AIM: IMPLEMENTATION AND ANALYSIS OF CLUSTERING ALGORITHMS LIKE K MEANS AGGLOMERATIVE

THEORY:

K-Means Clustering Algorithm

K-Means Clustering is an unsupervised learning algorithm that is used to solve the clustering problems in machine learning or data science. In this topic, we will learn what is K-means clustering algorithm, how the algorithm works, along with the Python implementation of k-means clustering.

What is K-Means Algorithm?

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

It is an iterative algorithm that divides the unlabeled dataset into k different clusters in such a way that each dataset belongs only one group that has similar properties.

It allows us to cluster the data into different groups and a convenient way to discover the categories of groups in the unlabeled dataset on its own without the need for any training.

It is a centroid-based algorithm, where each cluster is associated with a centroid. The main aim of this algorithm is to minimize the sum of distances between the data point and their corresponding clusters.

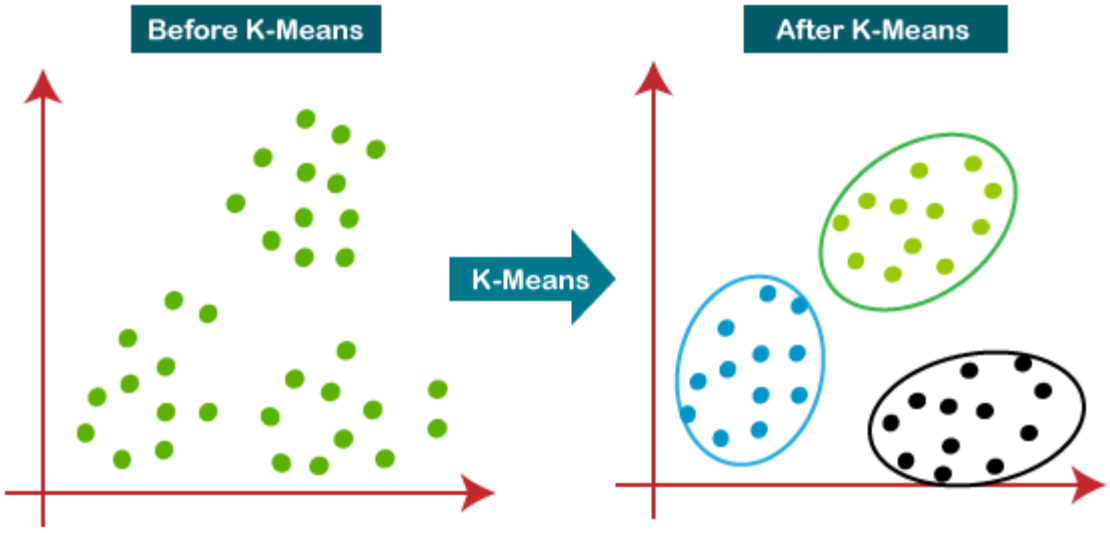
The algorithm takes the unlabeled dataset as input, divides the dataset into k-number of clusters, and repeats the process until it does not find the best clusters. The value of k should be predetermined in this algorithm.

The k-means clustering algorithm mainly performs two tasks:

- Determines the best value for K center points or centroids by an iterative process.
- Assigns each data point to its closest k-center. Those data points which are near to the particular k-center, create a cluster.

Hence each cluster has datapoints with some commonalities, and it is away from other clusters.

The below diagram explains the working of the K-means Clustering Algorithm:



How does the K-Means Algorithm Work?

The working of the K-Means algorithm is explained in the below steps:

Step-1: Select the number K to decide the number of clusters.

Step-2: Select random K points or centroids. (It can be other from the input dataset).

Step-3: Assign each data point to their closest centroid, which will form the predefined K clusters.

Step-4: Calculate the variance and place a new centroid of each cluster.

Step-5: Repeat the third steps, which means reassign each datapoint to the new closest centroid of each cluster.

Step-6: If any reassignment occurs, then go to step-4 else go to FINISH.

Step-7: The model is ready.

