setups/hadoop-3.2.0.tar.gz /home/Hadoop

```
hadoop@ubuntu:~$ sudo mv /home/udit/Downloads/hadoop-3.2.0.tar.gz /home/hadoop
[sudo] password for hadoop:
hadoop@ubuntu:~$
```

Extract hadoop using the following command

we'll use the tar command with the -x flag to extract, -z to uncompressed, -v for verbose output, and -f to specify that we're extracting from a file.

tar xzvf hadoop-3.2.0.tar.gz

```
hadoop@ubuntu:~$ tar xzvf hadoop-3.2.0.tar.gz
```

```
hadoop-3.2.0/share/hadoop/mapreduce/sources/hadoop-mapreduce-client-core-3.2.0-sources.jar
hadoop-3.2.0/share/hadoop/mapreduce/sources/hadoop-mapreduce-client-core-3.2.0-test-sources.jar
hadoop-3.2.0/share/hadoop/mapreduce/sources/hadoop-mapreduce-client-hs-3.2.0-sources.jar
hadoop-3.2.0/share/hadoop/mapreduce/sources/hadoop-mapreduce-client-hs-3.2.0-test-sources.jar
hadoop-3.2.0/share/hadoop/mapreduce/sources/hadoop-mapreduce-client-hs-plugins-3.2.0-sources.jar
hadoop-3.2.0/share/hadoop/mapreduce/sources/hadoop-mapreduce-client-hs-plugins-3.2.0-test-sources.
jar
hadoop-3.2.0/share/hadoop/mapreduce/sources/hadoop-mapreduce-client-jobclient-3.2.0-sources.jar
hadoop-3.2.0/share/hadoop/mapreduce/sources/hadoop-mapreduce-client-jobclient-3.2.0-test-sources.j
ar
hadoop-3.2.0/share/hadoop/mapreduce/sources/hadoop-mapreduce-client-nativetask-3.2.0-sources.jar
hadoop-3.2.0/share/hadoop/mapreduce/sources/hadoop-mapreduce-client-nativetask-3.2.0-test-sources.
jar
hadoop-3.2.0/share/hadoop/mapreduce/sources/hadoop-mapreduce-client-shuffle-3.2.0-sources.jar
hadoop-3.2.0/share/hadoop/mapreduce/sources/hadoop-mapreduce-client-shuffle-3.2.0-test-sources.jar
hadoop-3.2.0/share/hadoop/mapreduce/sources/hadoop-mapreduce-examples-3.2.0-sources.jar
hadoop-3.2.0/share/hadoop/mapreduce/sources/hadoop-mapreduce-examples-3.2.0-test-sources.jar
hadoop-3.2.0/share/hadoop/client/
```

Set path for Java & Hadoop

sudo nano .bashrc

```
hadoop@ubuntu:~$ sudo nano .bashrc
```

export JAVA_HOME=/lib/jvm/java-11-openjdk-amd64

export PATH=$PATH:$JAVA_HOME/bin

export HADOOP_HOME=/home/hadoop/hadoop-3.2.0

export HADOOP_COMMON_HOME=/home/hadoop/hadoop-3.2.0

export HADOOP_MAPRED_HOME=/home/hadoop/hadoop-3.2.0

export PATH=$PATH:$HADOOP_COMMON_HOME/bin

export PATH=$PATH:$HADOOP_COMMON_HOME/sbin

```
export JAVA_HOME=/lib/jvm/java-11-openjdk-amd64
export PATH=$PATH:$JAVA_HOME/bin
export HADOOP_HOME=/home/hadoop/hadoop-3.2.0
export HADOOP_COMMON_HOME=/home/hadoop/hadoop-3.2.0
export HADOOP_MAPRED_HOME=/home/hadoop/hadoop-3.2.0
export PATH=$PATH:$HADOOP_COMMON_HOME/bin
export PATH=$PATH:$HADOOP_COMMON_HOME/sbin
Save modified buffer?
 Y Yes
 N No           ^C Cancel
```

After doing all changes and saving the file run the following command to make changes through the .bashrc file.

