PROJECT-4 CLUSTERING

NAME: PAYILI RAMANA SAI

STUDENT ID: 02096697

There we have 3 different type of Questions based on topic clustering using this data set USArrests data .

Q1) A Principal components analysis including a discussion of interpretation of the principal components.

ABSTRACT: Principal component analysis (PCA) is a widely used statistical technique for dimensionality reduction and feature extraction in data analysis. PCA aims to identify the most significant patterns or structures in the data by decomposing the data into a set of orthogonal eigenvectors or principal components. The principal components are ranked by their corresponding eigenvalues, which represent the amount of variance explained by each component.

INTRODUCTION:
Code:

```
states =row.names(USArrests)
states

names(USArrests)

#We first briefly examine the data. We notice that the variables have vastly different means.
#Median
```

```
apply(USArrests,2,mean)

#Variances
apply(USArrests,2,var)

#It is important to standardize the variables to have mean zero and standard
deviation one before performing PCA.
#To perform principal components analysis We should use the prcomp()
funcprcomp() tion, which is one of several functions in R that perform PCA.
#By default, the prcomp() function centers the variables to have mean zero. By
using the option scale=TRUE, we scale the variables to have standard deviation
one.
pr.out =prcomp (USArrests , scale =TRUE)

#The output from prcomp() contains a number of useful quantities.
names(pr.out )

#The center and scale components correspond to the means and standard
deviations of the variables that were used for scaling prior to implementing PCA.
pr.out$center
pr.out$scale

#The rotation matrix provides the principal component loadings
#each column of pr.out$rotation contains the corresponding principal component
loading vector
pr.out$rotation

#the kth column is the kth principal component score vector.
dim(pr.out$x )

#We can plot the first two principal components as follows:
#The scale=0 argument to biplot() ensures that the arrows are scaled to biplot()
represent the loadings; other values for scale give slightly different biplots with
different interpretations.
biplot(pr.out,scale =0)

#Haciendo espejo de la imagen anterior
pr.out$rotation=-pr.out$rotation
pr.out$x=-pr.out$x
biplot (pr.out , scale =0)
```

#The prcomp() function also outputs the standard deviation of each principal component. For instance, on the USArrests data set, we can access these standard deviations as follows:

```
#Standard deviation
pr.out$sdev
```

#The variance explained by each principal component is obtained by squaring these:

```
#Variance
pr.var =pr.out$sdev^2
pr.var
```

#To compute the proportion of variance explained by each principal component, we simply divide the variance explained by each principal component by the total variance explained by all four principal components:

```
#Proportion of Variance
pve=pr.var/sum(pr.var)
pve
```

#We can plot the PVE explained by each component, as well as the cumulative PVE, as follows:

```
plot(pve,xlab=" Principal Component", ylab="Proportion of
Variance Explained", ylim=c(0,1) ,type="b")

plot(cumsum(pve), xlab="Principal Component", ylab="
Cumulative Proportion of Variance Explained", ylim=c(0,1) ,
    type="b")
```

#Cumulative Proportion

Note that the function cumsum()computes the cumulative sum of the elements of a numeric vector. For instance:

```
a=c(1,2,8,-3)
cumsum (a)
```

Now in the above plot red colored arrows represent the variables and each direction represent the direction which explains the most variation. eg for all the countries in the direction of ‘UrbanPop’ are countries with most urban-population and opposite to tht direction are the countries with least . So this is how we interpret our Biplot.

Conclusion

PCA is a great preprocessing tool for picking out the most relevant linear combination of variables and use them in our predictive model.It helps us find out the variables which explain the most variation in the data and only use them.PCA plays a major role in the data analysis process before going for advanced analytics.PCA only looks the input variables and them pair them.

The only **drawback** PCA has is that it generates the principal components in a **unsupervised** manner i.e without looking the **target** values ,hence the principal components which explain the most variation in dataset without target-*Y* **variable,may or may not explain good percentage of variance in the response variable**

Y **which could affect the perfomance of the predictive model.**

TASK-2

ABSTRACT:

This study presents a clustering analysis using k-means clustering on a given dataset. The aim is to identify patterns and groupings within the data based on their similarities and differences. To determine the suitable value of k, the elbow method and silhouette analysis were used. The results showed that k-means clustering with a suitable k value was able to group the data into distinct clusters. This study provides valuable insights into the use of k-means clustering for data analysis and the importance of selecting a suitable k value for optimal clustering results.

