Practical: 12

AIM: Implement K-means clustering Algorithm.

```
import numpy as np
import pandas as pd
from scipy.spatial import distance
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
%matplotlib inline
df = pd.read_csv('cluster_validation_data.txt', sep=",", header=None)
#normalize
X = df.values
sc = StandardScaler()
sc.fit(X)
X = sc.transform(X)
def kmeans(X,k=3,max_iterations=100):
    '''
    X: multidimensional data
    k: number of clusters
    max_iterations: number of repetitions before clusters are established

    Steps:
    1. Convert data to numpy aray
    2. Pick indices of k random point without replacement
    3. Find class (P) of each data point using euclidean distance
    4. Stop when max_iteration are reached of P matrix doesn't change

    Return:
    np.array: containg class of each data point
```

```
    '''
    if isinstance(X, pd.DataFrame):X = X.values
    idx = np.random.choice(len(X), k, replace=False)
    centroids = X[idx, :]
    P = np.argmin(distance.cdist(X, centroids, 'euclidean'),axis=1)
    for _ in range(max_iterations):
        centroids = np.vstack([X[P==i,:].mean(axis=0) for i in range(k)])
        tmp = np.argmin(distance.cdist(X, centroids, 'euclidean'),axis=1)
        if np.array_equal(P,tmp):break
        P = tmp
    return P
P = kmeans(X)
assert len(df) == len(P)
# denormalize data
X = sc.inverse_transform(X)
plt.figure(figsize=(15,10))
plt.scatter(X[:,0],X[:,1],c=P)
plt.show()
```

Datasets:

jupyter cluster_validation_data.txt ✔ 2 minutes ago

File Edit View Language

```
2.51007583519798,2.15930271020392
3.73977608389148,0.974175279708383
-0.142930289887991,2.96086572791275
2.81792942905799,2.26801289713400
2.30240725905328,2.11961852048437
0.759417954239056,2.52375917699794
1.58865849027065,2.25954692860121
2.32504210890868,1.09797722854993
5.39476541051290,1.74986977032754
4.62731865525440,1.78705534002171
0.719384805616995,1.53177641540639
4.87918120337060,3.51356612668269
2.68817887254165,1.15004533104548
1.94018089494428,1.54289188663282
2.67806465525994,1.30559252361566
1.80555212382376,0.892000948457531
1.88222633029982,1.82103571888967
3.41325123965188,1.70005641632594
3.33672748685084,3.41309753905715
3.34446677274457,1.72575600615143
2.63703811534278,0.972754420545599
0.854477323834293,3.55301892909512
2.68043232923195,3.15157032008966
3.54657699072313,1.73842499700344
2.46380535441574,0.558563940041997
2.98159597711869,1.55054647490490
2.68958378564190,1.83258252952235
1.71213066271050,2.26995249412184
2.27879095260820,1.74443880291898
1.25311795323180,2.44165973580764
2.84280609790951,2.34775472525983
```

Output: