K-means clustering Steps

- **Importing the libraries**
- **Importing the dataset**
- **Using the elbow method to find the optimal number of clusters**
- **Training the K-Means model on the dataset**
- **Visualising the clusters**

What is meant by the term ‘random-state’ in 'KMeans'

- Basic k-means algorithm is not optimal. That means, it is not sure to find the best solution
- You may be stuck into local minima, and hence the result of your algorithm depends of your initialization (of your centroids).

- A good practice in order to find a good minimum is to rerun the algorithm several times with several initializations and keep the best result.

Bear in mind that the KMeans function is stochastic (the results may vary even if you run the function with the same inputs' values).

Hence, in order to make the results reproducible, you can specify a value for the random_state parameter.

Random state in Kmeans function of sklearn mainly helps to

1. Start with same random data point as centroid if you use Kmeans++ for initializing centroids.

2. Start with same K random data points as centroid if you use random initialization.
This helps when one wants to reproduce results at some later point in time.

We shall Use an int to make the randomness deterministic.

Using the elbow method to find the optimal number of clusters

```
from sklearn.cluster import KMeans
wcss = []
for i in range(1, 11):
  kmeans = KMeans(n_clusters = i, init = 'k-means++', random_state = 1)
  kmeans.fit(X)
 wcss.append(kmeans.inertia_)# appending wcss value
```

- #wcss is a empty list, we are going to add values in to this list one by one
- # range starts with 1 and ends at 10, upper bound is excluded, lower bound is included. we try number of cluster from 1 to 10
- #initialization is k-menas++ which gives a better initailiziation of centroid and avoids random initialization trap
- # training k-means algorithm using input features as in X

inertia_ **(**float)

It is a Sum of squared distances of samples to their closest cluster center, weighted by the sample
weights if provided.

- Fit_predict()- This method Compute cluster centers and predict cluster index for each sample.
- Fit()- Compute k-means clustering.

Hierarchical clustering Steps

- **Importing the libraries**
- **Importing the dataset**
- **Using the dendrogram to find the optimal number of clusters**
- **Training the Hierarchical clustering model on the dataset**
- **Visualising the clusters**

Using the dendrogram to find the optimal number of clusters

- import scipy.cluster.hierarchy as sch

Or from scipy.cluster import hierarchy as sch

- dendrogram = sch.dendrogram(sch.linkage(X, method = 'ward’))

- Here dendrogram is a function in sch
- Linkage is another function with two arguments- first matrix feature in which u want to find cluster and 2nd argument is clustering technique
- In hierarchical clustering, the most recommended and one that produces most relevant results is the method of minimum variance
- method=ward, is the method of minimum variance, this minimize the variance in clustering

- from sklearn.cluster import AgglomerativeClustering
- hc = AgglomerativeClustering(n_clusters = 5, affinity = 'euclidean', linkage = 'ward')
- y_hc = hc.fit_predict(X)