-:CONTENTS:-

PAGE No.

Introduction :-

Data Mining Technique can be categorised into three major components – Unsupervised learning, Supervised learning and Reinforcement. In Unsupervised learning we draw inferences from datasets consisting of input data without labelled responses. The most common unsupervised learning method is cluster analysis, which is used for exploratory data analysis to find hidden patterns or grouping in data. The clusters are modelled using a measure of similarity which is defined upon metrics such as Euclidean or probabilistic distance. Cluster Analysis is a class of techniques that are used to classify objects or cases into relative groups called clusters. Cluster analysis has been used in marketing to identify homogenous groups of buyers. This idea can be generalized in various fields of data where a data analyst wants to see the probable grouping (homogeneous) in the data before further analysis. The basic task of clustering is to grouping (or making clusters) in such a way that variation within the groups is as small as possible and variation between the groups is as high as possible.

Applications of Clustering :-

Clustering is broadly applied in many sectors such as market research, pattern recognition, data analysis, and image processing. Clustering can also help marketers discover distinct groups in their customer base. And they can characterize their customer groups based on the purchasing patterns. In the field of biology, it can be used to derive plant and animal taxonomies, categorize genes with similar functionalities and gain insight into structures inherent to populations. Clustering also helps in classifying documents on the web for information discovery. Clustering is also used in outlier detection applications such as detection of credit card fraud. Statisticians often use clustering to detect outliers.

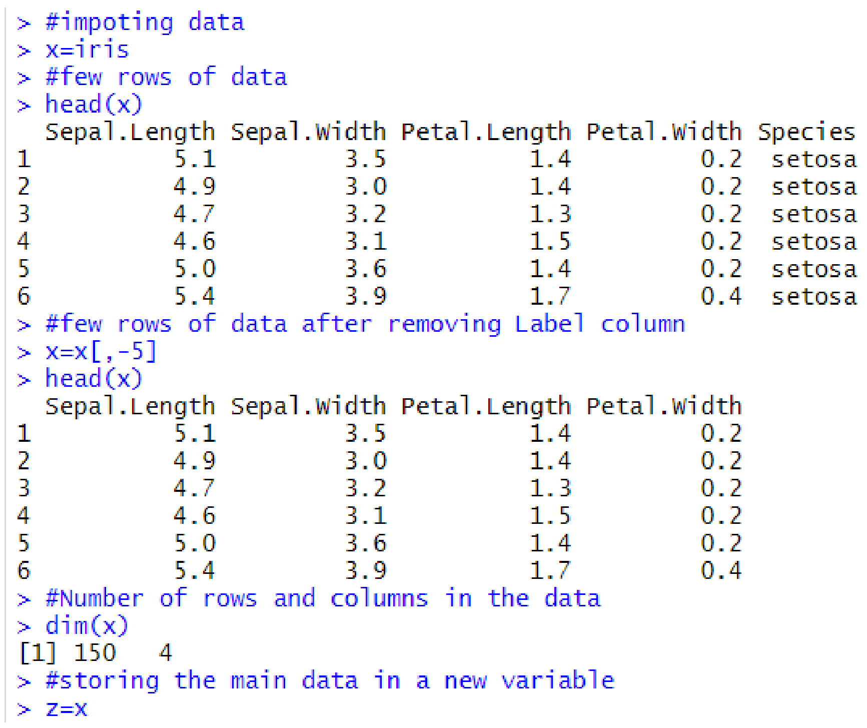
Objective of the Project :-

Here we use the famous Iris data for our analysis. This Iris data is categorized by Species – ‘setosa’, ‘versicolor’ and ‘verginica’. But for clustering the should have no labels and here label is present as Species column. So, we remove Species column and proceed for our analysis. Now the new Iris data have no information about the label. Now by using various statistical tools we will find our new clusters and then we compare these clusters with the given clusters ‘setosa’, ‘versicolor’ and ‘verginica’. To fulfil our purpose we run principal component analysis through our data. From here we will able get an idea of number of cluster of the data. Then we will do K-means Clustering and Hierarchical clustering to find final clusters.

****R-studio is used in the project for all calculation and analysis**

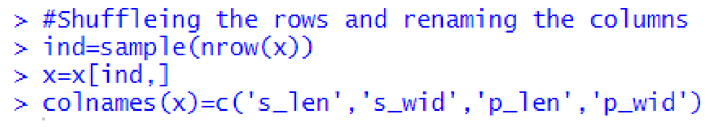
Principal Component Analysis :-

Principal Component Analysis (PCA) is a useful technique for exploratory data analysis, allowing to better visualize the variation present in a dataset with many variables. Here in the data we have 4 variables , so, we have 4 principal components namely PC1, PC2, PC3 and PC4, where these PCi's are linear combinations of main variables 'Sepal.Length', 'Sepal.width', 'Petal.Length', 'Petal.Width' for i=1,2,3,4 and information contained in PCi is more than information contained in PCj or Var(PCi) > Var(PCj) for all i> j.



