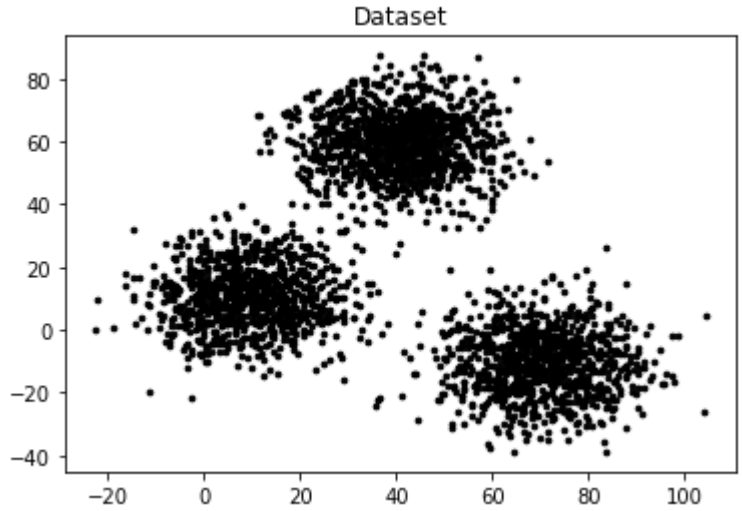
```
from sklearn.mixture import GaussianMixture
from sklearn.cluster import KMeans
import sklearn.metrics as sm
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
```

Apply EM algorithm to cluster a set of data stored in a .CSV file. Use the same data set for clustering using k-Means algorithm. Compare the results of these two algorithms and comment on the quality of clustering. You can add Java/Python ML library classes/API in the program.

```
data=pd.read_csv("/content/drive/MyDrive/My_csv_files/8-Kmeans_EM.csv")
print(data.shape)
data.head()
# Getting the values and plotting it
f1 = data['V1'].values
f2 = data['V2'].values
X = np.array(list(zip(f1, f2)))
plt.scatter(f1, f2, c='black', s=7)
plt.title('Dataset')
```

```
(3000, 2)
Text(0.5, 1.0, 'Dataset')
```

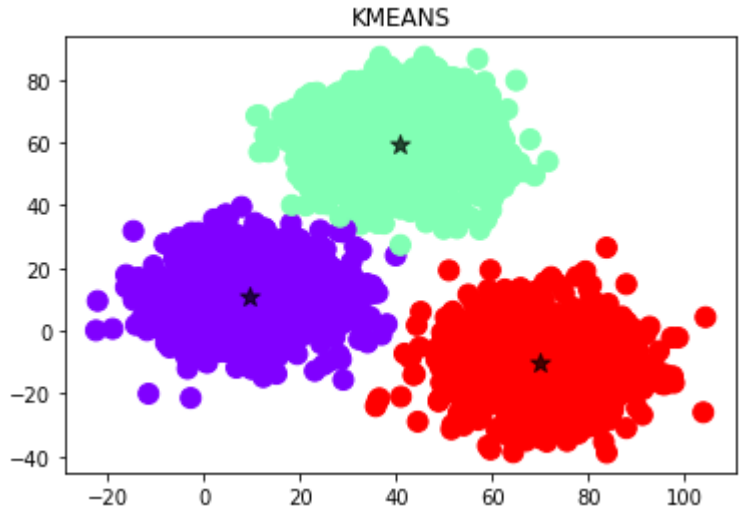


X

```
array([[  2.072345,  -3.241693],
       [ 17.93671 ,  15.78481 ],
       [  1.083576,   7.319176],
       ...,
       [ 64.46532 , -10.50136 ],
       [ 90.72282 , -12.25584 ],
       [ 64.87976 , -24.87731 ]])
```

```
# Number of clusters
k=3
kmeans = KMeans(n_clusters=k)
# Fitting the input data
kmeans = kmeans.fit(X)
# Getting the cluster labels
labels = kmeans.predict(X)
# Centroid values
centroids = kmeans.cluster_centers_
print(centroids)
#plotting the data
plt.title('KMEANS')
plt.scatter(X[:,0], X[:,1], c=labels, cmap='rainbow',s=100)
plt.scatter(kmeans.cluster_centers_[:,0] ,kmeans.cluster_centers_[:,1], marker='*',color='bla
```

```
[[  9.4780459   10.686052  ]
 [ 40.68362784  59.71589274]
 [ 69.92418447 -10.11964119]]
<matplotlib.collections.PathCollection at 0x7fbbcbf192d0>
```



