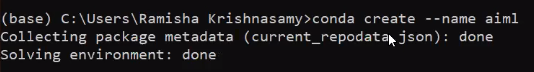
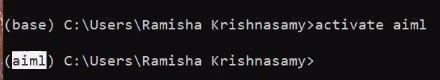
# CREATING VIRTUAL ENVIRONMENT

**Eg. Name “aiml”**



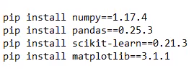


# INSTALLING THE PYTHON

**“python=3.10”**



# INSTALLING LIBRARIES

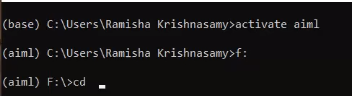




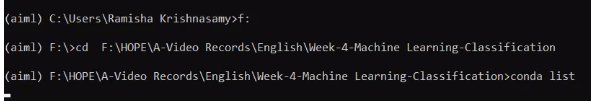
# INSTALLING JUPITER NOTEBOOK



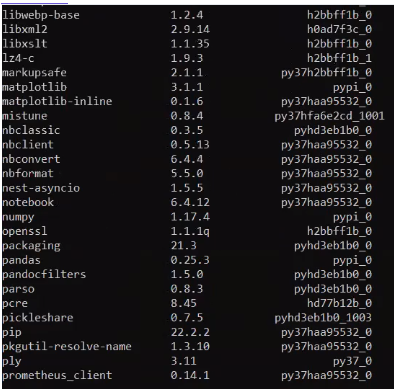
# “AIML”-WORKING DIRECTORIES



# CHECKING VERSION OF THE PROGRAM



**“conda list”**



# UN-SUPERVISING LEARNING

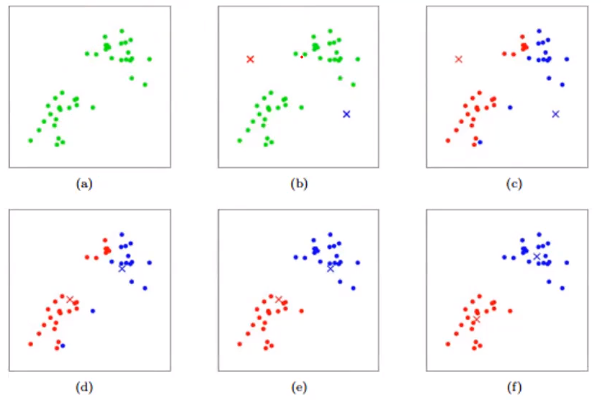
Un-supervising Learning has two algorithms:

K-Means (average)

Hierarchical

* Two Cluster formed based on the MEAN
* Compare the distance of the Data
* It makes the Centroid and Clustering also positioning its place.

# K-Means Clustering

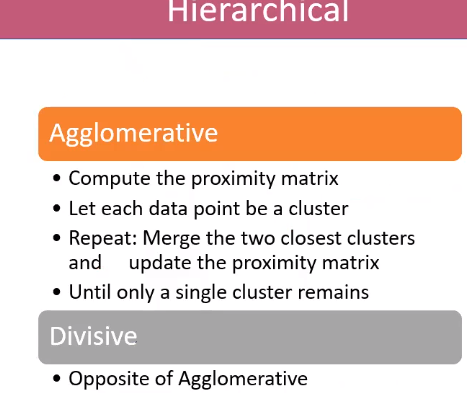


Positioning its place

# K-Means Multiple Clustering



# HIERARCHICAL



# **ALGORITHIMS**

**Points to Remember**:

No Output – Y (Dependent Variable/Un-supervise Learning)

Grouping or Clustering – X (Grouping)

Algorithm for locating the Column name while we are run with more than end number of columns

***X=dataset.iloc[:,[3,4]].values***

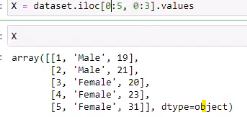


3-4 COLUMN

ROW



**Selection of COLUMN**



**Selection of COLUMN and ROW**

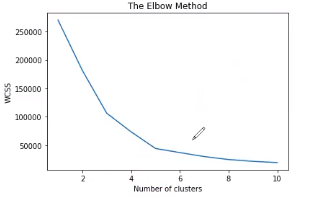
Positioning the Data



Inertia to pull the data for positioning

Call the List

11-cluster



Found the last interrupt and least distance Bend

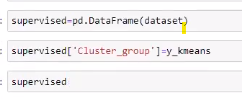
**5th Cluster**

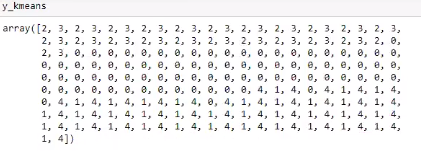
Fitting the Data: (Procedure)