FUNCTIONS USED IN R:

`hclust:` Performs hierarchical clustering on a distance or similarity structure.

`cutree:` Returns a vector of group numbers for the observations that were clustered. Specify either the number of groups desired or a clustering height.

A) IMPLEMENT K-MEANS CLUSTERING

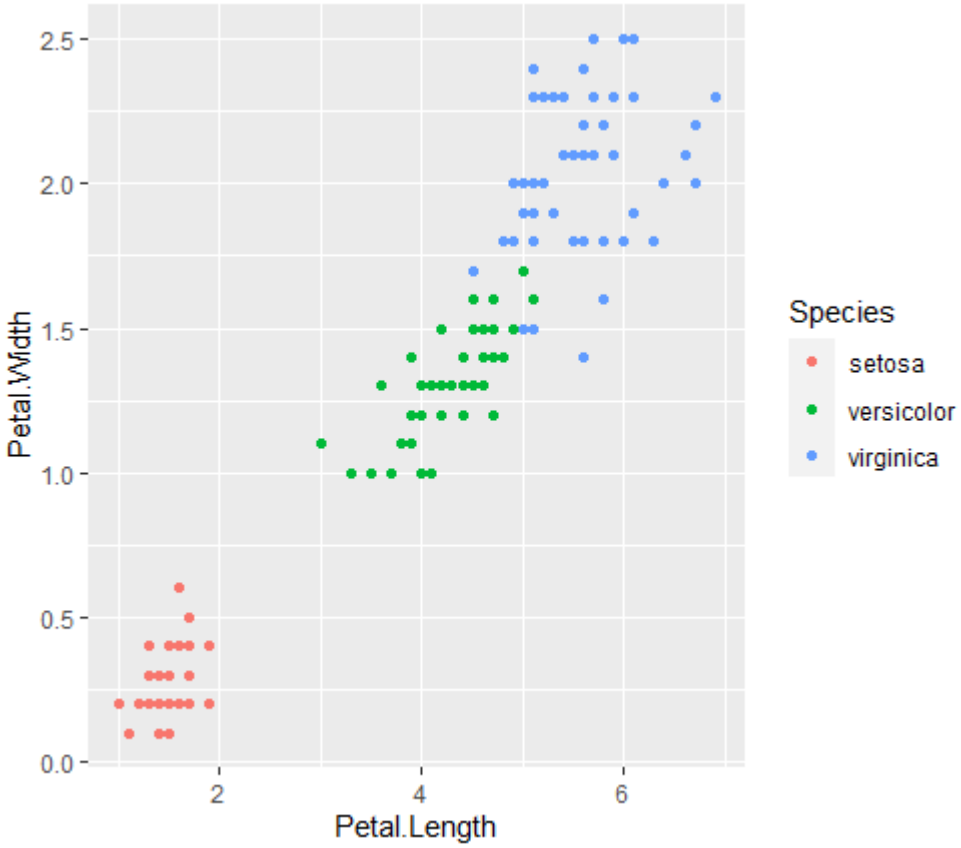
LOADING DATASET:

#K-Means clustering
head(iris)

```
> head(iris)
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
4          4.6         3.1          1.5         0.2  setosa
5          5.0         3.6          1.4         0.2  setosa
6          5.4         3.9          1.7         0.4  setosa
```

LOADING LIBRARY FOR PLOTING GRAPH:

library(ggplot2)
ggplot(iris, aes(Petal.Length, Petal.Width, color = Species)) + geom_point()



EXECUTING K-MENAS AND PLOTTING GRAPH:

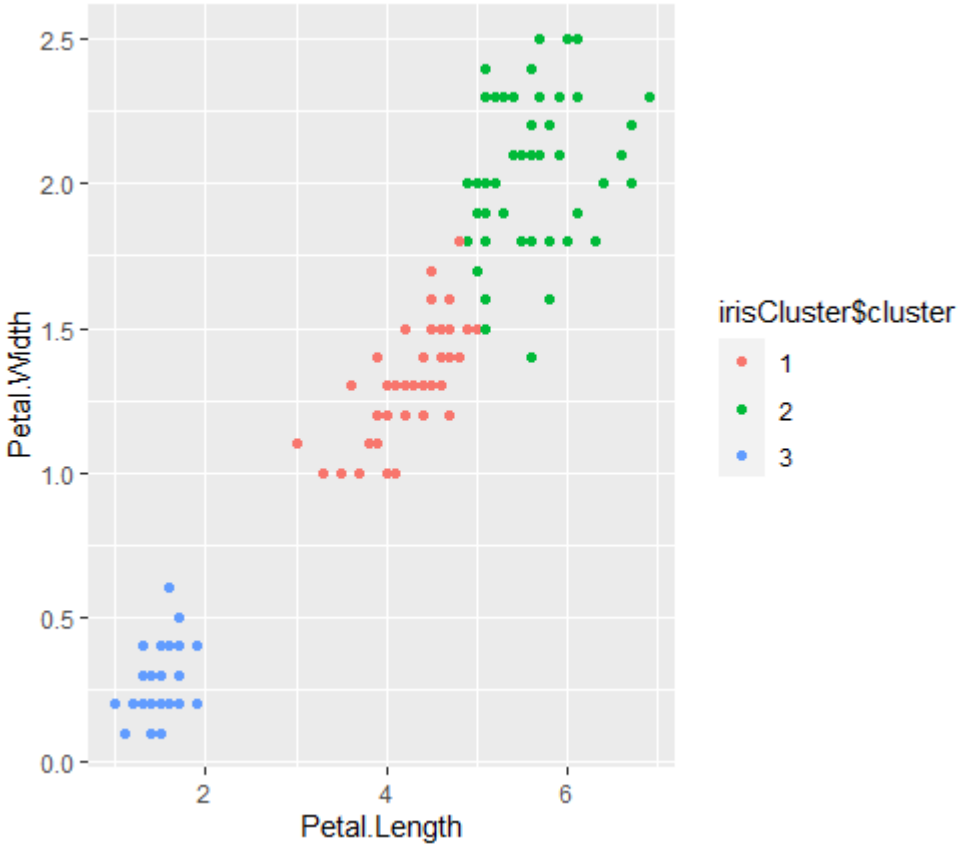
#Setting a seed in R means to initialize a pseudorandom number generator.
set.seed(20)

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

table(irisCluster$cluster, iris$Species)

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

```
> #Setting a seed in R means to initialize a pseudorandom number generator.
> set.seed(20)
>
> #Executing Kmeans
> irisCluster <- kmeans(iris[, 3:4], 3, nstart = 20)
> irisCluster
K-means clustering with 3 clusters of sizes 52, 48, 50

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

Clustering vector:
  [1] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
 [46] 3 3 3 3 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 2 1 1 1 1 1
 [91] 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 1 2 2
[136] 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2

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

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss" "betweenss"
[7] "size"         "iter"         "ifault"
`
> table(irisCluster$cluster, iris$Species)

    setosa versicolor virginica
  1      0         48         4
  2      0          2        46
  3     50          0         0
>
> irisCluster$cluster <- as.factor(irisCluster$cluster)
> ggplot(iris, aes(Petal.Length, Petal.Width, color = irisCluster$cluster)) + geom_point()
`
```


Petal.Width
Petal.Length
irisCluster$cluster
1
2
3
0.0
0.5
1.0
1.5
2.0
2.5
2
4
6

B) K-MEANS AGGOLOMERATIVE

LOADING DATASET:

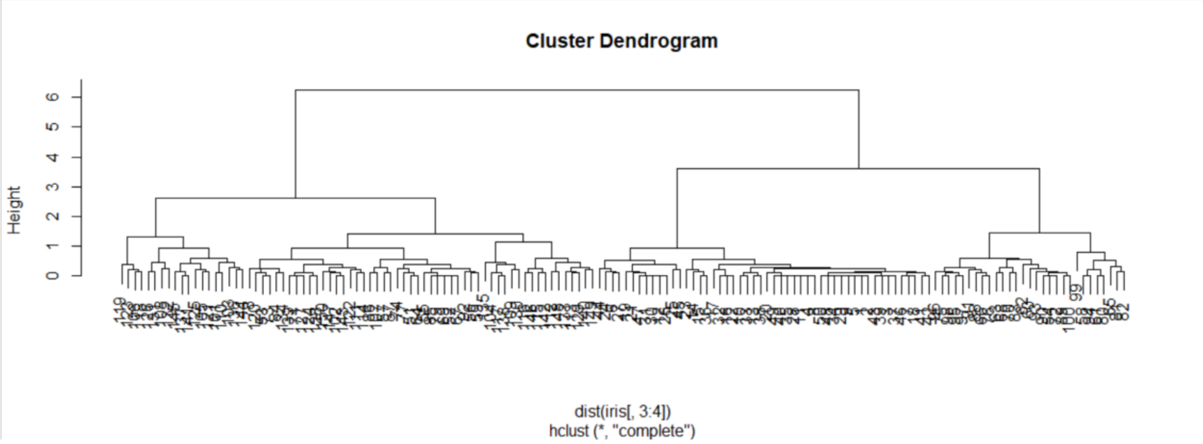
#K-Means Agglomerative
head(iris)

```
> #K-Means Agglomerative
> head(iris)
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
4          4.6         3.1          1.5         0.2  setosa
5          5.0         3.6          1.4         0.2  setosa
6          5.4         3.9          1.7         0.4  setosa
```

CREATING CLUSTERS:

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

```
> clusters <- hclust(dist(iris[, 3:4]))
> clusters

Call:
hclust(d = dist(iris[, 3:4]))

Cluster method   : complete
Distance         : euclidean
Number of objects: 150

> plot(clusters)
```



CREATING GROUP OF CLUSTERS:

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

```
> clusterCut<- cutree(clusters, 3)
> clusterCut
  [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
 [46] 1 1 1 1 1 2 2 2 3 2 2 2 3 2 3 3 3 3 2 3 3 2 3 2 3 2 3 2 2 3 3 2 2 2 3 3 3 3 2 2 2 2 3 3 3
 [91] 3 2 3 3 3 3 3 3 3 3 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
[136] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
> table(clusterCut, iris$Species)

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

CREATING AVERAGE LINKAGE CLUSTERS:

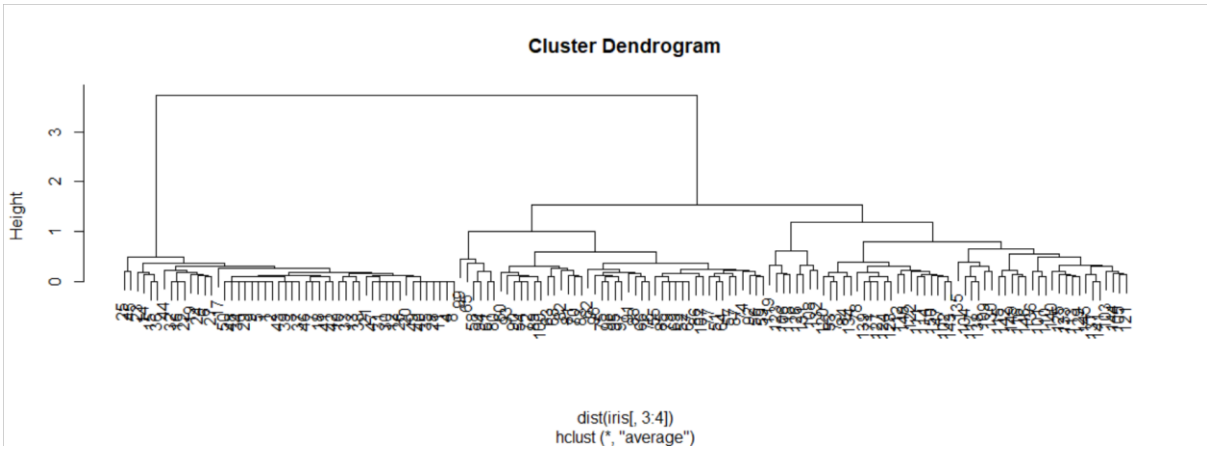
clusters <- hclust(dist(iris[, 3:4]), method = 'average')
clusters
plot(clusters)

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

Call:
hclust(d = dist(iris[, 3:4]), method = "average")

Cluster method   : average
Distance         : euclidean
Number of objects: 150

> plot(clusters)
```

