Introduction

Data Science and Machine Learning are furtive, they go un-noticed but are present in all ways possible and everywhere. They contribute significantly in all the fields they are applied and leave us with evidence which we can rely on and take data-driven directions. A very interesting area is an example of Data Science and Machine Learning is 'Crimes'. To focus on types of crimes taken place across 50 states in the USA and cluster them for the following reasons:

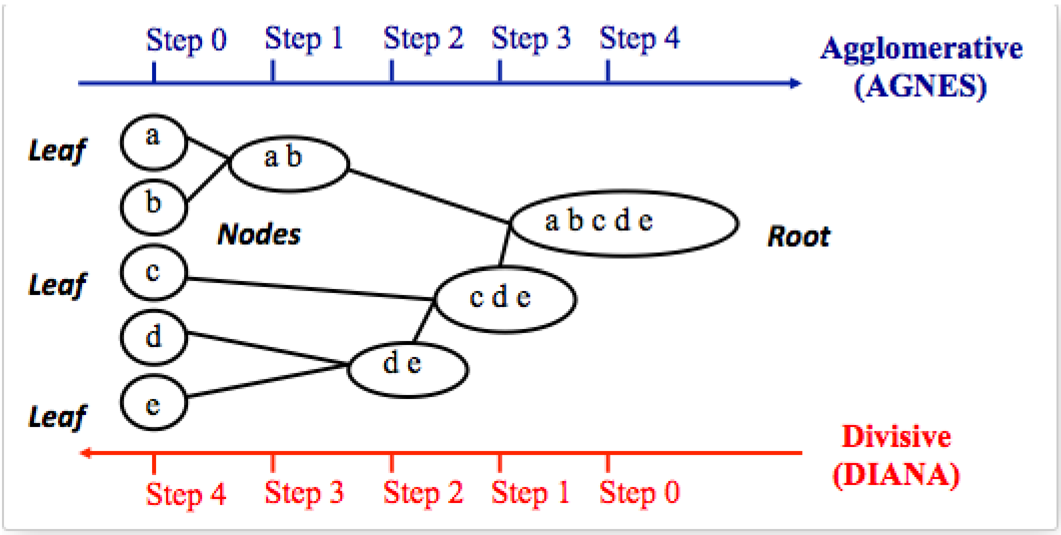
1. To understand state-wise crime demographics
2. To make laws applicable in states depending upon the type of crime taking place most often
3. Getting police and forces ready by the type of crime done in respective states
4. Predicting the crimes that may happen and thus taking measures in advance

The above are the few applications which can be considered but they are not exhaustive, depending upon the data reports produced by the algorithms there can be more such applications which can be deployed. Hierarchical clustering can be implemented in two ways i.e. DIANA and AGNES for the USA Arrests data. Will be focusing on AGNES.

Algorithm

AGNES (Agglomerative) clustering works in a "bottom-up" manner. That is, each object is initially considered as a single-element cluster (leaf). At each step of the algorithm, the two clusters that are the most similar are combined into a new bigger cluster (nodes). This procedure is iterated until all points are member of just one single big cluster (root).

The inverse of agglomerative clustering is divisive clustering, which is also known as **DIANA** (Divise Analysis) and it works in a "top-down" manner. It begins with the root, in which all objects are included in a single cluster. At each step of iteration, the most heterogeneous cluster is divided into two. The process is iterated until all objects are in their own cluster.



Steps to agglomerative hierarchical clustering

1. Preparing the data
2. Computing (dis)similarity information between every pair of objects in the data set.
3. Using linkage function to group objects into hierarchical cluster tree, based on the distance information generated at step 1. Objects/clusters that are in close proximity are linked together using the linkage function.
4. Determining where to cut the hierarchical tree into clusters. This creates a partition of the data.

Data Structure and Preparation

The data should be a numeric matrix with rows representing observations (individuals) and columns representing variables.

```
df <- scale(USArrests)        #standardize the data
head(df, nrow=6)              #show the first 6 rows
## Murder Assault UrbanPop Rape
## Alabama 1.2426 0.783 -0.521 -0.00342
## Alaska 0.5079 1.107 -1.212 2.48420
## Arizona 0.0716 1.479 0.999 1.04288
## Arkansas 0.2323 0.231 -1.074 -0.18492
## California 0.2783 1.263 1.759 2.06782
## Colorado 0.0257 0.399 0.861 1.86497
```

Similarity Measures

In order to decide which objects/clusters should be combined or divided, need methods for measuring the (dis)similarity between objects like Euclidean and Manhattan distances. In R software, function dist() is used to compute the distance between every pair of object in a data set. The results of this computation is known as a distance or dissimilarity matrix.

```
# Compute the dissimilarity matrix
# df = the standardized data
res.dist <- dist(df, method = "euclidean")
```