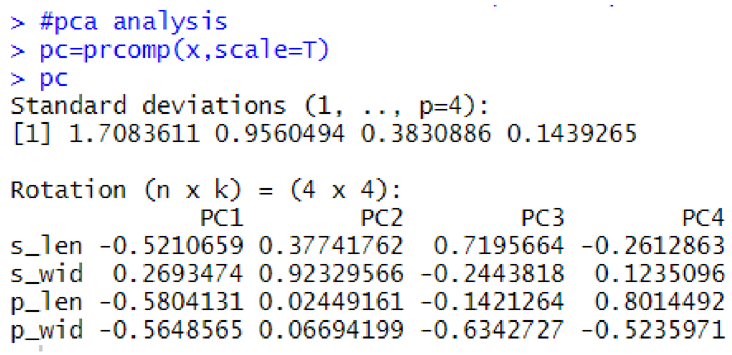
```
> #impoting data
> x=iris
> #few rows of data
> head(x)
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
4          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
> #few rows of data after removing Label column
> x=x[,-5]
> head(x)
  Sepal.Length Sepal.Width Petal.Length Petal.Width
1          5.1         3.5          1.4         0.2
2          4.9         3.0          1.4         0.2
3          4.7         3.2          1.3         0.2
4          4.6         3.1          1.5         0.2
5          5.0         3.6          1.4         0.2
6          5.4         3.9          1.7         0.4
> #Number of rows and columns in the data
> dim(x)
[1] 150   4
> #storing the main data in a new variable
> z=x
```

So, in this data we have 150 observations and 4 variables. We have removed label column but data is organized as first 50 observations as 'setosa', second 50 observations are 'versicolor' and last 50 observations are 'verginica'. So, we need to shuffle the rows. Again we rename the variables into shorter name.



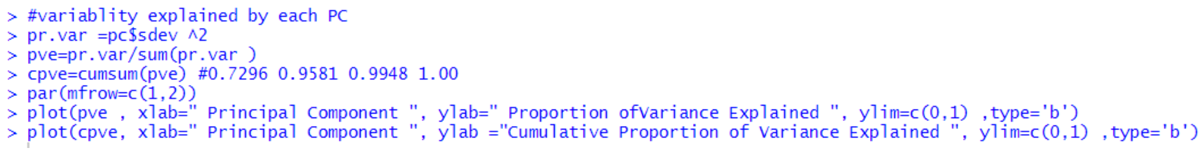
```
> #Shuffleing the rows and renaming the columns
> ind=sample(nrow(x))
> x=x[ind,]
> colnames(x)=c('s_len','s_wid','p_len','p_wid')
```

So, our data is ready and now will apply principal component analysis using prcomp() function. Here the columns of the data is standardized to project the original data onto directions which maximize the variance. Thus we will obtain the PC of standardized variables.

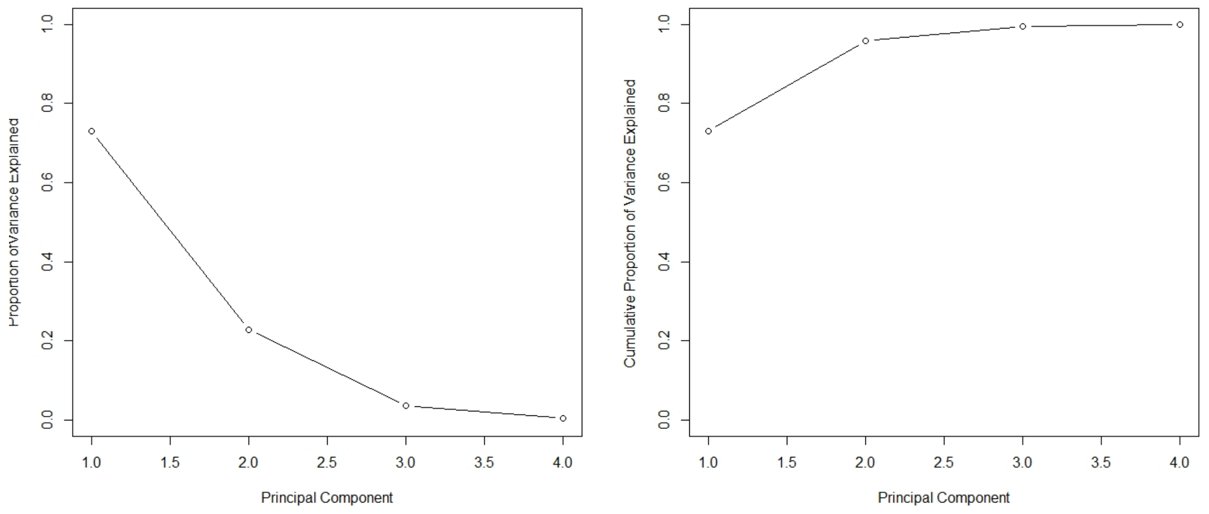


```
> #pca analysis
> pc=prcomp(x,scale=T)
> pc
Standard deviations (1, .., p=4):
[1] 1.7083611 0.9560494 0.3830886 0.1439265

Rotation (n x k) = (4 x 4):
             PC1        PC2        PC3        PC4
s_len -0.5210659 0.37741762  0.7195664 -0.2612863
s_wid  0.2693474 0.92329566 -0.2443818  0.1235096
p_len -0.5804131 0.02449161 -0.1421264  0.8014492
p_wid -0.5648565 0.06694199 -0.6342727 -0.5235971
```

Here in the above picture PCi columns are loadings of originally defined PCi for i=1,2,3,4. That means originally defined PCi= (PCi col)’ x (‘Sepal.Length’ col, ‘Sepal.Width’ col, ‘Petal.Length’ col, ‘Petal.Width’ col) where ‘col‘ means column. Now we will see how much variability is explained by each PC.