source ~/.bashrc

```
hadoop@ubuntu:~$ source ~/.bashrc
hadoop@ubuntu:~$ cat .bashr
```

Checking java and Hadoop

Command: java -version

Command: Hadoop version

```
hadoop@ubuntu:~$ java -version
openjdk version "11.0.11" 2021-04-20
OpenJDK Runtime Environment (build 11.0.11+9-Ubuntu-0ubuntu2.20.04)
OpenJDK 64-Bit Server VM (build 11.0.11+9-Ubuntu-0ubuntu2.20.04, mixed mode, sharing)
hadoop@ubuntu:~$ hadoop version
Hadoop 3.2.0
Source code repository https://github.com/apache/hadoop.git -r e97acb3bd8f3befd27418996fa5d4b50bf2
e17bf
Compiled by sunilg on 2019-01-08T06:08Z
Compiled with protoc 2.5.0
From source with checksum d3f0795ed0d9dc378e2c785d3668f39
This command was run using /home/hadoop/hadoop-3.2.0/share/hadoop/common/hadoop-common-3.2.0.jar
```

3 -Configuring Hadoop's Java Home

Hadoop requires that you set the path to Java, either as an environment variable or in the Hadoop configuration file.

To Configure Hadoop's Java Home, begin by opening Hadoop-env.sh

Sudo nano $HADOOP_HOME/etc/Hadoop/Hadoop-env.sh

```
hadoop@ubuntu:~$ sudo nano \$HADOOP_HOME/etc/hadoop/hadoop-env.sh
```

Add the following line at the end of .sh file

export JAVA_HOME=/usr/lib/JVM/java-11-OpenJDK-amd64/

to save the changes in the file, press Ctrl and x together.

then press **Y** then press Enter key

```
#
# To prevent accidents, shell commands be (superficially) locked
# to only allow certain users to execute certain subcommands.
# It uses the format of (command)_(subcommand)_USER.
#
# For example, to limit who can execute the namenode command,
# export HDFS_NAMENODE_USER=hdfs
export JAVA_HOME=/usr/lib/jvm/java-11-openjdk-amd64/
Save modified buffer?
 Y Yes
 N No           ^C Cancel
```

4- Running Hadoop
Now we should be able to run Hadoop:

$HADOOP_HOME/bin/Hadoop

```
hadoop@ubuntu:~/hadoop-3.2.0/bin$ hadoop
Usage: hadoop [OPTIONS] SUBCOMMAND [SUBCOMMAND OPTIONS]
 or    hadoop [OPTIONS] CLASSNAME [CLASSNAME OPTIONS]
  where CLASSNAME is a user-provided Java class

  OPTIONS is none or any of:

buildpaths                       attempt to add class files from build tree
--config dir                     Hadoop config directory
--debug                          turn on shell script debug mode
--help                           usage information
hostnames list[,of,host,names]   hosts to use in slave mode
hosts filename                   list of hosts to use in slave mode
loglevel level                   set the log4j level for this command
workers                          turn on worker mode

  SUBCOMMAND is one of:


    Admin Commands:
```

The help means we've successfully configured Hadoop to run in stand-alone mode.

Next we will create a directory called input in our home directory and copy Hadoop's configuration files into it to use those files as our data**.**

mkdir ~/input

cp $HADOOP_HOME/etc/hadoop/*.xml ~/input

```
hadoop@ubuntu:~/hadoop-3.2.0/bin$ mkdir ~/input
hadoop@ubuntu:~/hadoop-3.2.0/bin$ cp $HADOOP_HOME/etc/hadoop/*.xml ~/input
```