Linkage

The linkage function takes the distance information, returned by the function dist(), and groups pairs of objects into clusters based on their similarity. Next, these newly formed clusters are linked to each other to create bigger clusters. This process is iterated until all the objects in the original data set are linked together in a hierarchical tree. hclust() can be used to create a hierarchical tree.

```
res.hc <- hclust(d = res.dist, method = "ward.D2")
```

- d: a dissimilarity structure as produced by the dist() function.
- method: the agglomeration (linkage) method to be used for computing distance between clusters. Allowed values is one of “ward.D”, “ward.D2”, “single”, “complete”, “average”, “mcquitty”, “median” or “centroid”.

The most common linkage method:

- Maximum or complete linkage: The distance between two clusters is defined as the maximum value of all pairwise distances between the elements in cluster 1 and the elements in cluster 2. It tends to produce more compact clusters.
- Minimum or single linkage: The distance between two clusters is defined as the minimum value of all pairwise distances between the elements in cluster 1 and the elements in cluster 2. It tends to produce long, “loose” clusters.
- Mean or average linkage: The distance between two clusters is defined as the average distance between the elements in cluster 1 and the elements in cluster 2.
- Centroid linkage: The distance between two clusters is defined as the distance between the centroid for cluster 1 (a mean vector of length p variables) and the centroid for cluster 2.
- Ward’s minimum variance method: It minimizes the total within-cluster variance. At each step the pair of clusters with minimum between-cluster distance are merged.

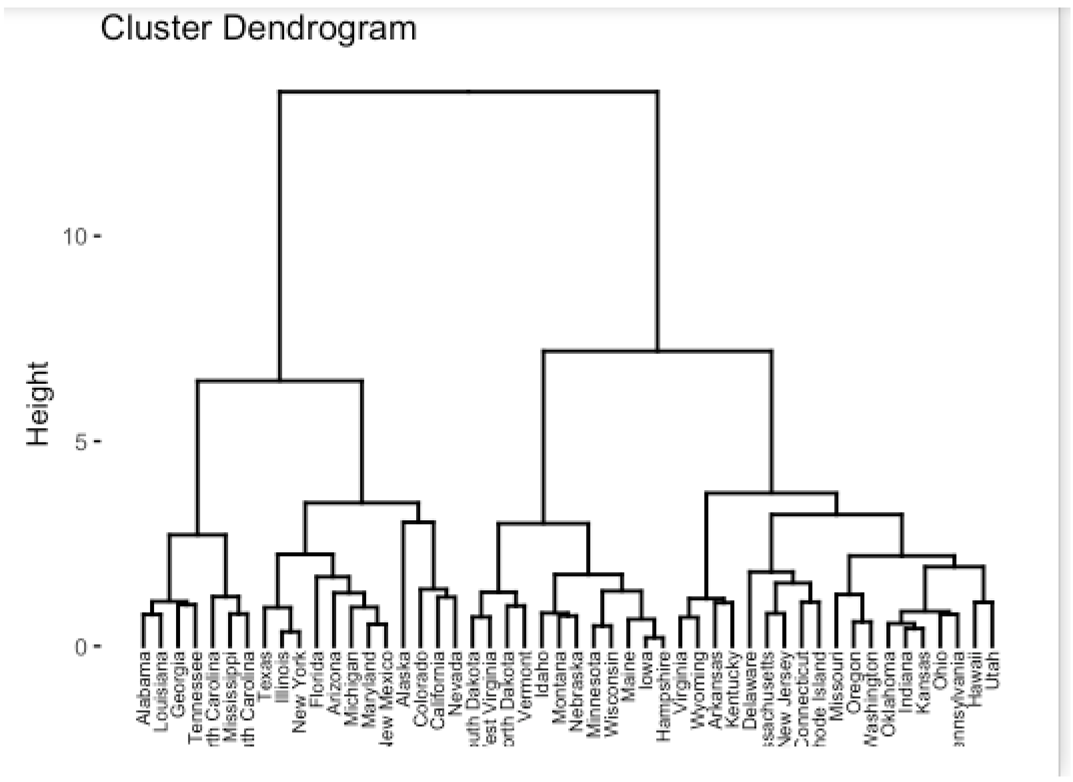
Dendogram

Dendrograms correspond to the graphical representation of the hierarchical tree generated by the function hclust(). Dendrogram can be produced in R using the base function plot(res.hc), where res.hc is the output of hclust(). Function fviz_dend()[in factoextra R package] is used to produce a beautiful dendrogram.