CREATING GROUP CLUSTER FOR AVERGE LINKAGE

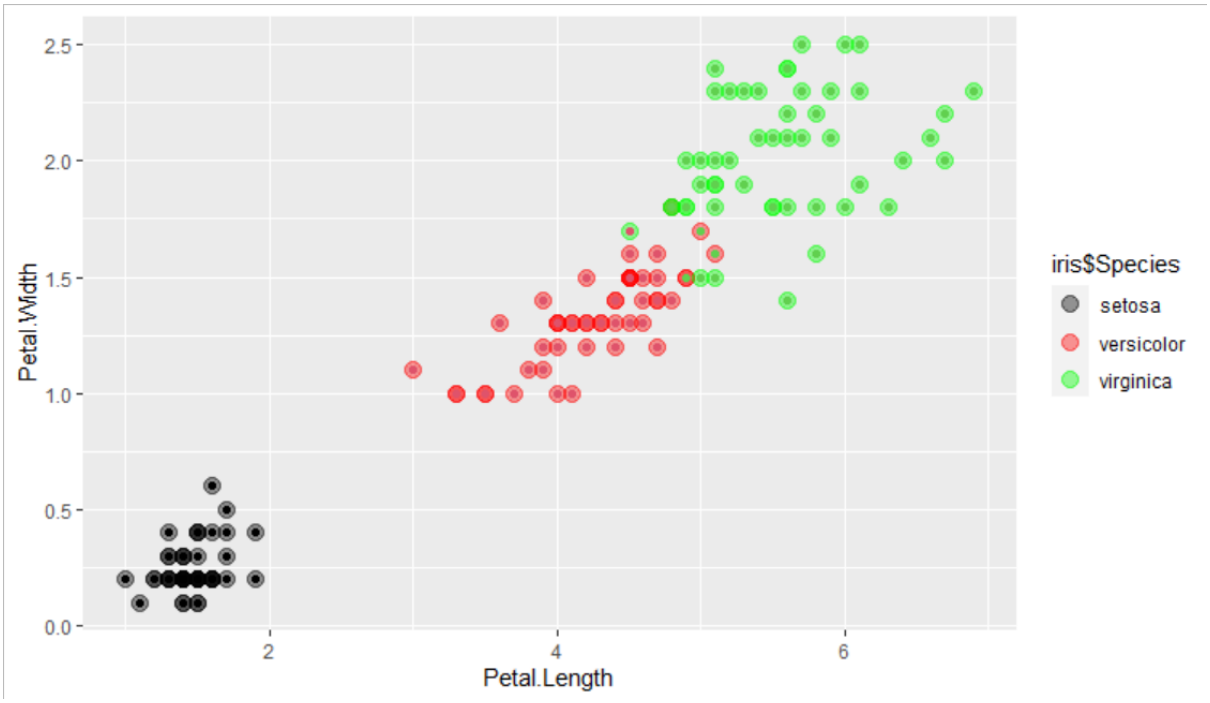
```
clusterCut <- cutree(clusters, 3)
clusterCut
table(clusterCut,iris$Species)
```

```
> clusterCut
  [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
 [46] 1 1 1 1 1 2 2 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 2 3 2 2 2 2 3 2 2 2 2 2 3 2 2 2 2 2 2
 [91] 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 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 3 3 3
[136] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
> table(clusterCut,iris$Species)

clusterCut setosa versicolor virginica
         1     50          0         0
         2      0         45         1
         3      0          5        49
```



PLOTTING CLUSTERD GRAPH:

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



CONCLUSION:

From this practical, I have learned how to implement k – means & aggolomerative clustering in R.