Next, we can use the following command to run the MapReduce hadoop-mapreduce-examples program, a Java archive with several options.
We'll invoke its grep program, one of the many examples included in hadoop-mapreduce-examples, followed by the input directory, input and the output directory grep_example. The MapReduce grep program will count the matches of a literal word or regular expression. Finally, we'll supply the regular expression allowed[.]* to find occurrences of the word allowed within or at the end of a declarative sentence.
The expression is case-sensitive, so we wouldn't find the word if it were capitalized at the beginning of a sentence:
$HADOOP_HOME/bin/hadoop jar $HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.2.0.jar grep ~/input ~/grep_example 'allowed[.]*'

```
hadoop@ubuntu:~/hadoop-3.2.0/bin$ $HADOOP_HOME/bin/hadoop jar $HADOOP_HOME/share/hadoop/mapreduce/
hadoop-mapreduce-examples-3.2.0.jar grep ~/input ~/grep_example 'allowed[.]*'
2022-05-15 00:24:51,362 INFO impl.MetricsConfig: Loaded properties from hadoop-metrics2.properties
2022-05-15 00:24:51,640 INFO impl.MetricsSystemImpl: Scheduled Metric snapshot period at 10 second
(s).
2022-05-15 00:24:51,640 INFO impl.MetricsSystemImpl: JobTracker metrics system started
```

When the task completes, it provides a summary of what has been processed and errors it has encountered, but this doesn't contain the actual results.

Results are stored in the output directory and can be checked by running cat on the output directory:

cat ~/grep_example/*

```
hadoop@ubuntu:~/hadoop-3.2.0/bin$ cat ~/grep_example/*
21      allowed.
1       allowed
```

CONCLUSION

The Practical to Install Hadoop in Stand-Alone Mode on Ubuntu was successfully executed.

PRACTICAL NO 10

AIM- Install Hadoop in Pseudo Distributed Mode on Ubuntu

STEPS

1- To get started, we'll update our package list:

sudo apt update

```
hadoop@ubuntu:~/hadoop-3.2.0/bin$ sudo apt update
[sudo] password for hadoop:
Get:1 http://security.ubuntu.com/ubuntu focal-security InRelease [114 kB]
Hit:2 http://us.archive.ubuntu.com/ubuntu focal InRelease
Get:3 http://us.archive.ubuntu.com/ubuntu focal-updates InRelease [114 kB]
Get:4 http://security.ubuntu.com/ubuntu focal-security/main amd64 DEP-11 Metadata [40.8 kB]
Get:5 http://us.archive.ubuntu.com/ubuntu focal-backports InRelease [108 kB]
Get:6 http://security.ubuntu.com/ubuntu focal-security/universe amd64 DEP-11 Metadata [66.4 kB]
Get:7 http://security.ubuntu.com/ubuntu focal-security/multiverse amd64 DEP-11 Metadata [2,464 B]
Get:8 http://us.archive.ubuntu.com/ubuntu focal-updates/main amd64 Packages [1,789 kB]
Get:9 http://us.archive.ubuntu.com/ubuntu focal-updates/main amd64 DEP-11 Metadata [278 kB]
Get:10 http://us.archive.ubuntu.com/ubuntu focal-updates/universe amd64 Packages [921 kB]
Get:11 http://us.archive.ubuntu.com/ubuntu focal-updates/universe amd64 DEP-11 Metadata [391 kB]
Get:12 http://us.archive.ubuntu.com/ubuntu focal-updates/multiverse amd64 DEP-11 Metadata [944 B]
Get:13 http://us.archive.ubuntu.com/ubuntu focal-backports/main amd64 DEP-11 Metadata [9,584 B]
Get:14 http://us.archive.ubuntu.com/ubuntu focal-backports/universe amd64 DEP-11 Metadata [30.8 kB
```

Next, we'll install OpenJDK, the default Java Development Kit on Ubuntu 18.04:

sudo apt install default-JDK

Once the installation is complete, let's check the version.

java -version

This output verifies that OpenJDK has been successfully installed.

```
hadoop@ubuntu:~/hadoop-3.2.0/bin$ java -version
openjdk version "11.0.11" 2021-04-20
OpenJDK Runtime Environment (build 11.0.11+9-Ubuntu-0ubuntu2.20.04)
OpenJDK 64-Bit Server VM (build 11.0.11+9-Ubuntu-0ubuntu2.20.04, mixed mode, sharing)
hadoop@ubuntu:~/hadoop-3.2.0/bin$
```