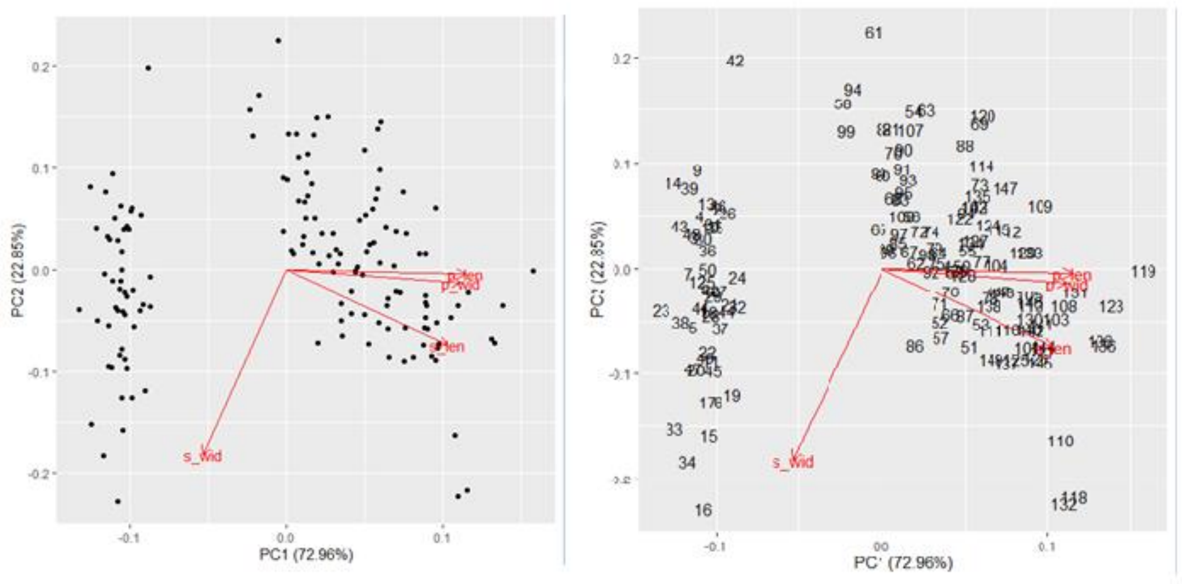


```
> #variablity explained by each PC
> pr.var =pc$sdev ^2
> pve=pr.var/sum(pr.var )
> cpve=cumsum(pve) #0.7296 0.9581 0.9948 1.00
> par(mfrow=c(1,2))
> plot(pve , xlab=" Principal Component ", ylab=" Proportion ofVariance Explained ", ylim=c(0,1) ,type='b')
> plot(cpve, xlab=" Principal Component ", ylab ="Cumulative Proportion of Variance Explained ", ylim=c(0,1) ,type='b')
```

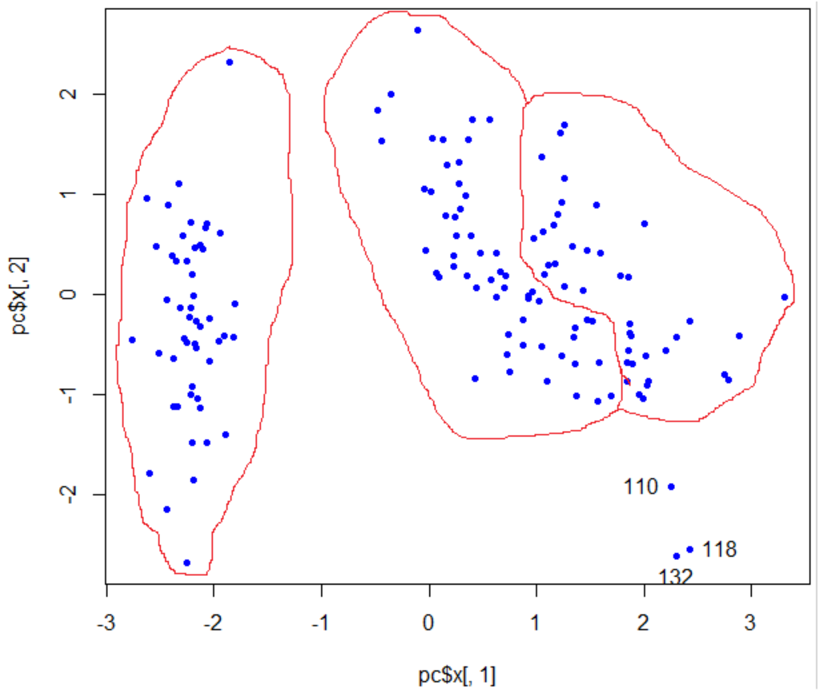


First two PC have explained more than 95% variation (which is clear from the graphs also. In the right hand panel of the graph we see that the curve become smooth onwards second point.), so, instead of using four PC we will use first two PC from now as it is explaining more than enough variation (that is reduction of variables). Now we will see the scatter plots derived from first two PC.

```
> #scatterplot with label and without label
> library(ggfortify)
> par(mfrow=c(1,2))
> autoplot(pc,loadings=T,loadings.label=T)
> autoplot(pc,loadings=T,loadings.label=T,shape=F)
```



From the above graphs roughly we see that roughly there are three clusters and three outliers with labels ‘110’, ‘132’, ‘118’.



