Learning outcomes:

After completing this exercise, you should be able to understand and perform below tasks.

- Applying K-means Clustering and Hierarchical clustering
- Understand various cluster metrics generated in R.
- Evaluating the performance of clusters.
- Understanding the importance of standardizing data.
- Visualization and interpretation of results.

Clustering Activity:

On the inbuild 'mtcars' data set, we will be clustering the similar cars based on different features using K-means and Hierarchial clustering.

Objective: can we find cars that are similar on various aspects?

R Code:

1. Load inbuild 'mtcars' data available in R
2. Understand the data and apply the necessary pre-processing steps.
3. Normalize/Scale the data.
   Note: Identify the cluster performance with and without normalizing/scaling the data and identify the importance of the scaling the data.

#Hierarchical Clustering Activity:

1. Calculate the distance between different cars using "dist" function using different distance methods.
   ```
   d <- dist(mydata, method = "euclidean") # distance matrix
   d
   ```
   Note: Experiment with different distance methods.
2. Build the hierarchical clustering using "hclust" function using agglomerative method ward.D2
   ```
   fit <- hclust(d, method="ward.D2")
   ```
   Note: You can explore different methods single, complete, average
3. Visualize the clusters. Tree like structure is called as dendrogram.
   ```
   plot(fit)
   ```
   # dendrogram displays all possible clusters from the data in bottom up approach
4. Creating 5 clusters using cutree function, "K" specifies number of cluster to create.
   ```
   groups <- cutree(fit, k=5) # cut tree into 5 clusters
   groups
   ```
   # draw dendogram with red borders around the 5 clusters
   ```
   rect.hclust(fit, k=5, border="red")
   ```
5. Append cluster labels to the actual data frame
   ```
   Mydata_cluster <- data.frame(mydata, groups)
   ```

Reference:
https://websites.pmc.ucsc.edu/~mclapham/Rtips/cluster.htm
http://rstudio-pubs-static.s3.amazonaws.com/5378_6b71c972feb04fd19776daa9063df5cd.html

K-means clustering:

6. Build the cluster using kmeans function by mentioning the number of clusters. Ensure there should be no target attribute as this is an unsupervised learning technique.

```
# K-means clustering
fit<-kmeans(mydata,centers=2)
fit
```

7. Check sum of inter cluster distance(betweenness) and Intra cluster distances(With-in sum of squares).

```
fit$withinss
sum(fit$withinss)
#Cluster Centers
fit$centers
#To check cluster number of each row in data
fit$cluster
```

8. Identifying the ideal number of cluster:
   - Write a for loop which should start with 2 clusters and build k-means model up to 15 clusters.
   - Capture the within-sum of squares for different number of cluster, save sum(fit$withinss) for each model.
   - Plot sum(fit$withinss) generated in all models
   - Find the best cluster based on the curve.

9. For new data, how do you know which cluster does it belong to?

   Lets select on record randomly from the data. Given the cluster centres we can find the distance between the new point and cluster centre. The one with smallest distance will be the cluster number assigned to the new data.

```
test_datapoint <- mtcars[sample(1:nrow(mtcars),1),]
closest.cluster <- function(x) {
    cluster.dist <- apply(fit$centers, 1, function(y) sqrt(sum((x-y)^2)))
    print(cluster.dist)
    return(which.min(cluster.dist)[1])
}

closest.cluster(test_datapoint)
```

10. Stability of clusters is important. Ideally, we do not want the clusters to change when new data is added. The way to evaluate is take a 90 or 95% of the records and apply k-means algorithm with the same number of clusters as decided for complete data. Using the 'adj.rand.index' function we can see how much is the deviation in clustering.

```
## stability check
set.seed(12)
index <- (sample(nrow(mydata),.90*nrow(mydata)))
dataTest <- mydata[index,]
StabClus <- kmeans(dataTest,5)
dataTest$clusters <- StabClus$cluster

group1 <- mydata[index,12]
group2 <- dataTest$clusters
group <- cbind(group1, group2)
write.csv(group, "clusgroup.csv")

#install.packages("fossil")
library(fossil)
stabilitycheck <- adj.rand.index(group1, group2)
stabilitycheck
```

For more: https://davetang.org/muse/2017/09/21/adjusted-rand-index/

```
#install.packages("clusteval")
library(clusteval)
Stabindex <- cluster_similarity(group1, group2, similarity = "jaccard",
method="independence")
Stabindex
```