Create a new user- Hadoop

sudo adduser Hadoop

```
udit@ubuntu:~$ sudo adduser hadoop
Adding user `hadoop' ...
Adding new group `hadoop' (1001) ...
Adding new user `hadoop' (1001) with group `hadoop' ...
Creating home directory `/home/hadoop' ...
Copying files from `/etc/skel' ...
New password:
Retype new password:
passwd: password updated successfully
Changing the user information for hadoop
Enter the new value, or press ENTER for the default
        Full Name []: h
        Room Number []: 1
        Work Phone []: 1
        Home Phone []: 1
        Other []: 1
Is the information correct? [Y/n] Y
```

Give all priviliges to hadoop

sudo nano /etc/sudoers

```
udit@ubuntu:~$ sudo nano /etc/sudoers
```

Add the line

user_name ALL=(ALL:ALL) ALL

```
# User privilege specification
root    ALL=(ALL:ALL) ALL
hadoop  ALL=(ALL:ALL) ALL


# Members of the admin group may gain root privileges
%admin ALL=(ALL) ALL

# Allow members of group sudo to execute any command
%sudo   ALL=(ALL:ALL) ALL

# See sudoers(5) for more information on "#include" directives:

#includedir /etc/sudoers.d

File Name to Write: /etc/sudoers
^G Get Help          M-D DOS Format      M-A Append          M-B Backup File
^C Cancel            M-M Mac Format      M-P Prepend         ^T To Files
```

switch user to hadoop15

su - hadoop

The prompt shoud look like this - hadoop@ubuntu:/

```
udit@ubuntu:~$ su - hadoop
Password:
hadoop@ubuntu:~$
```

It is also required to set up key-based ssh

ssh-keygen -t rsa -P '' -f ~/.ssh/id_rsa

```
hadoop@ubuntu:~$ ssh-keygen -t rsa -P '' -f ~/.ssh/id_rsa
Generating public/private rsa key pair.
Created directory '/home/hadoop/.ssh'.
Your identification has been saved in /home/hadoop/.ssh/id_rsa
Your public key has been saved in /home/hadoop/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:UV4dFDl7Texdh4h5M5az+I2xA76hRKb2syG5NVOM9Eo hadoop@ubuntu
The key's randomart image is:
+---[RSA 3072]----+
|          .o.=+* |
|         oo.O = =|
|        o .+ = ==|
|       .o=o o . =|
|       +E.+o = . |
|      oo.oo = .  |
|     .oo*. o .   |
|       +++.      |
|      . .o       |
+----[SHA256]-----+
hadoop@ubuntu:~$
```

cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys

chmod 0600 ~/.ssh/authorized_keys

```
hadoop@ubuntu:~$ cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
hadoop@ubuntu:~$ chmod 0600 ~/.ssh/authorized_keys
```

verify key-based login

ssh localhost

```
hadoop@ubuntu:~$ ssh localhost
The authenticity of host 'localhost (127.0.0.1)' can't be established.
ECDSA key fingerprint is SHA256:8yW0+DBp9qRq5VYzD5SGEVF339b+8fE5KYVmprzHhXA.
Are you sure you want to continue connecting (yes/no/[fingerprint])? y
Please type 'yes', 'no' or the fingerprint: yes
Warning: Permanently added 'localhost' (ECDSA) to the list of known hosts.
```

2- Downloading & Installing Hadoop

Visit the Apache Hadoop Releases page to find the most recent stable release.

https://hadoop.apache.org/release/3.2.0.html

we'll install Hadoop 3.2.0.