Now we exclude these three observations and proceed our analysis.

```
> #Removing outliers
> c=c('118','132','110')
> for(i in 1:length(c))
+   x=x[-which(rownames(x)==c[i]),]
```

K-means Clustering :-

In the new data set x we have 147 observations and 4 columns and from PCA we came to know that there should be K=3 clusters. K-means clustering (McQueen 1967) is one of the most commonly used unsupervised algorithm for partitioning a given dataset into a set of k groups (i.e. k clusters). There are several k-means algorithms available. The standard algorithm is the Hartigan-Wong algorithm (Hartigan and Wong 1979) which defines the total within cluster variation as the sum of squared distances between observations within a cluster and in R this algorithm is used by default.

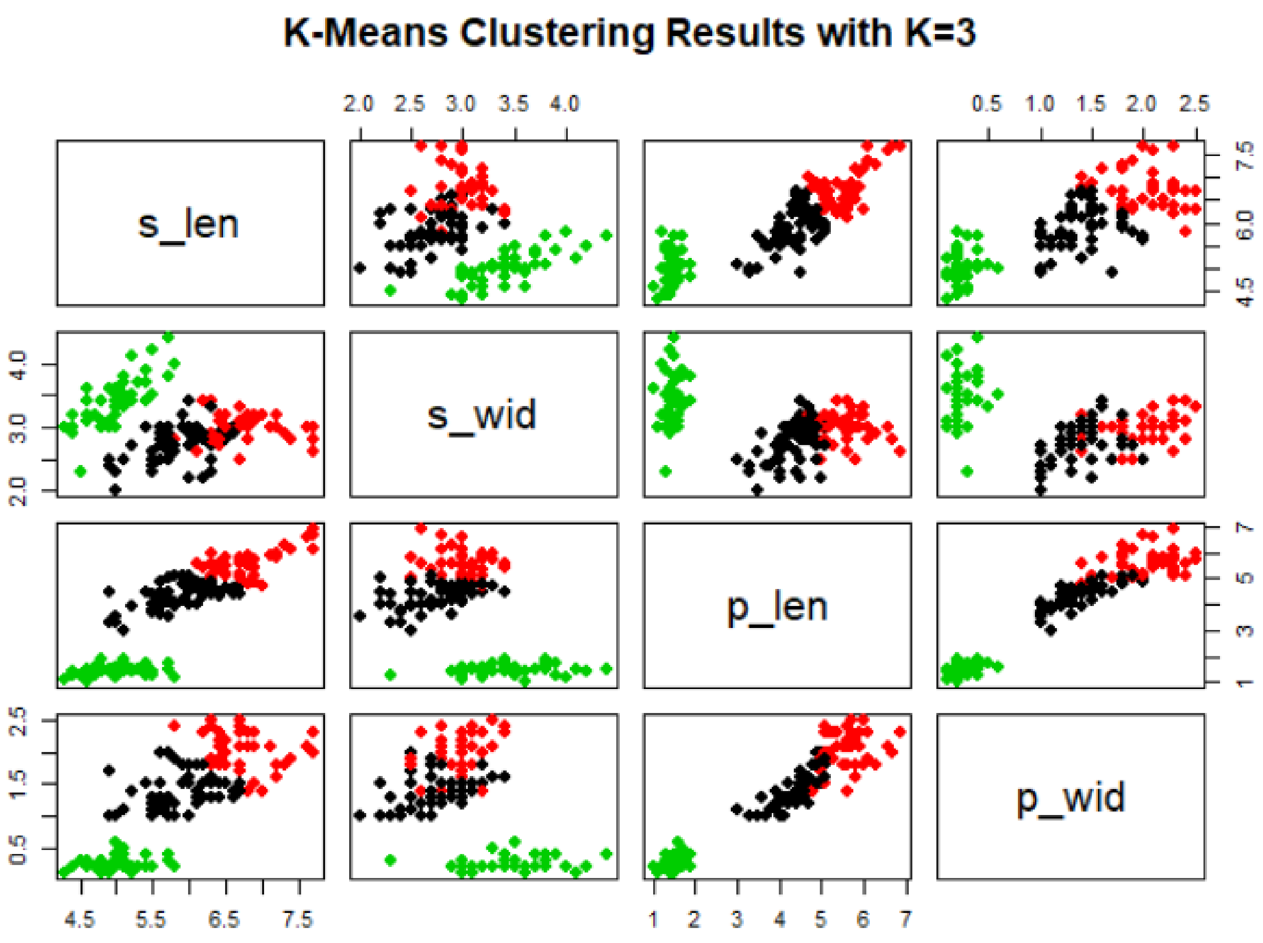
```
> ##kmeans clustering
> set.seed(1234)
> km=kmeans(x,3, nstart =50)
> km

K-means clustering with 3 clusters of sizes 37, 60, 50

Cluster means:
     s_len    s_wid    p_len    p_wid
1 6.764865 3.016216 5.643243 2.048649
2 5.885000 2.740000 4.376667 1.418333
3 5.006000 3.428000 1.462000 0.246000

Clustering vector:
 79  30  83 122  70  86  99 115 124  97  49  47   4  42  91  74  17 135  15  39  53 101 142  27 129  38  13  75 
  2   3   2   2   2   2   2   1   2   2   3   3   3   3   2   2   3   1   3   3   1   1   1   3   1   3   3   2 
 61  54  52  72 112  51 114  16  76  87  96 150 104  36 123  14  41  19  64  33 138  24  78  95  10 107  77 136 
  2   2   2   2   1   1   2   3   2   2   2   2   1   3   1   3   3   3   2   3   1   3   1   2   3   2   2   1 
131 125 108 120 140 106  88  63  18  40 111  20 145 147   3  90  31  43 134 137  85  26  67 128  21 144  34  93 
  1   1   1   2   1   1   2   2   3   3   1   3   1   2   3   2   3   3   2   1   2   3   2   2   3   1   3   2 
 50 139  92 127  84  28 105 109 100 149  23 126 121  56  12 103  98  35  82 113 130 146 148 143  48  11  69  94 
  3   2   2   2   2   3   1   1   2   1   3   1   1   2   3   1   2   3   2   1   1   1   1   2   3   3   2   2 
 66  29 119  73   7   2   5  22  58  37   9  32  59 116  62  44  57  60  71  55   6   1  68 141   8  45  80  81 
  2   3   1   2   3   3   3   3   2   3   3   3   2   1   2   3   2   2   2   2   3   3   2   1   3   3   2   2 
 25  89  46 117  65 102 133 
  3   2   3   1   2   2   1 

Within cluster sum of squares by cluster:
[1] 21.41784 36.81767 15.15100
 (between_SS / total_SS =  88.6 %)

Available components:

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

Here in kmeans(x,3,nstart=50), '3' indicates the number of clusters. In k-means algorithm firstly we choose a random seed of centroid then pass through the data and assign each observation to a cluster whose distance is minimum than other cluster and then again recalculate centroid and again pass through the data until there will be no new assignments. By 'nstart=50' we mean that whole process is repeated 50 times and the process is chosen with minimum within sum of square.

```
> #within sum of square and betwwn sum of square
> km$tot.withinss
[1] 73.3865
> prop_ws=km$tot.withinss/(km$tot.withinss+km$betweenss)*100
> km$betweenss
[1] 571.9045
> prop_bs=km$betweenss/(km$tot.withins+km$betweenss)*100
> prop_ws
[1] 11.37262
> prop_bs
[1] 88.62738

> km=kmeans(x,3, nstart =1)
> km$tot.withinss
[1] 73.46262
> prop_ws=km$tot.withinss/(km$tot.withinss+km$betweenss)*100
> km$betweenss
[1] 571.8284
> prop_bs=km$betweenss/(km$tot.withins+km$betweenss)*100
> prop_ws
[1] 11.38442
> prop_bs
[1] 88.61558
```

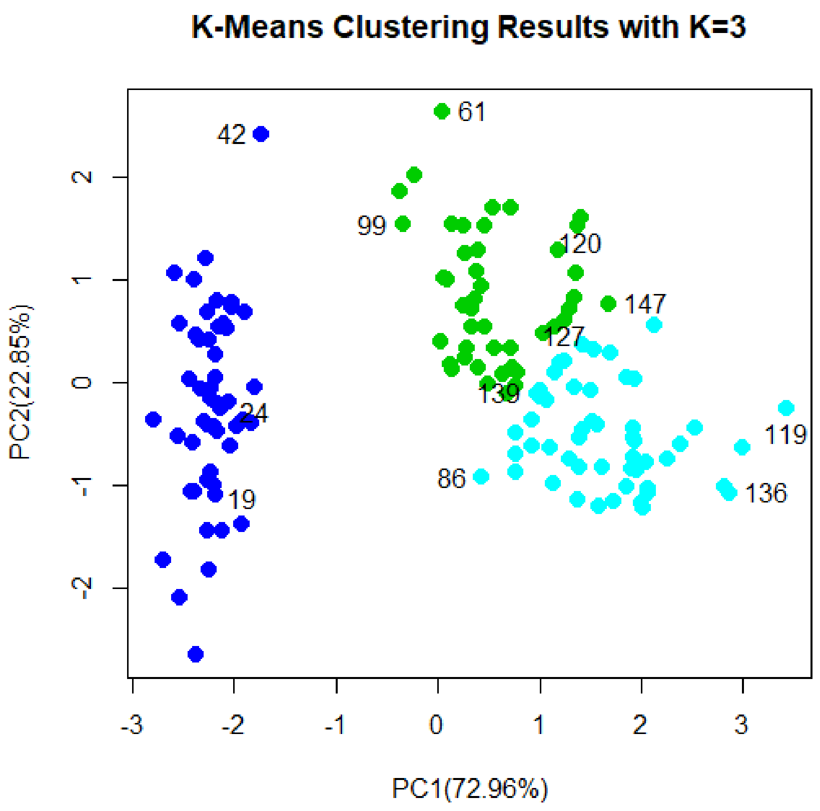
For ‘nstart=50’ within ss is 73.3865 which is 11.37262% of total variance where for ‘nstart=1’ within ss is 73.46262 which is 11.38442% of total variance, so, difference is clear by use of ‘nstart’. Here we have four variables and graph of k-means clustering is shown below :-

```
> plot(x, col =(km$cluster ) , main="K-Means Clustering Results with K=3", pch =20, cex =2)
```



For better visualization we use first two principal components and will identify the points of each cluster-