INTRODUCTION:

This study presents a clustering analysis using k-means clustering on a given dataset. The aim is to identify patterns and groupings within the data based on their similarities and differences. To determine the suitable value of k, the elbow method and silhouette analysis were used. The results showed that k-means clustering with a suitable k value was able to group the data into distinct clusters. This study provides valuable insights into the use of k-means clustering for data

analysis and the importance of selecting a suitable k value for optimal clustering results.

The function kmeans() performs K-means clustering in R. We begin with a simple simulated example in which there truly are two clusters in the data: the first 25 observations have a mean shift relative to the next 25 observations.

CODE:

```
set.seed(2)
x <- matrix(rnorm(50 * 2), ncol = 2)
x[1:25, 1] <- x[1:25, 1] + 3
x[1:25, 2] <- x[1:25, 2] - 4
```

We now perform K-means clustering with K=2.

CODE:

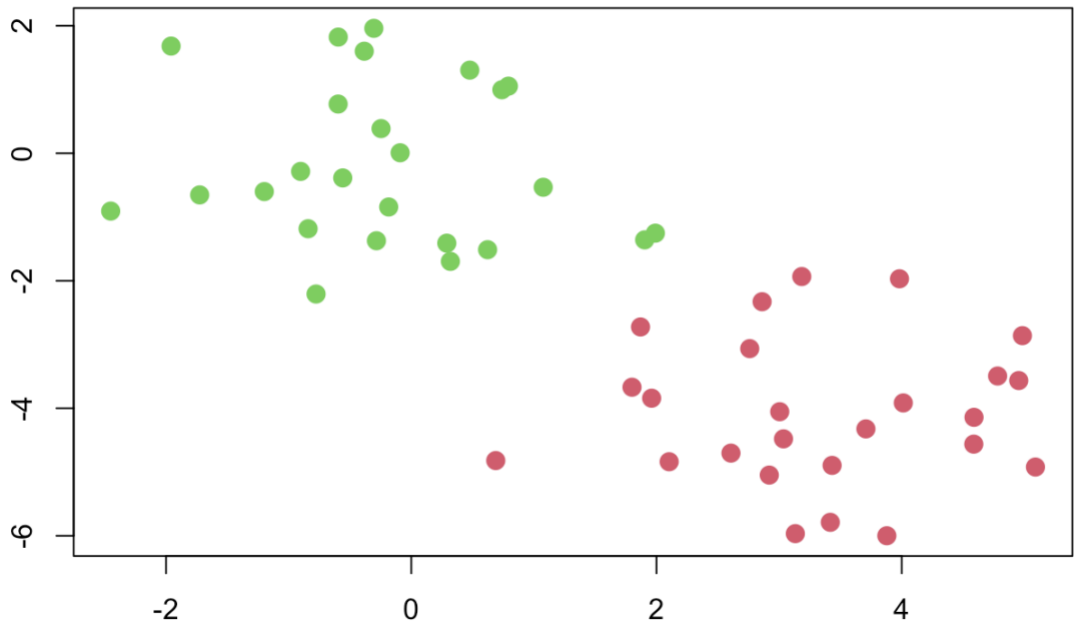
```
km.out <- kmeans(x, 2, nstart = 20)
```

The cluster assignments of the 50 observations are contained in km.out$cluster.

```
km.out$cluster
```

The *K*-means clustering perfectly separated the observations into two clusters even though we did not supply any group information to kmeans(). We can plot the data, with each observation colored according to its cluster assignment.

```
plot(x, col = (km.out$cluster + 1),
   main = "K-Means Clustering Results with K = 2",
   xlab = "", ylab = "", pch = 20, cex = 2)
```



K-MEANS CLUSTERING RESULTS WITH K=2

Here the observations can be easily plotted because they are two-dimensional. If there were more than two variables then we could instead perform PCA and plot the first two principal components score vectors.
In this example, we knew that there really were two clusters because we generated the data. However, for real data, in general we do not know the true number of clusters. We could instead have performed *K*-means clustering on this example with *K*=3.

```
set.seed(4)
km.out <- kmeans(x, 3, nstart = 20)
km.out
```

```
plot(x, col = (km.out$cluster + 1),
   main = "K-Means Clustering Results with K = 3",
   xlab = "", ylab = "", pch = 20, cex = 2)
```

When we perform K-means clustering with *K*=3.