There are 2 options to download and install Hadoop

1. Download and install using one single command

wget https://hadoop.apache.org/release/3.2.0.html/hadoop-3.2.0.tar.gz

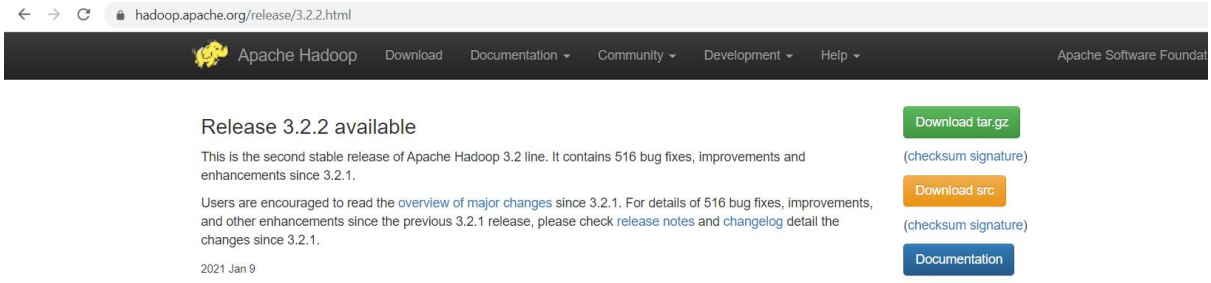
This will download the mentioned version and then install it on your system

2. If the above does not work then go to the webpage of Apache Hadoop, download it manually and then install it

visit website

https://hadoop.apache.org/release/3.2.0.html

Click button that says "Download tar.gz" (download the latest version)



copy the existing setup to /home/hadoop/

sudo mv /home/udit/Desktop/Prac/setups/hadoop-3.2.0.tar.gz /home/Hadoop

```
hadoop@ubuntu:~$ sudo mv /home/udit/Downloads/hadoop-3.2.0.tar.gz /home/hadoop
[sudo] password for hadoop:
hadoop@ubuntu:~$
```

install hadoop using the following command

we'll use the tar command with the -x flag to extract, -z to uncompress, -v for verbose output, and -f to specify that we're extracting from a file.

tar xzvf hadoop-3.2.0.tar.gz

```
hadoop@ubuntu:~$ tar xzvf hadoop-3.2.0.tar.gz
```

Finally, we'll move the extracted files into /usr/local, the appropriate place for locally installed software.

sudo mv hadoop-3.2.0 /home/Hadoop

```
hadoop-3.2.0/share/hadoop/mapreduce/sources/hadoop-mapreduce-client-core-3.2.0-sources.jar
hadoop-3.2.0/share/hadoop/mapreduce/sources/hadoop-mapreduce-client-core-3.2.0-test-sources.jar
hadoop-3.2.0/share/hadoop/mapreduce/sources/hadoop-mapreduce-client-hs-3.2.0-sources.jar
hadoop-3.2.0/share/hadoop/mapreduce/sources/hadoop-mapreduce-client-hs-3.2.0-test-sources.jar
hadoop-3.2.0/share/hadoop/mapreduce/sources/hadoop-mapreduce-client-hs-plugins-3.2.0-sources.jar
hadoop-3.2.0/share/hadoop/mapreduce/sources/hadoop-mapreduce-client-hs-plugins-3.2.0-test-sources.
jar
hadoop-3.2.0/share/hadoop/mapreduce/sources/hadoop-mapreduce-client-jobclient-3.2.0-sources.jar
hadoop-3.2.0/share/hadoop/mapreduce/sources/hadoop-mapreduce-client-jobclient-3.2.0-test-sources.j
ar
hadoop-3.2.0/share/hadoop/mapreduce/sources/hadoop-mapreduce-client-nativetask-3.2.0-sources.jar
hadoop-3.2.0/share/hadoop/mapreduce/sources/hadoop-mapreduce-client-nativetask-3.2.0-test-sources.
jar
hadoop-3.2.0/share/hadoop/mapreduce/sources/hadoop-mapreduce-client-shuffle-3.2.0-sources.jar
hadoop-3.2.0/share/hadoop/mapreduce/sources/hadoop-mapreduce-client-shuffle-3.2.0-test-sources.jar
hadoop-3.2.0/share/hadoop/mapreduce/sources/hadoop-mapreduce-examples-3.2.0-sources.jar
hadoop-3.2.0/share/hadoop/mapreduce/sources/hadoop-mapreduce-examples-3.2.0-test-sources.jar
hadoop-3.2.0/share/hadoop/client/
```

