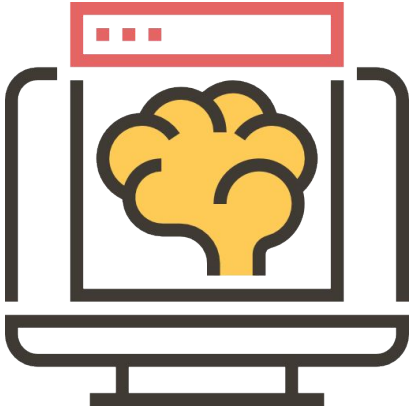
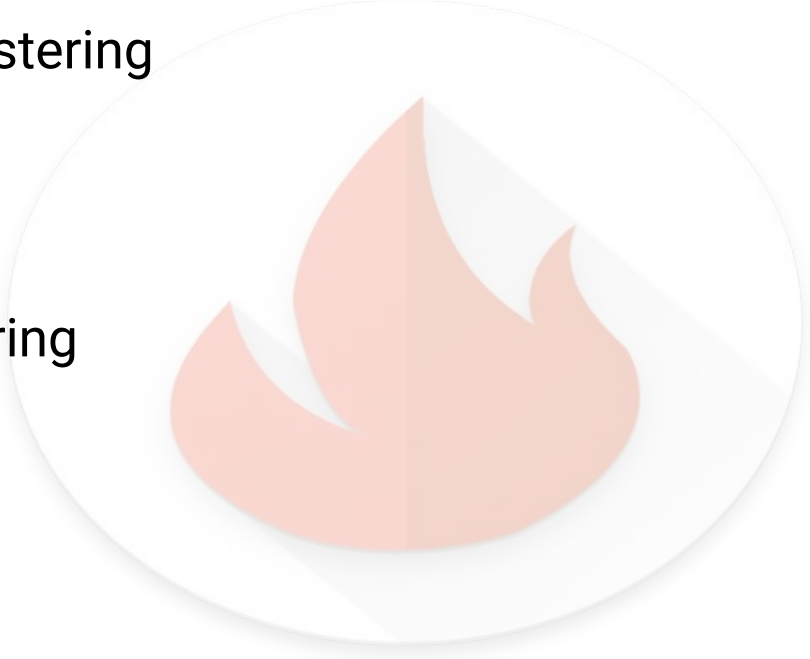


Unsupervised Learning

Index

- Introduction To Unsupervised learning
- Introduction To Clustering
- K Means Clustering
- Elbow Method
- Hierarchical Clustering



Index

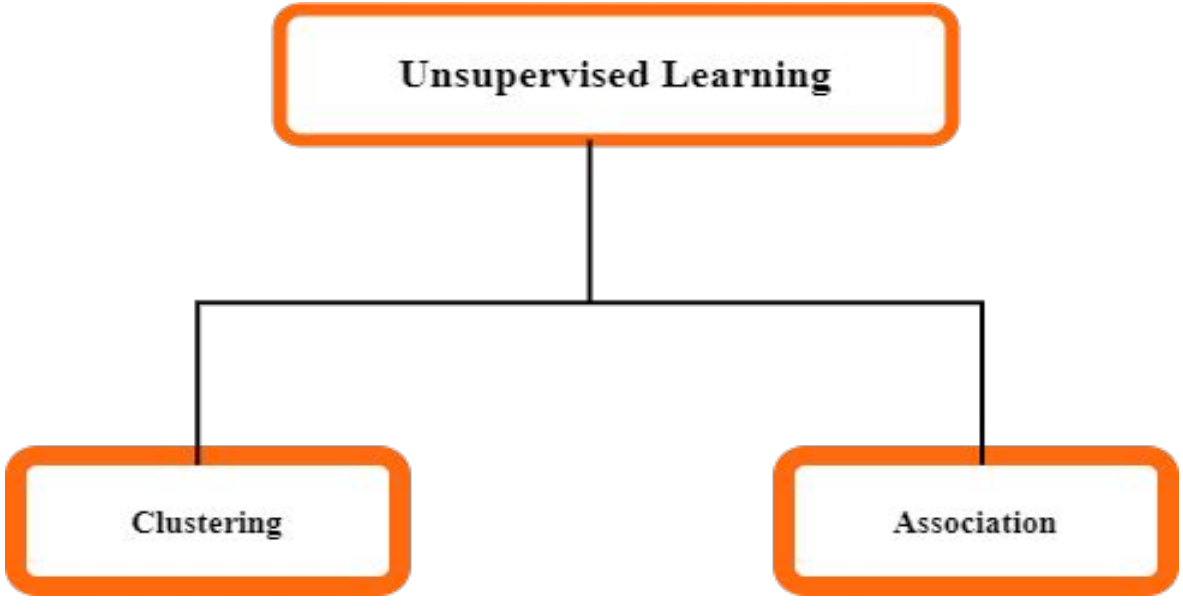
Introduction
To Unsupervised Learning

Introduction To Unsupervised learning

- Unsupervised learning is where you only have **input data (X)** and no corresponding output variables.
- In unsupervised learning, models itself find the hidden patterns and insights from the given data.
- Unsupervised learning **cannot** be directly applied to a **regression** or **classification** problem.