11. How to deal with mixture of attribute types? (Good to know)

In all the above steps we have assumed that the data is numeric and have scaled them. This is mainly because we were applying 'Eucledian' distance measure. However, there is a distance metric 'gower' which can deal with mixture of attribute types.

```
mydata2<-mtcars
names(mydata2)
#categorical data
data1<-as.data.frame(apply(mydata2[,c(2,8,9,10)],2,as.factor))
#Numeric data - standardization
```

```
data2<-scale(mydata2[,-c(2,8,9,10)],scale=T,center = T)
data_allnum<-scale(mydata2,scale=T,center=T)

# combine
dat_gower_numcat<-cbind(data2,data1)
# distance using gower measure
distMat <- daisy(dat_gower_numcat, metric = "gower")
# hierarchical clustering
fitGower<-hclust(distMat,method="single")
dev.off()
plot(fitGower)
```

What is the alternate way of deal with mixture of attribute types?

Convert all the categorical attributes into numeric and later scale before computing distance matrix

12. Now the challenge is, how to explain the clusters to business and use them. Can we find some cluster characteristics?

    Let's inspect the cluster centres from K-means output and try to name them. This step has to be done in collaboration with the business teams and agree.

    fit$centers

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
1	-0.5404546	0.6415922	0.2819522	1.6235100	0.3680169	-0.0482903	-1.6688182	-0.8680278	1.1899014	1.7789276	1.9734398
2	-0.0565170	-0.1049878	-0.5399595	-0.4402893	0.5862165	-0.1262191	-0.1000312	0.124004	0.1878792	0.4235542	0.7352031
3	-0.8363478	1.0148821	1.0238513	0.6924910	-0.8897477	0.9063586	-0.3952280	-0.8680278	-0.8141431	-0.9318192	0.1676779
4	1.3247791	-1.2248578	-1.1062677	-0.9453003	1.0982062	-1.2008698	0.3364684	0.8680278	1.1899014	0.7623975	-0.8125929
5	0.2570746	-0.7769098	-0.4244185	-0.7713723	-0.3115188	-0.1239195	1.4915135	1.1160357	-0.8141431	-0.3896699	-0.8745047

Cluster 1 – cars with high 'HP', 'AM', 'Gear', 'Carb'
Cluster 3 – cars with high 'Disp' and 'Cyl'
Cluster 4 – cars with high 'MPG' and 'Drat'
Cluster 5 – cars with high 'Qsec' and 'VS'
Cluster 2 – otherwise

Assignment:

Cereals data: This data set contains nutritional information for 77 different breakfast cereals. It was used for the 1993 Statistical Graphics Exposition as a challenge data set. We retrieved this data from StatLib at CMU. The data is from the nutritional labels and is in CSV format.

Objective: the schools wants to pick a cereal each day of the week from similar cereals.

The variables are:

- Cereal name;
- manufacturer (e.g.,Kellogg's);
- type (cold/hot);
- calories (number);
- protein (g);
- fat (g);
- sodium (mg);
- dietary fiber (g);
- complex carbohydrates (g);
- sugars (g);
- display shelf (1, 2, or 3, counting from the floor);
- potassium (mg);
- vitamins and minerals (0, 25, or 100, respectively);
- weight (in ounces) of one serving (serving size);
- cups per serving.

Manufacturers are represented by their first initial: A=American Home Food Products, G=General Mills, K=Kelloggs, N=Nabisco, P=Post, Q=Quaker Oats, R=Ralston Purina.

(Download)

Using K-means technique identify/cluster the similar cereals.

- Load the cereals data into R.
- Analyze the data and apply the required pre-processing steps and prepare data for clustering.
- Use a distance metric to compute distance matrix.(optional, unless trying hierarchical clustering)
- Apply k-means clustering technique, identify the ideal number of cluster.
- Identify the similar cereals based on the clusters.

K-medoids clustering. Please use the below R code as reference and then implement on your data set

```
## generate 25 objects, divided into 2 clusters.
x <- rbind(cbind(rnorm(10,0,0.5), rnorm(10,0,0.5)),
        cbind(rnorm(15,5,0.5), rnorm(15,5,0.5)))

library(cluster)
pamx <- pam(x, 2)
plot(pamx)
## use obs. 1 & 16 as starting medoids -- same result (typically)
(p2m <- pam(x, 2, medoids = c(1,16)))

# rather naive initial medoids:
p3m <- pam(x, 3, medoids = 3:1, trace = 1)
# By default the distance metric is euclidean
# you can choose the other metrics
pam(daisy(x, metric = "manhattan"), 2, diss = TRUE)
```