verify by navigating to /home/Hadoop

Update 6 important files

1. **.bashrc**

==========

Set path for Java & Hadoop

sudo nano .bashrc

```
hadoop@ubuntu:~$ sudo nano .bashrc
```

export JAVA_HOME=/usr/lib/jvm/java-11-openjdk-amd64

export PATH=$PATH:$JAVA_HOME/bin

export HADOOP_HOME = /home/hadoop15/hadoop-3.2.0

export HADOOP_INSTALL = $HADOOP_HOME

export HADOOP_COMMON_HOME = $HADOOP_HOME

export HADOOP_MAPRED_HOME = $HADOOP_HOME

export HADOOP_HDFS_HOME = $HADOOP_HOME

export HADOOP_YARN_HOME = $HADOOP_HOME

export HADOOP_COMMON_LIB_NATIVE_DIR = $HADOOP_HOME/lib/native

export PATH = $PATH:$HADOOP_HOME/sbin:$HADOOP_HOME/bin

```
 GNU nano 4.8                                    .bashrc
  fi
fi

export JAVA_HOME=/usr/lib/jvm/java-11-openjdk-amd64
export PATH=$PATH:$JAVA_HOME/bin
export HADOOP_HOME=/home/hadoop/hadoop-3.2.0
export HADOOP_INSTALL=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export HADOOP_YARN_HOME=$HADOOP_HOME
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
export PATH = $PATH:$HADOOP_HOME/sbin:$HADOOP_HOME/bin
```

Execute by following line

source ~/.bashrc

```
hadoop@ubuntu:~$ source ~/.bashrc
hadoop@ubuntu:~$ cat .bashr
```

Checking of java and hadoop

Command: java -version

Command: hadoop version

```
hadoop@ubuntu:~$ java -version
openjdk version "11.0.11" 2021-04-20
OpenJDK Runtime Environment (build 11.0.11+9-Ubuntu-0ubuntu2.20.04)
OpenJDK 64-Bit Server VM (build 11.0.11+9-Ubuntu-0ubuntu2.20.04, mixed mode, sharing)
hadoop@ubuntu:~$ hadoop version
Hadoop 3.2.0
Source code repository https://github.com/apache/hadoop.git -r e97acb3bd8f3befd27418996fa5d4b50bf2
e17bf
Compiled by sunilg on 2019-01-08T06:08Z
Compiled with protoc 2.5.0
From source with checksum d3f0795ed0d9dc378e2c785d3668f39
This command was run using /home/hadoop/hadoop-3.2.0/share/hadoop/common/hadoop-common-3.2.0.jar
```

==========

2 hadoop-env.sh - Configuring Hadoop's Java Home

==========

To Configure Hadoop15's Java Home, begin by opening hadoop-env.sh

sudo nano /home/hadoop/hadoop-3.2.0/etc/hadoop/hadoop-env.sh

```
hadoop@ubuntu:~$ sudo nano \$HADOOP_HOME/etc/hadoop/hadoop-env.sh
```

or use the variable $HADOOP_HOME

sudo nano $HADOOP_HOME/etc/hadoop/hadoop-env.sh

Add the following line at the end of .sh file

export JAVA_HOME=/usr/lib/jvm/java-11-openjdk-amd64

==========

3 core-site.xml

==========

sudo nano $HADOOP_HOME/etc/hadoop/core-site.xml

<property>

<name>hadoop.tmp.dir</name>

<value>/home/hadoop/tmpdata</value>

<description>A base for other temporarary directories</description>

</property>

<property>

<name>fs.default.name</name>

<value>hdfs://localhost:9000</value>

<description>The name of the default file system.</description>

</property>

```
  GNU nano 4.8          /home/hadoop/hadoop-3.2.0/etc/hadoop/core-site.xml          Modified
  distributed under the License is distributed on an "AS IS" BASIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
  See the License for the specific language governing permissions and
  limitations under the License. See accompanying LICENSE file.
-->

<!-- Put site-specific property overrides in this file. -->

<configuration>

<property>
        <name>hadoop.tmp.dir</name>
        <value>/home/hadoop/tmpdata</value>
        <description>A base for other temporarary directories</description>
</property>

<property>
        <name>fs.default.name</name>
        <value>hdfs://localhost:9000</value>
        <description>The name of the default file system.</description>
</property>
```