```
> #k means clustering using pc
> pc=prcomp(x,scale=T)
> y=pc$x[,1:2]
> km1=kmeans(y,3, nstart =50)
> plot(y, col =(km1$cluster +2 ) , main="K-Means Clustering Results with K=3",xlab='PC1(72.96%)',
ylab = 'PC2(22.85%)', pch =20, cex =2)
> identify(y[,1],y[,2],labels = rownames(y),plot = T)
```



We can track each points in the scatter-plot, few examples are shown above.

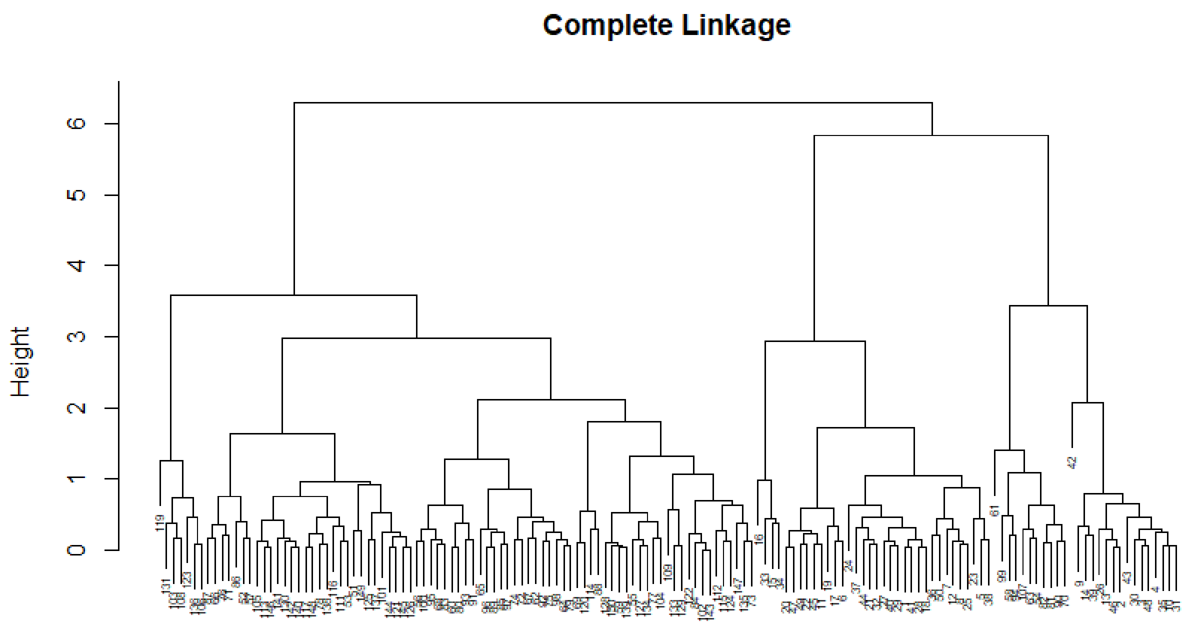
```
> km1
K-means clustering with 3 clusters of sizes 50, 52, 45

Cluster means:
        PC1        PC2
1  2.223998  0.2271097
2 -1.647628  0.5175360
3 -0.567183 -0.8503857

Clustering vector:
 79  30  83 122  70  86  99 115 124  97  49  47   4  42  91  74  17 135  15  39  53 101 142  27 129  38  13 
  3   1   3   3   3   2   3   2   2   3   1   1   1   1   3   3   1   3   1   1   2   2   2   1   2   1   1 
 75  61  54  52  72 112  51 114  16  76  87  96 150 104  36 123  14  41  19  64  33 138  24  78  95  10 107 
  3   3   3   2   3   2   2   3   1   2   2   3   2   2   1   2   1   1   1   3   1   2   1   2   3   1   3 
 77 136 131 125 108 120 140 106  88  63  18  40 111  20 145 147   3  90  31  43 134 137  85  26  67 128  21 
  2   2   2   2   2   3   2   2   3   3   1   1   2   1   2   3   1   3   1   1   2   2   3   1   3   2   1 
144  34  93  50 139  92 127  84  28 105 109 100 149  23 126 121  56  12 103  98  35  82 113 130 146 148 143 
  2   1   3   1   2   3   2   3   1   2   2   3   2   1   2   2   3   1   2   3   1   3   2   2   2   2   3 
 48  11  69  94  66  29 119  73   7   2   5  22  58  37   9  32  59 116  62  44  57  60  71  55   6   1  68 
  1   1   3   3   2   1   2   3   1   1   1   1   3   1   1   1   2   2   3   1   2   3   2   2   1   1   3 
141   8  45  80  81  25  89  46 117  65 102 133 
  2   1   1   3   3   1   3   1   2   3   3   2 
```

From the above picture it is clear that 50 observations in the first cluster, 52 observations in the second cluster and 45 observations in the third cluster. All observations are identified in each clusters.

Hierarchical Clustering(Agglomerative) :-

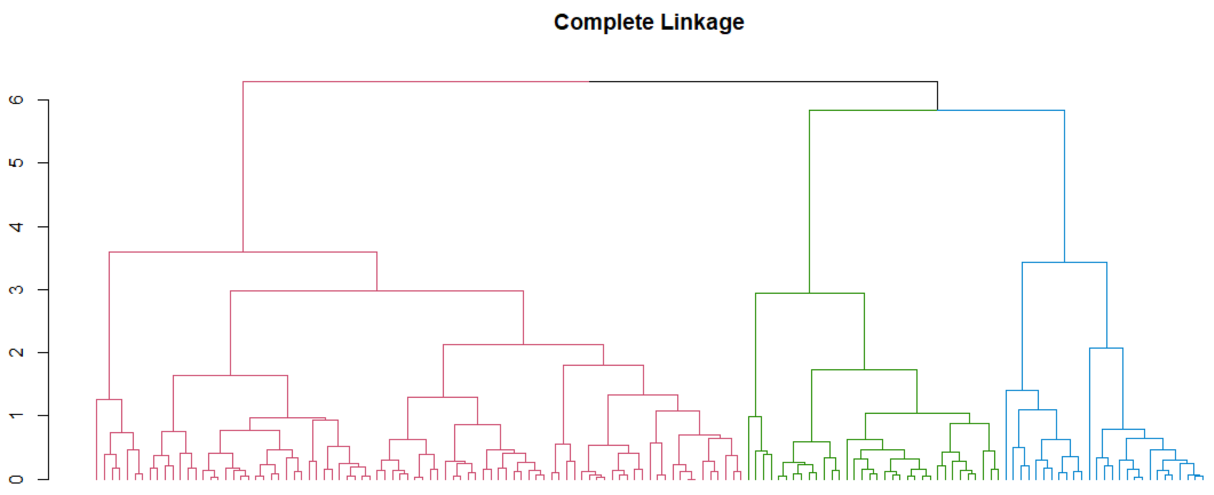
Agglomerative procedure starts with each observation in a singleton cluster (in total n clusters) and successively merges clusters together until all the n observations forms a single cluster. It is important to standardize the variables before clustering process as observations have p features and we deal with their distances. There are several ways to measure the distance between clusters in order to decide the rules for clustering, this measures are called Linkage method – complete linkage, average linkage, single linkage, centroid linkage etc. We will check by 'Complete Linkage'.