First install factoextra by typing this: install.packages(“factoextra”); next visualize the dendrogram as follow:

```
library("factoextra")

fviz_dend(res.hc, cex=0.5)                              #label size
```



In the dendrogram displayed above, each leaf corresponds to one object. As we move up the tree, objects that are similar to each other are combined into branches, which are themselves fused at a higher height.

The height of the fusion, provided on the vertical axis, indicates the (dis)similarity/distance between two objects/clusters. The higher the height of the fusion, the less similar the objects are. This height is known as the cophenetic distance between the two objects.

Verify the Cluster Tree

After linking the objects in a data set into a hierarchical cluster tree, one might want to assess that the distances (i.e., heights) in the tree reflect the original distances accurately. One way to measure how well the cluster tree generated by the *hclust*() function reflects your data is to compute the correlation between the *cophenetic* distances and the original distance data generated by the *dist*() function. If the clustering is valid, the linking of objects in the cluster tree should have a strong correlation with the distances between objects in the original distance matrix.

The closer the value of the correlation coefficient is to 1, the more accurately the clustering solution reflects your data. Values above 0.75 are felt to be good. The “average” linkage method appears to produce high values of this statistic. This may be one reason that it is so popular.

The R base function cophenetic() can be used to compute the cophenetic distances for hierarchical clustering.

```
res.coph <- cophenetic(res.hc)                    #compute cophentic distance

cor(res.dist, res.coph) outputs 0.698             #correlation between cophenetic distance and the original distance
```

Execute the *hclust*() function again using the average linkage method. Next, call cophenetic() to evaluate the clustering solution.

```
res.hc2 <- hclust(res.dist, method = "average")
cor(res.dist, cophenetic(res.hc2))                          #outputs 0.718
```

The correlation coefficient shows that using a different linkage method creates a tree that represents the original distances slightly better.

Cut the dendogram into different groups

One of the problems with hierarchical clustering is that, it does not tell us how many clusters there are, or where to cut the dendrogram to form clusters. One can cut the hierarchical tree at a given height in order to partition your data into clusters. The R base function cutree() can be used to cut a tree, generated by the hclust() function, into several groups either by specifying the desired number of groups or the cut height. It returns a vector containing the cluster number of each observation.

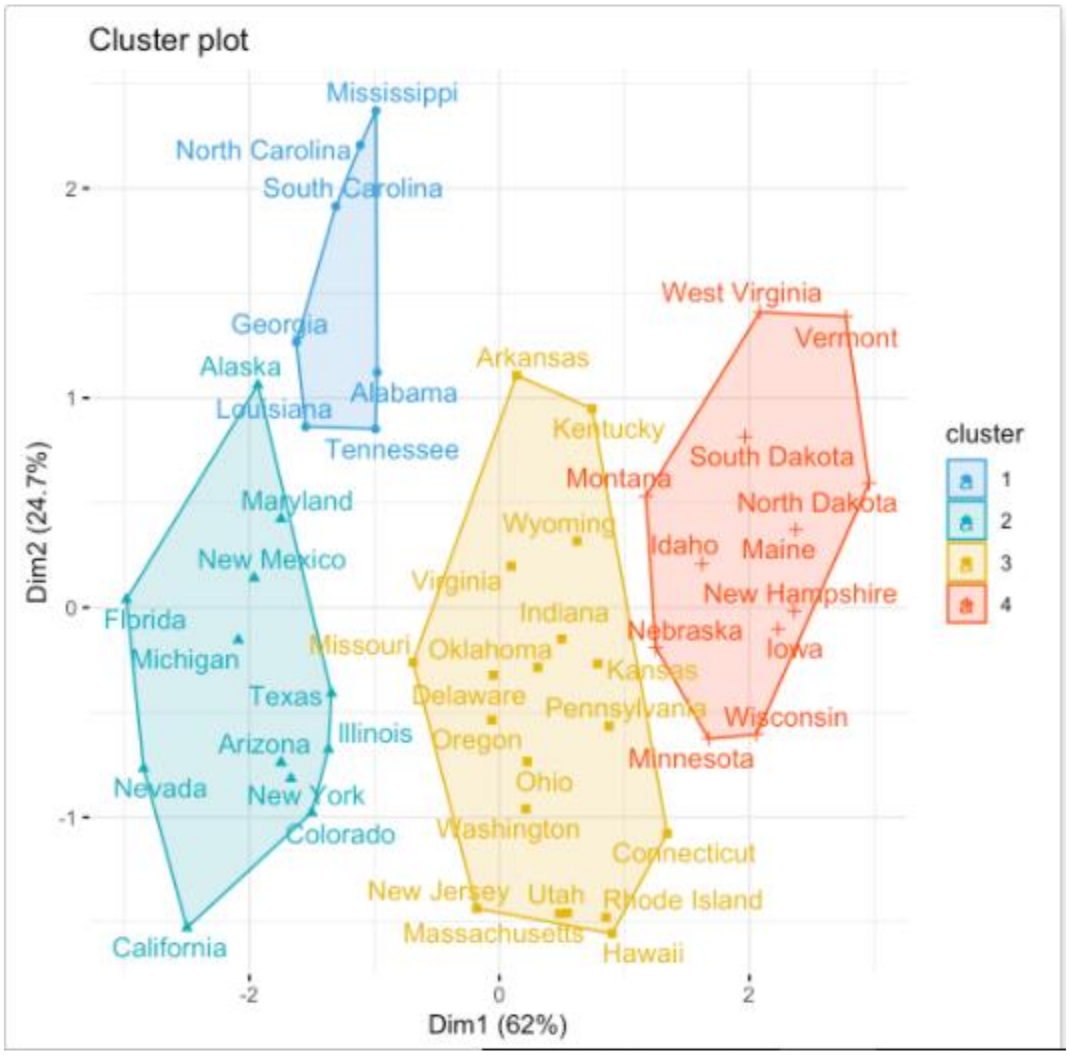
```
grp <- cutree(res.hc, k = 4) head(grp, n = 4)   # Cut tree into 4 groups

## Alabama Alaska Arizona Arkansas
## 1 2 2 3

table(grp)                                      # Number of members in each cluster
## 1 2 3 4
## 7 12 19 12

rownames(df)[grp == 1]                          # Get the names for the members of cluster 1
## [1] "Alabama" "Georgia" "Louisiana" "Mississippi"
## [5] "North Carolina" "South Carolina" "Tennessee"
```