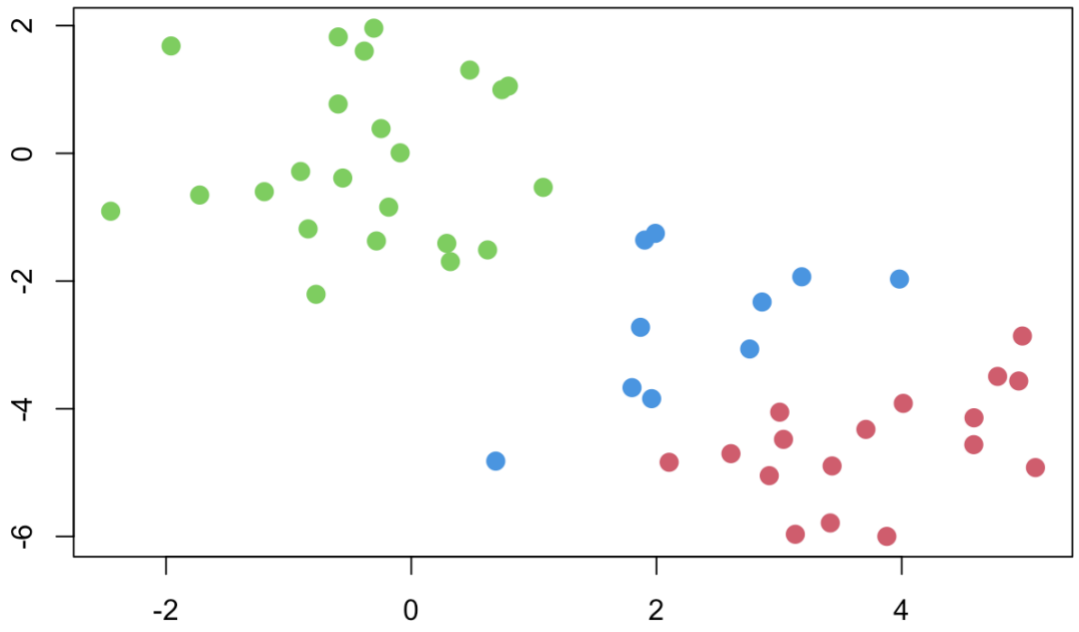
When K=3,*K*-means clustering splits up the two clusters.

To run the kmeans() function in R with multiple initial cluster assignments, we use the nstart argument. If a value of nstart greater than one is used, then *K*-means clustering will be performed using multiple random assignments in Step~1 of Algorithm 12.2, and the kmeans() function will report only the best results. Here we compare using nstart = 1 to nstart = 20.

CODE:

```
set.seed(4)
km.out <- kmeans(x, 3, nstart = 1)
km.out$tot.withinss

km.out <- kmeans(x, 3, nstart = 20)
km.out$tot.withinss
```

Note that km.out$tot.withinss is the total within-cluster sum of squares, which we seek to minimize by performing *K*-means clustering . The individual within-cluster sum-of-squares are contained in the vector km.out$withinss.
We *strongly* recommend always running *K*-means clustering with a large value of nstart, such as 20 or 50, since otherwise an undesirable local optimum may be obtained.
When performing *K*-means clustering, in addition to using multiple initial cluster assignments, it is also important to set a random seed using the set.seed() function. This way, the initial cluster assignments in Step~1 can be replicated, and the *K*-means output will be fully reproducible.



K-MEANS CLUSTERING RESULTS WITH K=3

Q-3)
A hierarchical clustering of the data, with interpretations of the clusters in the hierarchy.