Introduction To Unsupervised learning

- Unsupervised learning problems can be further grouped into,



Introduction To Clustering

- Clustering is a method of grouping the objects into clusters such that objects with most **similarities** remains into a group and has less or no similarities with the objects of another group.
- For example, clustering viewers into similar groups based on their interests, age, geography, etc can be done by using Unsupervised Learning algorithms like K-Means Clustering.

Introduction To Clustering

- The clustering technique can be widely used in various tasks. Some most common uses of this technique are:
  - Statistical data analysis
  - Social network analysis
  - E-commerce Application

Introduction To Clustering

- The clustering technique can be widely used in various tasks. Some most common uses of this technique are:
  - Statistical data analysis
  - Social network analysis
  - E-commerce Application

Index

K Means Clustering

K-Means Clustering

- K-means clustering is one of the most widely used unsupervised machine learning algorithms that forms clusters of data based on the similarity between data instances.
- The **K** in the K-means refers to the number of clusters.

K-Means Clustering

- It is a **centroid-based** algorithm, where each cluster is associated with a centroid.
- It starts with K as the input which is how many clusters you want to find. Place K centroids in random locations in your space.
- Now, using the euclidean distance between data points and centroids, assign each data point to the cluster which is close to it.
- Recalculate the cluster centers as a mean of data points assigned to it.

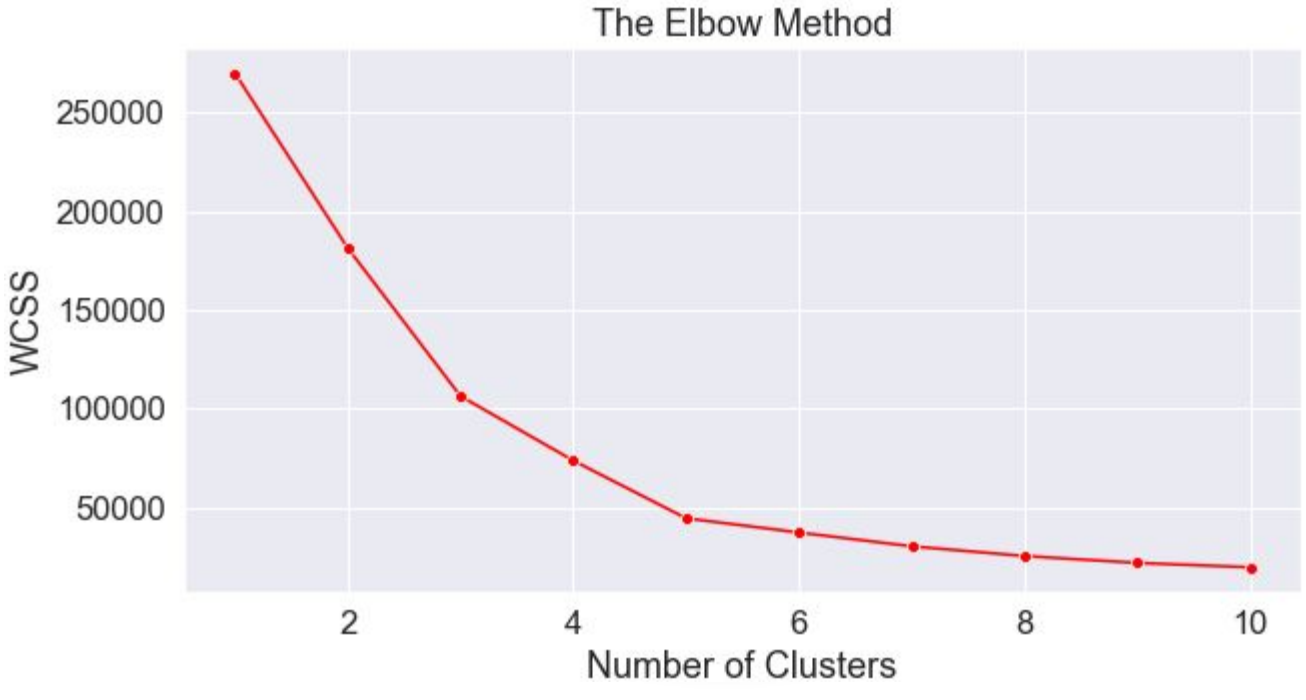
How decide the value of ‘K’

- One of the methods is called **“Elbow”** method can be used to decide an optimal number of clusters.
- This method uses the concept of WCSS value.
- WCSS stands for **Within Cluster Sum of Squares**, which defines the total variations within a cluster.

How decide the value of ‘K’

- The elbow method follows the below steps:
  - It executes the K-means clustering on a given dataset for different K values (ranges from 1-10).
  - For each value of K, calculates the WCSS value.
  - Plots a curve between calculated WCSS values and the number of clusters K.
  - The sharp point of bend or a point of the plot looks like an arm, then that point is considered as the best value of K.