==========

4 hdfs-site.xml

==========

sudo nano $HADOOP_HOME/etc/hadoop/hdfs-site.xml

```
hadoop@ubuntu:~$ sudo nano $HADOOP_HOME/etc/hadoop/hdfs-site.xml
```

<property>

<name>dfs.data.dir</name>

<value>/home/hadoop/dfsdata/namenode</value>

<description>Location of namenode</description>

</property>

<property>

<name>dfs.data.dir</name>

<value>/home/hadoop/dfsdata/datanode</value>

<description>Location of datanode</description>

</property>

<property>

<name>dfs.replication</name>

<value>1</value>

<description>Replication Factor</description>

</property>

```
 GNU nano 4.8                /home/hadoop/hadoop-3.2.0/etc/hadoop/hdfs-site.xml

<!-- Put site-specific property overrides in this file. -->

<configuration>
<property>
        <name>dfs.data.dir</name>
        <value>/home/hadoop/dfsdata/namenode</value>
        <description>Location of namenode</description>
</property>

<property>
        <name>dfs.data.dir</name>
        <value>/home/hadoop/dfsdata/datanode</value>
        <description>Location of datanode</description>
</property>

<property>
        <name>dfs.replication</name>
        <value>1</value>
        <description>Replication Factor</description>
</property>
```

==========

5 mapred-site.xml

==========

sudo nano $HADOOP_HOME/etc/hadoop/mapred-site.xml

```
hadoop@ubuntu:~$ sudo nano $HADOOP_HOME/etc/hadoop/mapred-site.xml
```

<property>

<name>mapreduce.framework.name</name>

<value>yarn</value>

<description>Name of my mapreduce framework</description>

</property>

```
<configuration>
<property>
        <name>mapreduce.framework.name</name>
        <value>yarn</value>
        <description>Name of my mapreduce framework</description>
</property>
```

==========

6 yarn-site.xml

==========

sudo nano $HADOOP_HOME/etc/hadoop/yarn-site.xml

```
hadoop@ubuntu:~$ sudo nano $HADOOP_HOME/etc/hadoop/yarn-site.xml
```

<property>

<name>yarn.nodemanager.aux-services</name>

<value>mapreduce_shuffle</value>

</property>

<property>

<name>yarn.nodemanager.aux-services.mapreduce.shuffle.class</name>

<value>org.apache.hadoop.mapred.ShuffleHandler</value>

</property>

<property>

<name>yarn.resourcemanager.hostname</name>

<value>127.0.0.1</value>

</property>

<property>

<name>yarn.acl.enable</name>

<value>0</value>

</property>

<property>

<name>yarn.nodemanager.env-whitelist</name>

<value>JAVA_HOME, HADOOP_COMMON_HOME, HADOOP_HDFS, HADOOP_YARN</value>

</property>

```
 GNU nano 4.8            /home/hadoop/hadoop-3.2.0/etc/hadoop/yarn-site.xml
</property>

<property>
<name>yarn.acl.enable</name>
<value>0</value>
</property>

<property>
<name>yarn.nodemanager.env-whitelist</name>
<value>JAVA_HOME, HADOOP_COMMON_HOME, HADOOP_HDFS, HADOOP_YARN</value>
</property>

</configuration>
```

==========

Format Namenode

==========

move to the /bin folder

cd /home/hadoop/hadoop-3.2.0/bin

```
hadoop@ubuntu:~$ cd /home/hadoop/hadoop-3.2.0/bin
hadoop@ubuntu:~/hadoop-3.2.0/bin$
```

Now format the namenode using the following command

hdfs namenode –format

```
hadoop@ubuntu:~/hadoop-3.2.0/bin$ hdfs namenode -format
```

```
hadoop@ubuntu:~/hadoop-3.2.0/bin$ hdfs namenode -format
2022-05-15 01:55:09,798 INFO namenode.NameNode: STARTUP_MSG:
/************************************************************
STARTUP_MSG: Starting NameNode
STARTUP_MSG:   host = ubuntu/127.0.1.1
STARTUP_MSG:   args = [-format]
STARTUP_MSG:   version = 3.2.0
```

