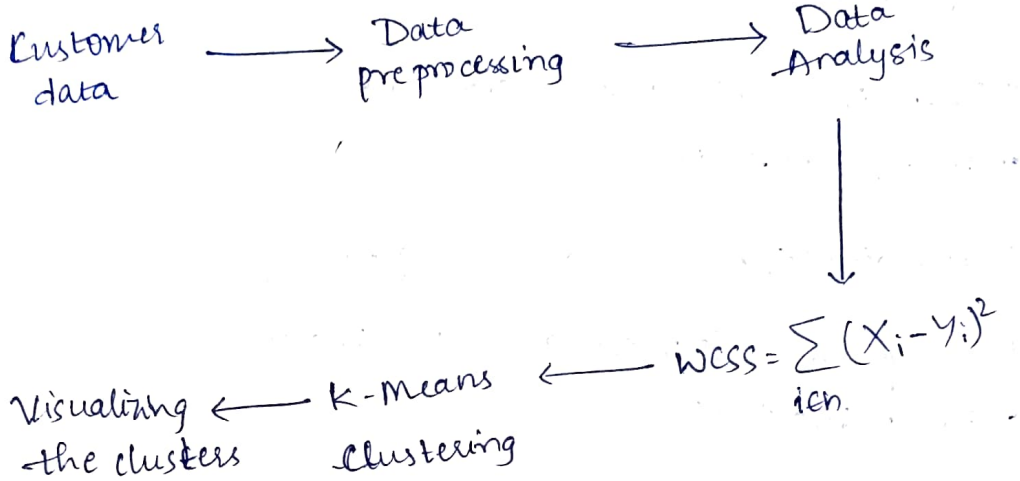
Customer Segmentation using K-means Clustering with Python.

- Workflow:-



Code:-

* Importing the Dependencies (required libraries & functions)

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.cluster import KMeans
```

numpy - A way to handle matrices and vectors in python

pandas - Pandas is used for making dataframes i.e structured table.

matplotlib & seaborn - These are the data visualization libraries

sklearn.cluster - From sklearn library in cluster module we import Kmeans algorithm.

* Data Collection & Analysis

```
#loading the data from csv
customers_data = pd.read_csv('/content/Mall_Customers.csv')
```

```
#first 5 rows in the dataframe
customers_data.head()
```

head() - It is used to give first 5 rows in dataframe.

```
#finding the number of rows and columns
customer_data.shape
```

Shape = gives the total no of rows & columns.

```
#getting some information about the dataset
customers_data.info()
```

```
#checking for missing values
customers_data.isnull().sum()
```

* Choosing the annual income column & spending score column.

```
X = customers_data.iloc[:,[3,4]].values
print(X)
```

iloc[:,[3,4]].values = Used to choose the 3rd & 4th columns from the dataframe.
":" is used to specify the we need 3rd & 4th "columns" other it will consider rows.

* Choosing the number of cluster

```
#finding wcss values for different number of clusters
wcss = [ ]
for i in range(1, 11):
    kmeans = KMeans(n_clusters = i, init = 'k-means++', random_state = 42)
    kmeans.fit(X)
    wcss.append(kmeans.inertia_)
```

range(1,11) = Here it checks for 10 clusters.

kmeans = variable created to load KMeans cluster function.

n_clusters = Represents the num of clusters we want.

kmeans.fit(X) = Here we are fitting our data to kmeans.

kmeans.inertia = Gives us the wcss value and that value will be stored in the empty list

wcss (within Clusters Sum Of Squares)

```
#plot an elbow graph
sns.set()
plt.plot(range(1,11),wcss)
plt.title('The Elbow Point Graph')
plt.xlabel('Number of Clusters')
plt.ylabel('WCSS')
plt.show()
```

- By running this code we will get an elbow graph where we can have elbow points. We have to select a point so that after that point there should no be any sharp significant drop.
- By this we can say correct optimum number of clusters

* Training the k-means clustering model.

```
kmeans = KMeans(n_clusters = 5, init = 'k-means++', random_state = 42)
# returning a label for each datapoint based on their cluster
Y = kmeans.fit_predict(X)
print(Y)
```

* Visualizing all the clusters

```
#plotting all the clusters and their centroids
plt.figure(figsize = (8,8))
plt.scatter(X[Y==0,0], X[Y==0,1], s=50, c='green', label='Cluster1')
plt.scatter(X[Y==1,0], X[Y==1,1], s=50, c='red', label='Cluster2')
plt.scatter(X[Y==2,0], X[Y==2,1], s=50, c='yellow', label='Cluster3')
plt.scatter(X[Y==3,0], X[Y==3,1], s=50, c='blue', label='Cluster4')
plt.scatter(X[Y==4,0], X[Y==4,1], s=50, c='violet', label='Cluster5')

#plot the centroids
plt.scatter(kmeans.cluster_centers_[:,0], kmeans.cluster_centers_[:,1], s=100,
            c='cyan', label='Centroids')

plt.title('Customer Groups')
plt.xlabel('Annual Income')
plt.ylabel('Spending Score')
plt.show()
```

```
plt.scatter(X[Y==0,0], X[Y==0,1])
```

- `X[Y==0,0]` — This indicates the cluster I
- `X[Y==0,0]` — This is the index value of annual income
- `X[Y==0,1]` — This is the index value of spending score
- `s=50` → indicates the size of the datapoints in the cluster
- `s=100` → indicates the size of the centroids in each cluster.

Conclusion:

- this the process to make clusters (groups among people) and make better recommendations for them.
- For example: Consider a movies ~~watching~~ streaming platform like netflix. Some people want to watch thriller and some other want horror. ~~So~~ when a ~~new~~ person watches a horror movie, it automatically recommend some mor horror movies.
- By this method we can give better recommendations for the people.