Clustering

Clustering is an **Unsupervised Machine Learning** technique used to group data points based on their similarity, such that data points in the same group (cluster) are more similar to each other than to those in other groups.

1. K-Means Clustering

Definition:

Partitions the dataset into K distinct clusters by minimizing the distance between points and their cluster centers (centroids), aiming to minimize the Within-Cluster Sum of Squares (WCSS).

Best for:

Large datasets with clearly separable spherical clusters.

Advantages:

Fast and simple to implement.

Works well for large datasets.

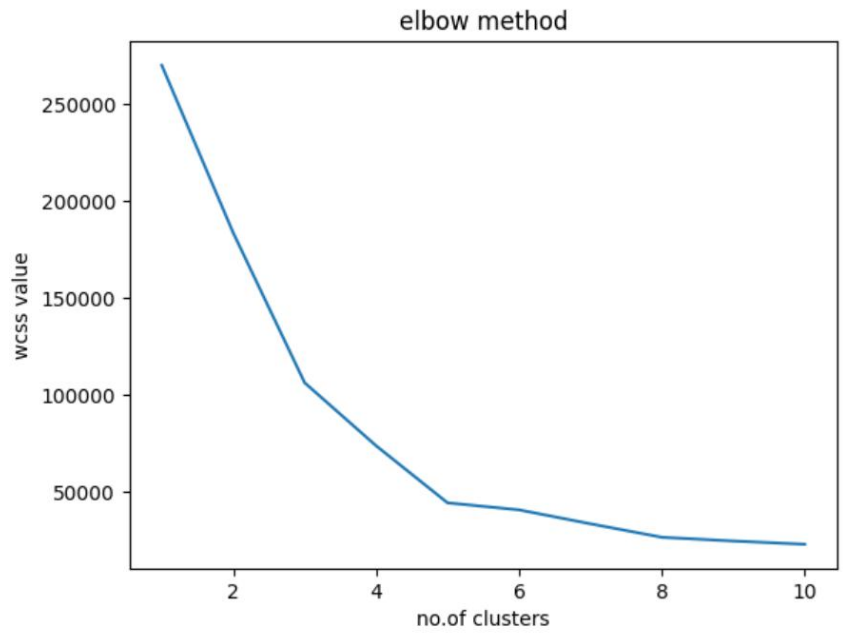
Limitations:

Must choose K (number of clusters) beforehand; this is often done using the Elbow Method.

Fails for non-spherical clusters.

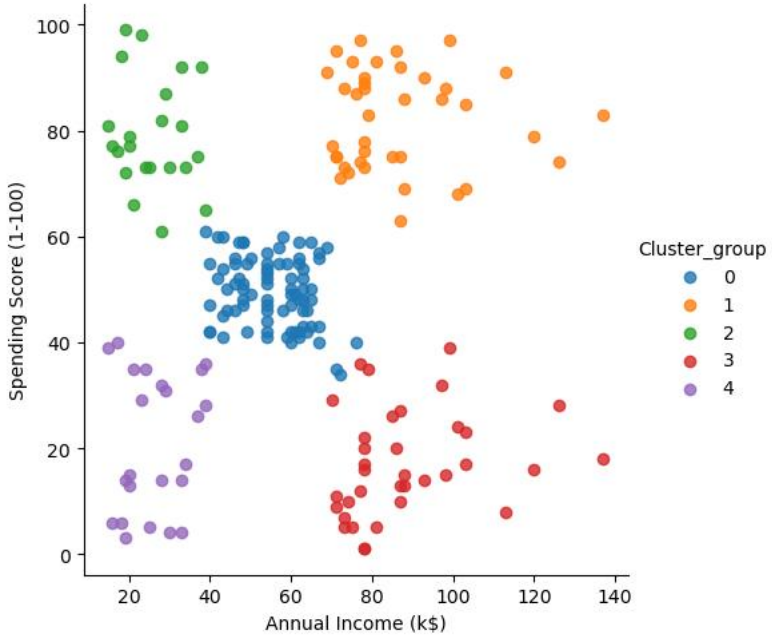
Sensitive to noise and outliers.

```
from sklearn.cluster import KMeans
list1=[]
for i in range(1,11):
    kmeans=KMeans(n_clusters=i, init='k-means++', random_state=42)
    kmeans.fit(X)
    list1.append(kmeans.inertia_)
plt.plot(range(1,11),list1)
plt.title('elbow method')
plt.xlabel('no.of clusters')
plt.ylabel('wcss value')
plt.show()
```




```
from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters = 5, init = 'k-means++', random_state = 42)
y_kmeans = kmeans.fit_predict(X)
```

```
import seaborn as sns
facet_kws = sns.lmplot(data=supervised, x=supervised.columns[3], y=supervised.columns[4],
                       hue=supervised.columns[5], fit_reg=False,
                       legend=True, facet_kws={"legend_out": True})
```