How decide the value of ‘K’



Index

STEPS TO PERFORM K-MEANS CLUSTERING USING SCIKIT-LEARN

K-Means Clustering

- **Import Libraries :**

```
import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

%matplotlib inline
```

- **Load Dataset:**

```
df = pd.read_csv('../Sample.csv')
```

K-Means Clustering

- **Check Missing Values presence:**

  `df.isnull().sum()`

- **If Present Drop them:**

  `df.dropna(inplace = True)` Or

- Replace missing values with Appropriate values, either by Median or Mean

K-Means Clustering

- **Preparing the Data:**

Prepare the data and store into a Numpy array.

```
X = df.values[: , :]
```

K-Means Clustering

- **Training the Algorithm:** Import the KMeans class from sklearn.cluster library, instantiate it with passing number of clusters to make and call the fit() method along with our training data.

```
from sklearn.cluster import KMeans

model = KMeans (n_clusters = 2)

model.fit(X)
```

Here n_clusters = 2 indicates that 2 clusters will be form based upon similarities between data instances by calculating Euclidean Distances.

K-Means Clustering

- **Generating Final Cluster Centroids:**

  Generate Final Centroid Values of clusters

  print(model.cluster_centers_)

K-Means Clustering

- **Evaluating Clustering:**

Evaluation of Clustering is done by measuring the silhouette_score which is the mean silhouette coefficient of all observations. Silhouette coefficients range between -1 and 1 with 1 indicating dense value which means well separated clusters.

```
x_labels = model.labels_

print(silhouette_score(X, x_labels))
```

K-Means Clustering

- **Classification:** Predicting target class for test dataset from the trained modeled from the training dataset.
- **Clustering:** Using different similarity measure to place the all the similar items in a group.
  - Clustering is groping the similar kind of things by considering the most satisfied condition all the items in the same group should be similar and no two different group items should not be similar.
  - To group the similar kind of items in clustering, different similarity measures could be used.

Elbow Method

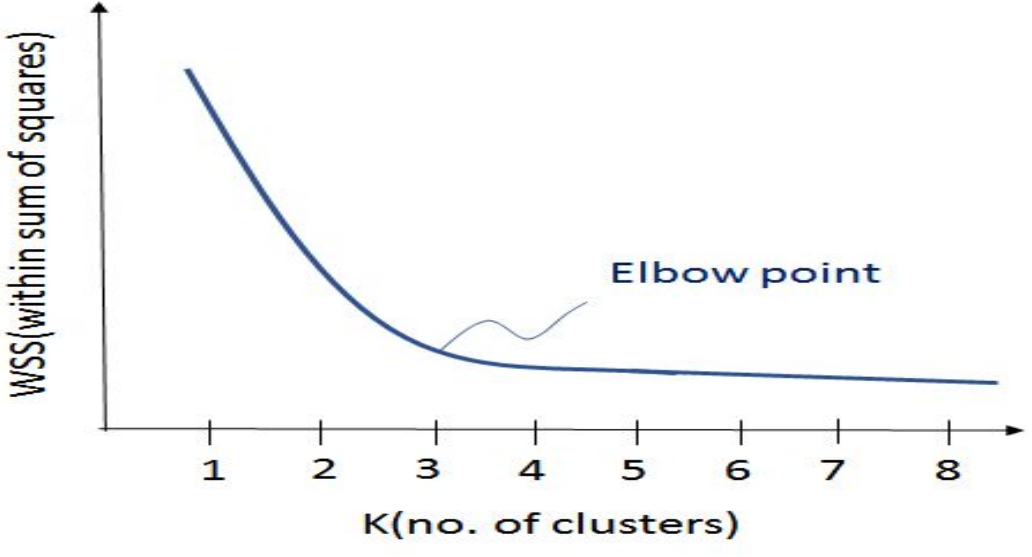
- **This is probably the most well-known method for determining the optimal number of clusters.**
- Calculate the Within-Cluster-Sum of Squared Errors (WSS) for different values of k, and choose the k for which WSS becomes first starts to diminish. In the plot of WSS-versus-k, this is visible as an elbow.

Elbow Method

- **Within-Cluster-Sum of Squared Errors (WSS) are divided into two part**
- The **Squared Error** for each point is the square of the distance of the point from its representation i.e. its predicted **cluster center.**
- The **WSS score** is the sum of these Squared Errors for all the points.

Elbow Method

Draw a curve between WSS (within sum of squares) and the number of clusters. It is called **elbow method** because the curve looks like a human arm and the elbow point gives us the **optimum number of clusters.**



Elbow Method

- **Import Libraries :**

```
import sklearn.metrics
from scipy.spatial.distance import cdist  # To Calculate Distance
distortions = []
K = range(1,10)
for k in K:
    kmeanModel = KMeans(n_clusters=k).fit(X)
    distortions.append(sum(np.min(cdist(X,kmeanModel.cluster_centers_
    ,'euclidean'),axis=1))/X.shape[0])
```

Elbow Method

- **Plot graph to show optimal value of k**

```
plt.plot(K,distortions,marker='x',color='blue')
plt.xlabel('k')
plt.ylabel('Distortion')
plt.title('The Elbow Method showing the optimal k')
plt.grid()
plt.show()
```

Thank you