```
> ##Hierarchical Clustering
> #using first two principal component
> dist=dist(y, method = 'euclidean')
> hclust_comp= hclust(dist, method = 'complete')
> plot(hclust_avg,main='Complete Linkage',cex= 0.5)
> ct=cutree(hclust_comp,3)
> t=table(ct,rownames(y))
```



From the above graph(Dendrogram) the formation of clusters is clear. From PCA we know that there should be three clusters. Now if we cut this dendrogram bet-ween merger level 4 and 5 we will get three cluster. These clusters are shown in the dendrogram below-

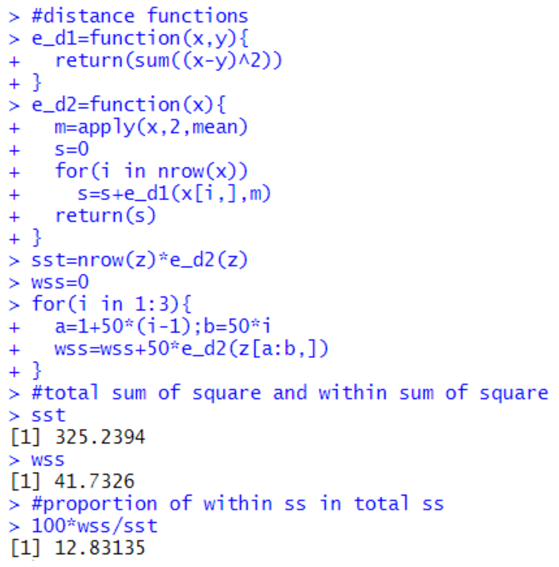
```
> #install.packages('dendextend', dependencies = TRUE)
> suppressPackageStartupMessages(library(dendextend))
> comp_dend_obj=as.dendrogram(hclust_comp)
> comp_col_dend=color_branches(comp_dend_obj, h = 4)
> plot(comp_col_dend,main='Complete Linkage')
```



Comparison :-

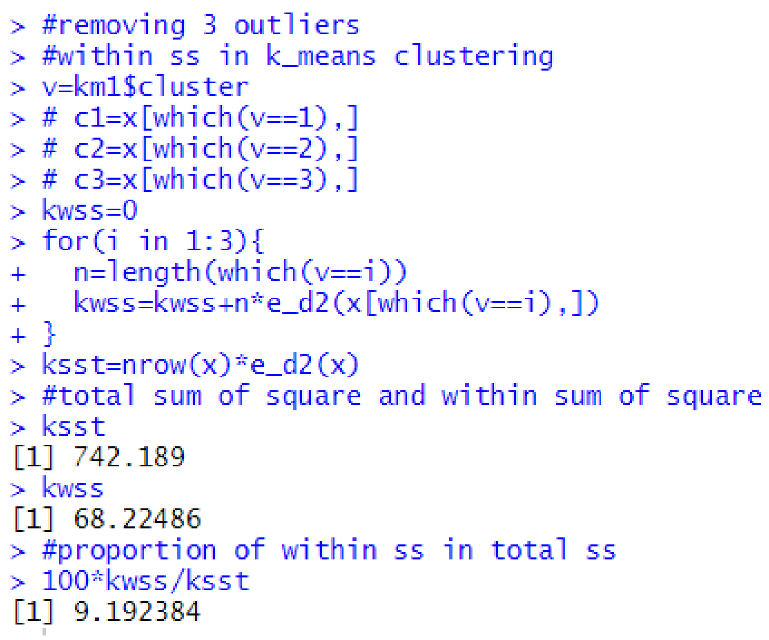
In clustering process one’s objective is to minimize within sum of squares. Let we have n observation $x_1, x_2, \dots, x_n$ and k cluster. Let C(i)=j means ith observation in jth cluster, $m_j$be the cluster mean, $n_j$be the number of observations in jth cluster for all j=1(1)k and i=1(1) n_j. Then we will minimize $W(c) = \sum_{j=1}^{k} n_j \sum_{C(i)=j} |\underline{x_i} - \underline{m_j}\,|^2$, where total sum of square $SST = n \sum_{l=1}^{n} |\underline{x_l} - \overline{x}|^2$ where $\overline{x} = \frac{1}{n} \sum_{l=1}^{n} \underline{x_i}$ $and\ \ n = \sum_j n_j$

<u>Main Data Label vs K-means Clustering</u> :



