**K – MEAN CLUSTERING**

**Aim**

K-means is a clustering algorithm whose main goal is to group similar elements or data points into a cluster (K is the number of clusters).

**Problem Description**

Suppose you are part of social media analytics team responsible for understanding and optimizing user engagement on a live social media platform. The dataset contains real-time information about user reactions to different types of status updates, including the number of reactions, comments, likes, shares, loves, wows, haha, sad and angry reactions.

**Implementation**

Start with choosing a database. Select the number of clusters to be identified. Then randomly select K data points and declare them as centroids. Measure the Euclidean distance from first point to centroids. Assign each point to the nearest centroid. Recalculate the centroids of each cluster by computing the mean of all data points assigned to that cluster. Measure the distance from data points to the new centroids. Whenever the new and old data points are converged into same cluster, then the iteration can be stopped and conclude the clustering by assigning data points to respective clusters.

**Algorithm**

1. Randomly select K data points from the dataset as initial cluster centroids.
2. Assign each data point to the nearest centroid by measuring the distance between each data point to the centroids.
3. Recalculate the centroids of each cluster by computing the mean of all data points assigned to that cluster.
4. Check the distance between data points and new centroids and assign the data points to respective clusters.
5. Check If the algorithm converged by comparing the change in centroids between iterations. If the change is below a predefined threshold, stop the algorithm
6. Repeat the steps 2 to 5 until convergence or the maximum number of iterations is reached.

**Code – Using ScikitLearn**

import numpy as np # linear algebra

import pandas as pd # data processing, CSV file I/O (e.g. pd.read\_csv)

import matplotlib.pyplot as plt # for data visualization

import seaborn as sns # for statistical data visualization

%matplotlib inline

import os

for dirname, \_, filenames in os.walk('/kaggle/input'):

for filename in filenames:

print(os.path.join(dirname, filename))

import warnings

warnings.filterwarnings('ignore')

data = (r'C:\Users\Sreevidya\Desktop\ML\k means\live.csv')

df = pd.read\_csv(data)

df.shape

df.head()

df.info()

df.isnull().sum()

df.drop(['Column1', 'Column2', 'Column3', 'Column4'], axis=1, inplace=True)

df.info()

df.describe()

# view the labels in the variable

df['status\_id'].unique()

# view how many different types of variables are there

len(df['status\_id'].unique())

# view the labels in the variable

df['status\_published'].unique()

# view how many different types of variables are there

len(df['status\_published'].unique())

# view the labels in the variable

df['status\_type'].unique()

# view how many different types of variables are there

len(df['status\_type'].unique())

df.drop(['status\_id', 'status\_published'], axis=1, inplace=True)

df.info()

df.head()

X = df

y = df['status\_type']

from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()

X['status\_type'] = le.fit\_transform(X['status\_type'])

y = le.transform(y)

X.info()

X.head()

cols = X.columns

from sklearn.preprocessing import MinMaxScaler

ms = MinMaxScaler()

X = ms.fit\_transform(X)

X = pd.DataFrame(X, columns=[cols])

X.head()

from sklearn.cluster import KMeans

kmeans = KMeans(n\_clusters=2, random\_state=0)

kmeans.fit(X)

kmeans.cluster\_centers\_

kmeans.inertia\_

labels = kmeans.labels\_

# check how many of the samples were correctly labeled

correct\_labels = sum(y == labels)

print("Result: %d out of %d samples were correctly labeled." % (correct\_labels, y.size))

print('Accuracy score: {0:0.2f}'. format(correct\_labels/float(y.size)))

from sklearn.cluster import KMeans

cs = []

for i in range(1, 11):

kmeans = KMeans(n\_clusters = i, init = 'k-means++', max\_iter = 300, n\_init = 10, random\_state = 0)

kmeans.fit(X)

cs.append(kmeans.inertia\_)

plt.plot(range(1, 11), cs)

plt.title('The Elbow Method')

plt.xlabel('Number of clusters')

plt.ylabel('CS')

plt.show()

from sklearn.cluster import KMeans

kmeans = KMeans(n\_clusters=2,random\_state=0)

kmeans.fit(X)

labels = kmeans.labels\_

# check how many of the samples were correctly labeled

correct\_labels = sum(y == labels)

print("Result: %d out of %d samples were correctly labeled." % (correct\_labels, y.size))

print('Accuracy score: {0:0.2f}'. format(correct\_labels/float(y.size)\*100))

kmeans = KMeans(n\_clusters=3, random\_state=0)

kmeans.fit(X)

# check how many of the samples were correctly labeled

labels = kmeans.labels\_

correct\_labels = sum(y == labels)

print("Result: %d out of %d samples were correctly labeled." % (correct\_labels, y.size))

print('Accuracy score: {0:0.2f}'. format(correct\_labels/float(y.size)\*100))

kmeans = KMeans(n\_clusters=4, random\_state=0)

kmeans.fit(X)

# check how many of the samples were correctly labeled

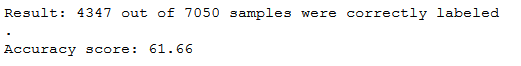
labels = kmeans.labels\_

correct\_labels = sum(y == labels)

print("Result: %d out of %d samples were correctly labeled." % (correct\_labels, y.size))

print('Accuracy score: {0:0.2f}'. format(correct\_labels/float(y.size)\*100))

**Output**

****