Using the function *fviz_cluster*() [in *factoextra*], can visualize the result in a scatter plot. Observations are represented by points in the plot, using principal components.



Cluster R Package

The R package *cluster* makes it easy to perform cluster analysis in R. It provides the function *agnes*() and *diana*() for computing agglomerative and divisive clustering, respectively.

library("cluster")

# Agglomerative Nesting (Hierarchical Clustering)

res.agnes <- agnes(x = USArrests, stand = TRUE, metric = "euclidean", method = "ward")

After running agnes(), use the function *fviz_dend*()[in *factoextra*] to visualize the output:

fviz_dend(res.agnes, cex = 0.6, k = 4)

Another application of hierarchical clustering to gene expression data analysis

In gene expression data analysis, clustering is generally used as one of the first step to explore the data. We are interested in whether there are groups of genes or groups of samples that have similar gene expression patterns. Several clustering distance measures can be used to decide which items have to be grouped together or not. These measures can be used to cluster genes or samples that are similar.
For most common clustering software, the default distance measure is the Euclidean distance. The most popular methods for gene expression data are to use log2(expression + 0.25), correlation distance and complete linkage agglomerative clustering.

Single and Complete linkage give the same dendrogram using raw data, the log of the data or any other transformation of the data that preserves the order because what matters is having the smallest distance. The other methods are sensitive to the measurement scale.

In principle it is possible to cluster all the genes, although visualizing a huge dendrogram might be problematic. Usually, some type of preliminary analysis, such as differential expression analysis is used to select genes for clustering. Selecting genes based on differential expression analysis removes genes which are likely to have only chance patterns. This should enhance the patterns found in the gene clusters.

Conclusion

Clustering is an unsupervised machine learning algorithm in which we compute analytics mostly without a pre-defined aim to understand the relationships between the data. Once we get the understanding and trends in the data we can accordingly take necessary actions and data-driven decisions. In this case, viewing the states in cluster 1 there can be more strict law enforcement in the cluster 1 states to reduce the number of murders, assaults or rape cases. Police and officials can be more equipped and trained for such situations as well.

Hierarchical clustering is a cluster analysis method, which produces a tree-based representation (i.e.: dendrogram) of a data. Objects in the dendrogram are linked together based on their similarity.

To perform hierarchical cluster analysis in R, the first step is to calculate the pairwise distance matrix using the function dist(). Next, the result of this computation is used by the hclust() function to produce the hierarchical tree. Finally, you can use the function fviz_dend() [in factoextra R package] to plot easily a beautiful dendrogram.

It's also possible to cut the tree at a given height for partitioning the data into multiple groups (R function cutree()).

Citations:

https://www.datasciencecentral.com/profiles/blogs/usarrests-hierarchical-clustering-using-diana-and-agnes

http://hanj.cs.illinois.edu/cs412/bk3/10.pdf