2. Hierarchical Agglomerative Clustering

Definition:

Builds a hierarchy of clusters by merging small clusters into larger ones (bottom-up approach). It uses a linkage criterion to measure the distance between existing clusters.

Best for:

Small to medium-sized datasets where visualizing the cluster relationship (hierarchy) is important.

Advantages:

No need to predefine K upfront.

The visual Dendrogram helps in finding the optimal cluster count.

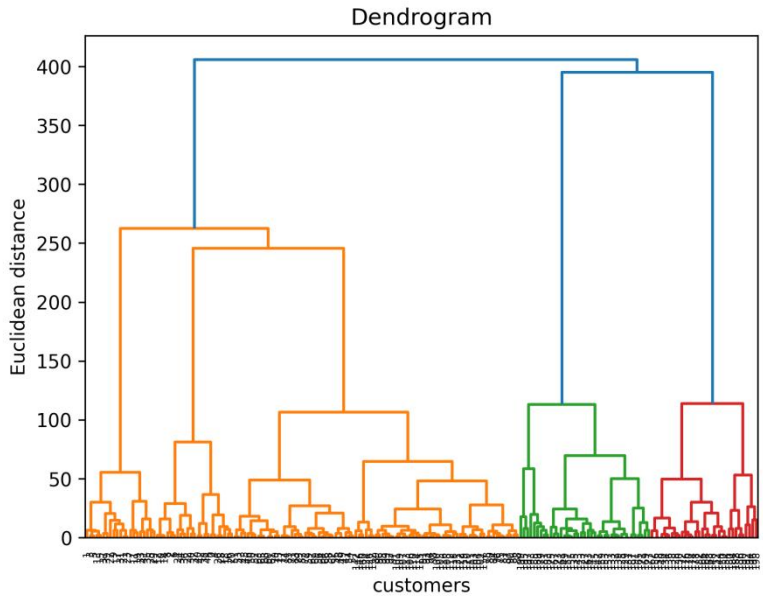
Limitations:

Slow for large datasets (high computational complexity).

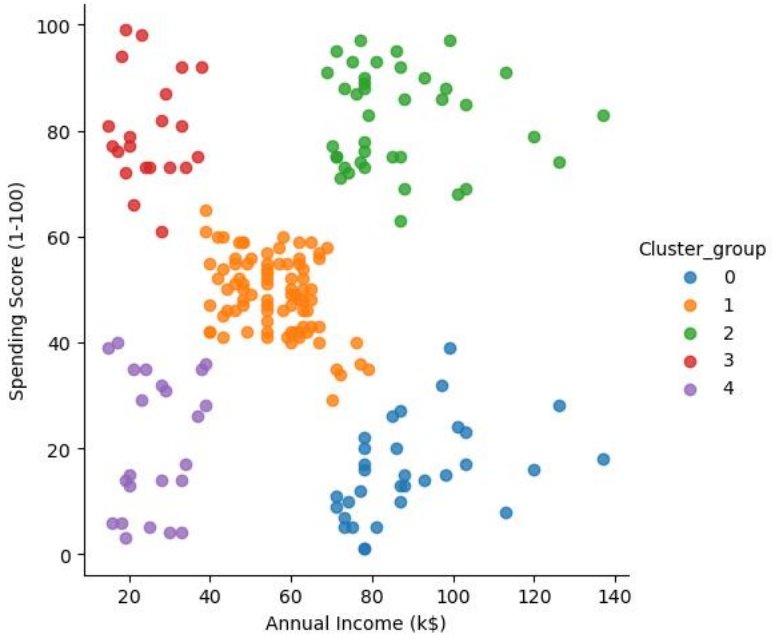
Once a merge is done, it cannot be undone.