```
> #distance functions
> e_d1=function(x,y){
+   return(sum((x-y)^2))
+ }
> e_d2=function(x){
+   m=apply(x,2,mean)
+   s=0
+   for(i in nrow(x))
+     s=s+e_d1(x[i,],m)
+   return(s)
+ }
> sst=nrow(z)*e_d2(z)
> wss=0
> for(i in 1:3){
+   a=1+50*(i-1);b=50*i
+   wss=wss+50*e_d2(z[a:b,])
+ }
> #total sum of square and within sum of square
> sst
[1] 325.2394
> wss
[1] 41.7326
> #proportion of within ss in total ss
> 100*wss/sst
[1] 12.83135
```

Main data is labelled by ‘setosa’, ‘versicolor’ ans ‘verginica’. Regarding this labels or clusters within sum of squares is 41.7326 and it’s 12.83% of total sum of square.

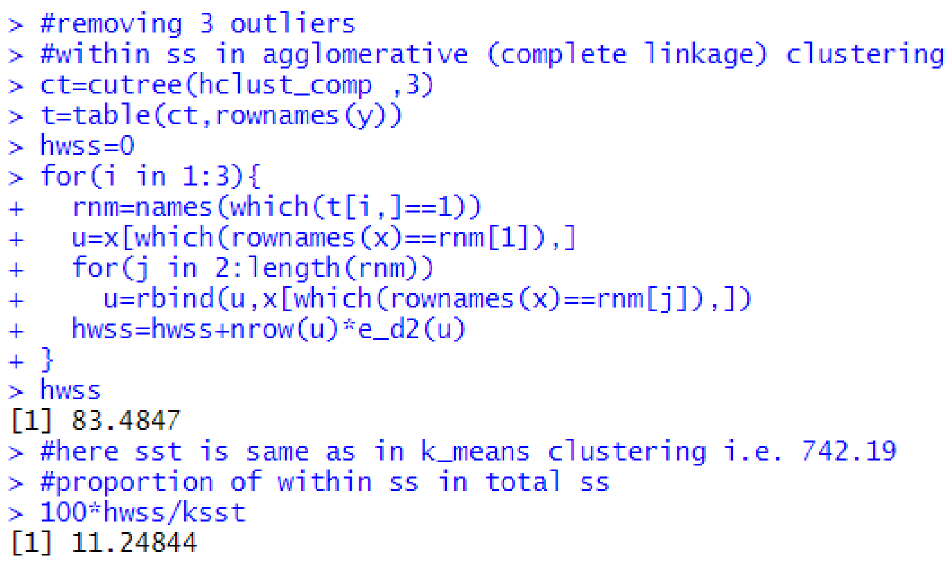


```
> #removing 3 outliers
> #within ss in k_means clustering
> v=km1$cluster
> # c1=x[which(v==1),]
> # c2=x[which(v==2),]
> # c3=x[which(v==3),]
> kwss=0
> for(i in 1:3){
+   n=length(which(v==i))
+   kwss=kwss+n*e_d2(x[which(v==i),])
+ }
> ksst=nrow(x)*e_d2(x)
> #total sum of square and within sum of square
> ksst
[1] 742.189
> kwss
[1] 68.22486
> #proportion of within ss in total ss
> 100*kwss/ksst
[1] 9.192384
```

After removing 3 outliers , in k-means clustering we found that within sum of squares is 68.22486 and it's 9.2% of total sum of square(742.19).

So, k-means clustering partition the data better than actual labellling with respect to within sum of square (as 12.83 > 9.2).

<u>Main Data Label vs Hierarchical Clustering</u> :



```
> #removing 3 outliers
> #within ss in agglomerative (complete linkage) clustering
> ct=cutree(hclust_comp ,3)
> t=table(ct,rownames(y))
> hwss=0
> for(i in 1:3){
+   rnm=names(which(t[i,]==1))
+   u=x[which(rownames(x)==rnm[1]),]
+   for(j in 2:length(rnm))
+     u=rbind(u,x[which(rownames(x)==rnm[j]),])
+   hwss=hwss+nrow(u)*e_d2(u)
+ }
> hwss
[1] 83.4847
> #here sst is same as in k_means clustering i.e. 742.19
> #proportion of within ss in total ss
> 100*hwss/ksst
[1] 11.24844
```

In hierarchical clustering we found that within sum of squares is 83.4847 and it's 11.25% of total sum of square(742.19).

So, here hierarchical clustering partition the data better than actual labelling with respect to within sum of square (as 12.83 > 11.25).

References :-

1. An Introduction to Statistical Learning: with Applications in R (Springer Texts in Statistics)
2. YouTube Videos
3. www.datacamp.com
4. www.statisticssolution.com
5. www.mathworks.com
6. www.tutorialspoint.com
7. www.datanovia.com
8. www.r-bloggers.com