No need Train Set and Test Set

Checking the category into Cluster

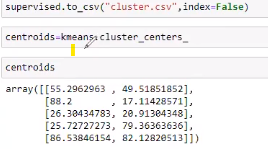
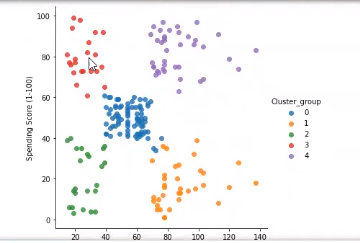






We find the cluster group as we know that which category

See the list: 5 categories (Centroid Value)

**Customer purchased 40% even their Annual Income falls 80-100$**

**Customer purchased 80% even their Annual Income falls 20-40$**

**Customer purchased 40% even their Annual Income falls 20-40$**

**Customer purchased 60% even their Annual Income falls 40-80$**

**Customer purchased 80% even their Annual Income falls 80-140$**

This helpful for us to get indirect data of (customer by spending money how the savings in-built)

# Importing the Model:

**Supervised column Label with Colour –5**

**Spending Score (1-100)**

**Annual Income (k$) – Column 3**

**Points to Remember**:



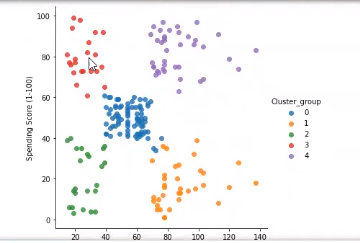
Install ‘Seaborn’

**Display Cluster Group inside the Graph**

**No Regression Fit—False**

**t**





Explanation of Coding:

* X = dataset.iloc[:, 3:5].values

: - Means All rows if need specific rows, e.g. 0:10

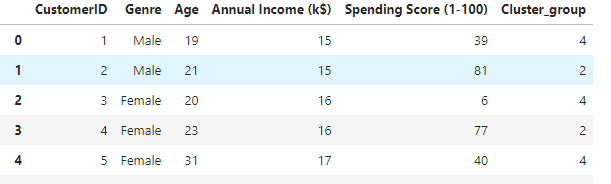
* from sklearn.cluster import KMeans
* kmeans = KMeans(n\_clusters = 5, init = 'k-means++', random\_state = 42)
* y\_kmeans = kmeans.fit\_predict(X)

“random\_state” mean “0” or “42”

random\_state is a parameter in train\_test\_split that controls the random number generator used to shuffle the data before splitting it. In other words, it ensures that the same randomization is used each time you run the code, resulting in the same splits of the data.

***If we just split data without shuffling, it will give good performance on the training dataset but poor performance on testing. For that, we need to do shuffling of data. So, if we use random\_state then we can avoid this problem***.

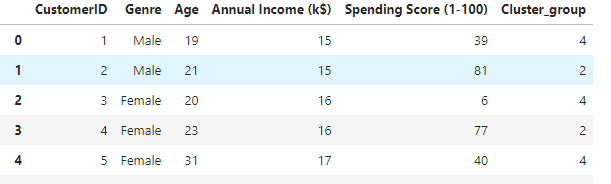
* supervised['Cluster\_group']=y\_kmeans



Creating the column with the name of “Cluster\_group”

* supervised.to\_csv("cluster.csv",index=False)

to save above table as csv in the name of “cluster.csv”



If we give index=false, which never re-write the index again and again

* dir(kmeans)

the clauses which used in kmeans function

* centroids=kmeans.cluster\_centers\_

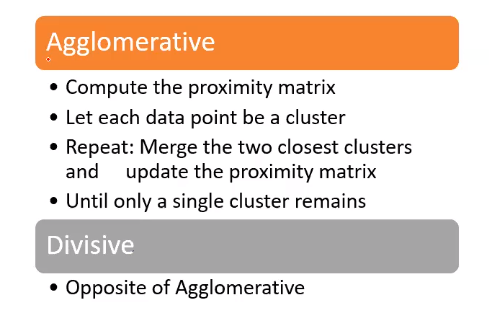
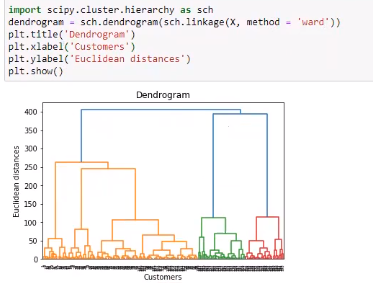
the cluster formed in centroid

# HIERARCHICAL CLUSTERING

The above clustering has two categories are:

Agglomerative

Divisive

ALGORITHM:

Import scipy.cluster.hierarchy as sch

The name of the graph is DENDOGRAM

(A dendrogram is a diagram that shows the hierarchical relationship between objects. It is most commonly created as an output from hierarchical clustering. The main use of a dendrogram is to work out the best way to allocate objects to clusters)

Plt.tittle(‘Dendrogram’)

Here, we have to mention the cluster