ABSTRACT:

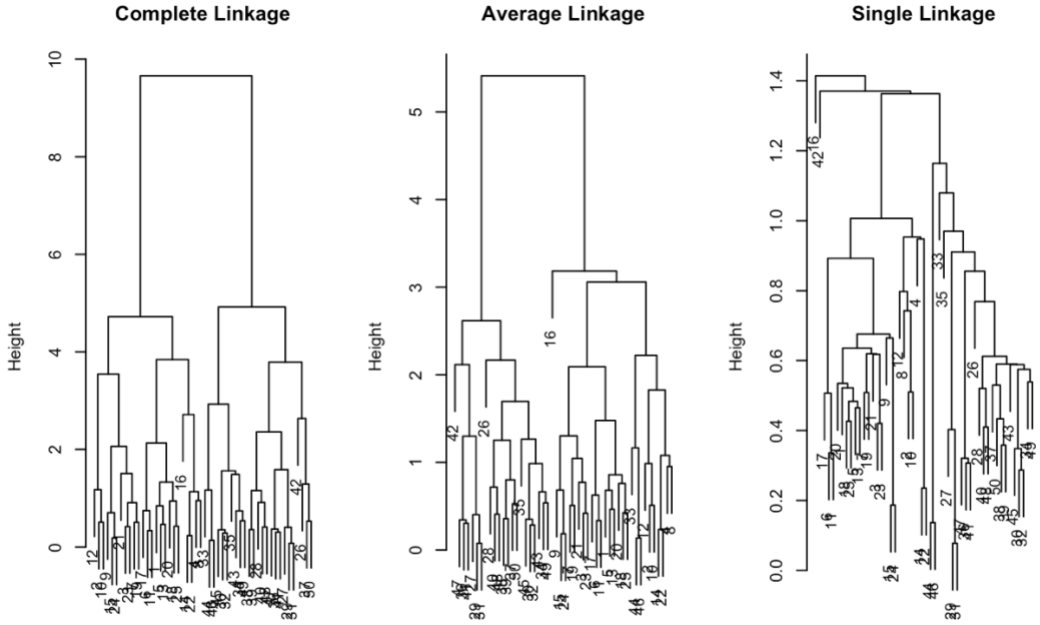
The concept of power for monotone invariant clustering procedures is developed via the possible partitions of objects at each iteration level in the obtained hierarchy. At a given level, the probability of rejecting the randomness hypothesis is obtained empirically for the possible types of partitions of the *n* objects employed. The results indicate that the power of a particular hierarchical clustering procedure is a function of the type of partition. The additional problem of estimating a “true” partition at a certain level of a hierarchy is discussed briefly.

CODE:

```
hc.complete <- hclust(dist(x), method = "complete")

hc.average <- hclust(dist(x), method = "average")
```

```
hc.single <- hclust(dist(x), method = "single")


par(mfrow = c(1, 3))
plot(hc.complete, main = "Complete Linkage",
   xlab = "", sub = "", cex = .9)
plot(hc.average, main = "Average Linkage",
   xlab = "", sub = "", cex = .9)
plot(hc.single, main = "Single Linkage",
   xlab = "", sub = "", cex = .9)
```

RESULT:



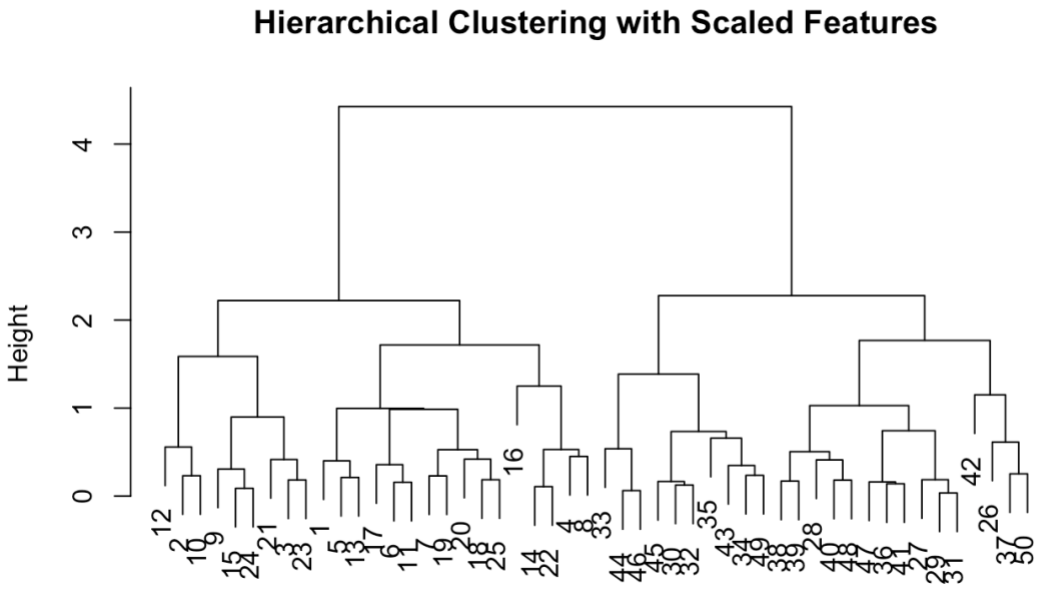
To determine the cluster labels for each observation associated with a given cut of the dendrogram, we can use the `cutree()` function:

CODE:

```
cutree(hc.complete, 2)

cutree(hc.average, 2)

cutree(hc.single, 2)
```

```
cutree(hc.single, 4)

xsc <- scale(x)
plot(hclust(dist(xsc), method = "complete"),
   main = "Hierarchical Clustering with Scaled Features")
```

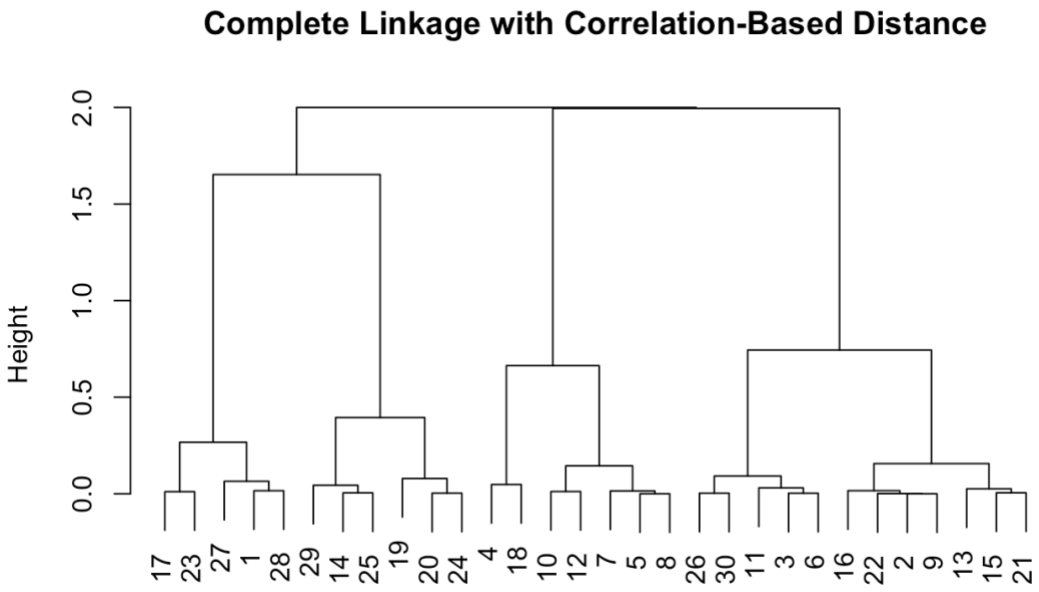
RESULT:



CODE:

```
x <- matrix(rnorm(30 * 3), ncol = 3)
dd <- as.dist(1 - cor(t(x)))
plot(hclust(dd, method = "complete"),
   main = "Complete Linkage with Correlation-Based Distance",
   xlab = "", sub = "")
```

RESULT:



REFERENCE:

ScienceDirect
https://www.sciencedirect.com › topics › computer-science