==========

Start the namenode, datanode

==========

move to the /sbin folder

/home/hadoop/hadoop-3.2.0/sbin

start-dfs.sh

start-yarn.sh

```
hadoop@ubuntu:~/hadoop-3.2.0/sbin$ start-dfs.sh
Starting namenodes on [localhost]
Starting datanodes
Starting secondary namenodes [ubuntu]
ubuntu: Warning: Permanently added 'ubuntu' (ECDSA) to the list of known hosts.
2022-05-15 01:59:09,340 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your
platform... using builtin-java classes where applicable
hadoop@ubuntu:~/hadoop-3.2.0/sbin$ start-yarn.sh
Starting resourcemanager
Starting nodemanagers
hadoop@ubuntu:~/hadoop-3.2.0/sbin$
```

==========

Start the task tracker and job tracker // Skip in case file not present

==========

start-mapred.sh

==========

To check if Hadoop started correctly

==========

jps

```
hadoop@ubuntu:~/hadoop-3.2.0/sbin$ jps
18520 SecondaryNameNode
19113 Jps
18187 NameNode
18319 DataNode
hadoop@ubuntu:~/hadoop-3.2.0/sbin$
```

==========

Copy file from local system to hdfs

==========

hdfs dfs -put source

```
hadoop@ubuntu:~/hadoop-3.2.0/bin$ hdfs dfs -put /home/udit/Downloads/employee_details.txt
2022-05-15 02:07:26,252 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your
platform... using builtin-java classes where applicable
put: `.': No such file or directory: `hdfs://localhost:9000/user/hadoop'
hadoop@ubuntu:~/hadoop-3.2.0/bin$ ls
container-executor  hadoop.cmd  hdfs.cmd  mapred.cmd    test-container-executor  yarn.cmd
hadoop              hdfs        mapred    oom-listener  yarn
hadoop@ubuntu:~/hadoop-3.2.0/bin$
```

check using ls

hdfs dfs -ls

```
hadoop@ubuntu:~/hadoop-3.2.0/sbin$ dfs ls

Command 'dfs' not found, did you mean:

  command 'df' from deb coreutils (8.30-3ubuntu2)
  command 'bfs' from deb bfs (1.5.2-1)
  command 'hfs' from deb hfsutils-tcltk (3.2.6-14)
  command 'dms' from deb anacrolix-dms (1.1.0-1)
  command 'zfs' from deb zfsutils-linux (0.8.3-1ubuntu12.13)
  command 'zfs' from deb zfs-fuse (0.7.0-20)
  command 'dfc' from deb dfc (3.1.1-1)
  command 'dcs' from deb drbl (2.30.5-1)
  command 'fs' from deb openafs-client (1.8.4~pre1-1ubuntu2.4)
```

==========

Check with browser

==========

DFS overview - http://localhost:9000/

Overview 'localhost:9000' (active)

Started:	Thu May 19 00:19:18 -0700 2022
Version:	3.2.0, re97acb3bd8f3befd27418996fa5d4b50bf2e17bf
Compiled:	Mon Jan 07 22:08:00 -0800 2019 by sunilg from branch-3.2.0
Cluster ID:	CID-39192ea9-c6c0-44ff-aabe-a213c7026fac
Block Pool ID:	BP-1245326821-127.0.1.1-1652944679537

Data node - **http://localhost:9866/**

DataNode on ubuntu:9866

Cluster ID:	CID-39192ea9-c6c0-44ff-aabe-a213c7026fac
Version:	3.2.0, re97acb3bd8f3befd27418996fa5d4b50bf2e17bf

Block Pools

Namenode Address	Block Pool ID	Actor State	Last Heartbeat	Last Block Report	Last Block Report Size (Max Size)
localhost:9000	BP-1245326821-127.0.1.1-1652944679537	RUNNING	1s	6 minutes	0 B (64 MB)

Volume Information

Directory	StorageType	Capacity Used	Capacity Left	Capacity Reserved	Reserved Space for Replicas	Blocks
/home/priya/dfsdata/datanode	DISK	24 KB	61.04 GB	0 B	0 B	0

Hadoop, 2019.

CONCLUSION

The Practical to Install Hadoop in Pseudo Distributed Mode on Ubuntu was successfully executed.