Sensitive to noise and outliers.

```
import scipy.cluster.hierarchy as sch
dendrogram=sch.dendrogram(sch.linkage(X, method='ward'))
plt.title('Dendrogram')
plt.xlabel('customers')
plt.ylabel('Euclidean distance')
plt.savefig("dendrogram.png", dpi=300, bbox_inches='tight')
plt.show()
```



```
from sklearn.cluster import AgglomerativeClustering
clusmodel = AgglomerativeClustering(n_clusters = 5, metric='euclidean')
label = clusmodel.fit_predict(X)
```



3. Affinity Propagation Clustering

Definition:

Finds representative data points called Exemplars by exchanging "messages" (responsibility and availability) between all data points until a consensus is reached.

Best for:

Datasets where the number of clusters is unknown and for finding representatives/summaries of a large set of items.

Advantages:

Does not require specifying K upfront.

Finds a set of representative data points (exemplars).

Limitations:

High computational complexity (slow) for large datasets.

Sensitive to the initial values for the "preference" parameter.

```
from sklearn.cluster import AffinityPropagation
clusmodel=AffinityPropagation(random_state=42)
label=clusmodel.fit_predict(X)
```

```
import seaborn as sns
facet_kws = sns.lmplot(data=supervised, x=supervised.columns[3],
                       y=supervised.columns[4], hue=supervised.columns[5],
                       fit_reg=False, legend=True, height=5, aspect=1,
                       facet_kws={"legend_out": True})
```

4. Mean Shift Clustering

Definition:

A non-parametric, density-based technique that shifts cluster centers toward the Mode (peak of density) of data points within a defined radius (bandwidth).

Best for:

Datasets with irregularly shaped clusters and varying densities; does not require specifying the number of clusters.

Advantages:

Does not require specifying K upfront.

Finds non-spherical clusters.

Less sensitive to outliers than K-Means.

Limitations:

Selecting the optimal bandwidth parameter is non-trivial.

Computationally expensive.

```
from sklearn.cluster import MeanShift, estimate_bandwidth
bandwidth = estimate_bandwidth(X, quantile=0.2)
clusmodel = MeanShift(bandwidth=bandwidth)
label = clusmodel.fit_predict(X)
```

5. DBSCAN Clustering

Definition:

Groups closely packed points into clusters based on ϵ (radius) and $MinPts$ (minimum points). Points in low density areas are marked as Noise/Outliers.

-1 for noise/Outliers.

Best for:

Identifying clusters of arbitrary shape and effectively detecting outliers.

Advantages:

Does not require specifying K.

Can find non-spherical, complexly shaped clusters.

Robust to noise and outliers.

Limitations:

Struggles with datasets of widely varying density.

Performance degrades rapidly with high dimensionality.

```
from sklearn.cluster import DBSCAN
clusmodel=DBSCAN(eps=11, min_samples=10)
label=clusmodel.fit_predict(X)
```

6. OPTICS Clustering

Definition:

An extension of DBSCAN that creates an ordering of points (using Reachability-distance) to reflect the density structure, making it effective for identifying clusters of varying density.

Best for:

Datasets where clusters have significantly different densities; allows exploration of cluster hierarchy.

Advantages:

Handles clusters of varying density well.

Does not require a fixed ϵ for density definition.

Identifies noise/outliers.

Limitations:

High memory requirement for storing the reachability ordering.

Result interpretation via the Reachability Plot can be complex.

```
from sklearn.cluster import OPTICS
clusmodel = OPTICS(min_samples=5, p=2, cluster_method='xi', xi=0.2)
label = clusmodel.fit_predict(X)
```

7. BIRCH Clustering

Definition:

Balanced Iterative Reducing and Clustering using Hierarchies. Designed for very large datasets by summarizing data into a small, compact structure (Clustering Feature (CF) Tree).

Best for:

Extremely Large Datasets where I/O costs are critical; works best on numerical data.

Advantages:

Highly scalable and fast (often linear complexity $O(N)$).

Reduces the amount of data needed for final clustering.

Limitations:

Only handles metric (numerical) data.

The initial summary phase may lose some information.

Performs poorly on non-spherical clusters.

```
from sklearn.cluster import Birch
clusmodel = Birch(threshold=1, branching_factor=100, n_clusters=5)
label = clusmodel.fit_predict(X)
```

For all algorithms same code is used to plot the graph

```
import seaborn as sns
facet = sns.lmplot(data=supervised, x=supervised.columns[3], y=supervised.columns[4],
                       hue=supervised.columns[5], fit_reg=False,
                       legend=True, facet_kws={"legend_out": True})
```