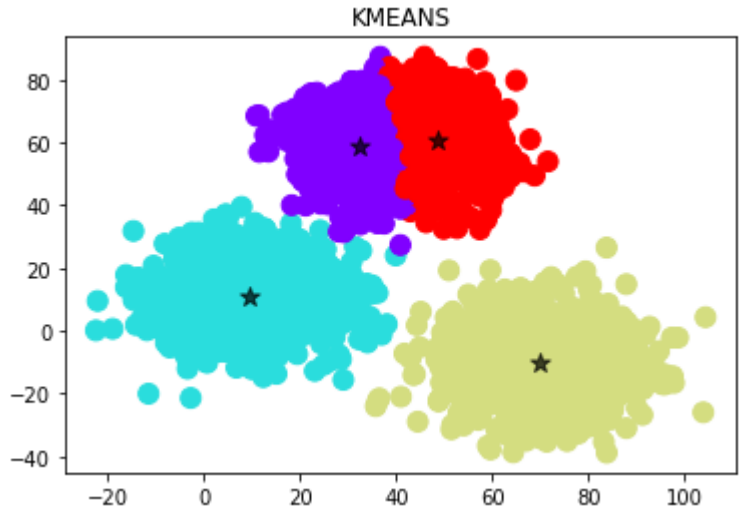

```
print(labels)
```

```
[0 0 0 ... 2 2 2]
```

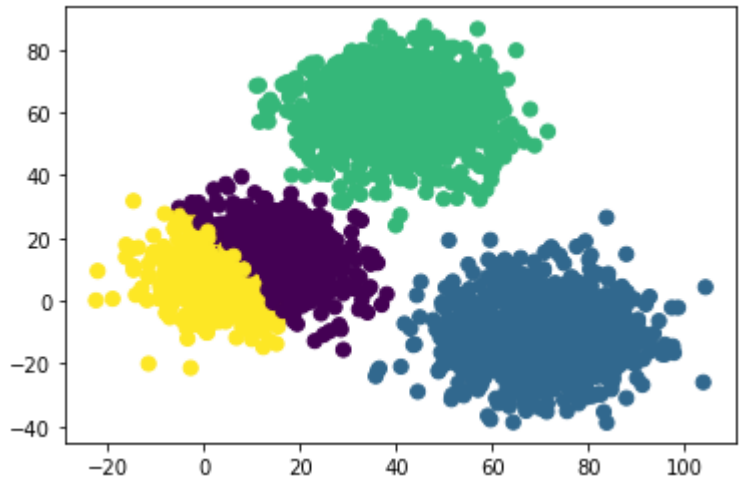
```
# Number of clusters
k=4
kmeans = KMeans(n_clusters=k)
# Fitting the input data
kmeans = kmeans.fit(X)
# Getting the cluster labels
labels = kmeans.predict(X)
# Centroid values
centroids = kmeans.cluster_centers_
print("CENTROIDS")
print(centroids)
```

```
#plotting the data
plt.title('KMEANS')
plt.scatter(X[:,0], X[:,1], c=labels, cmap='rainbow',s=100)
plt.scatter(kmeans.cluster_centers_[:,0] ,kmeans.cluster_centers_[:,1], marker='*' ,color='bl
```

```
CENTROIDS
[[ 32.38267828  58.84242755]
 [  9.41312839  10.61562231]
 [ 69.92418447 -10.11964119]
 [ 48.58547291  60.41087548]]
<matplotlib.collections.PathCollection at 0x7f3e32bb16d0>
```

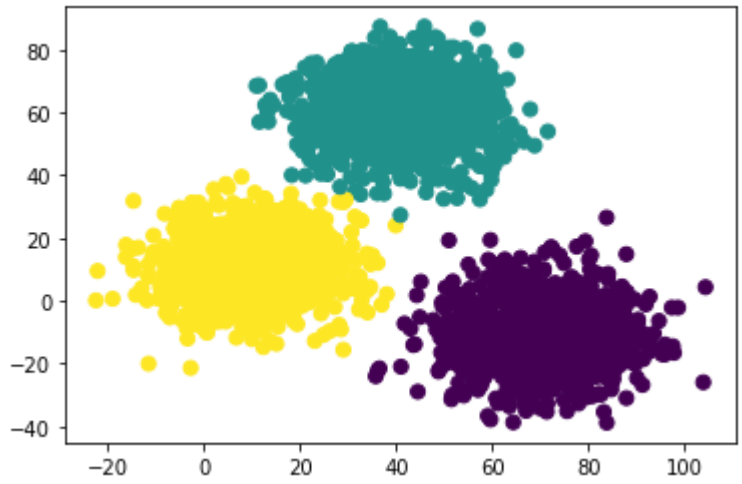



```
gmm = GaussianMixture(n_components=4).fit(X)
labels = gmm.predict(X)
plt.scatter(X[:, 0], X[:, 1], c=labels, s=50, cmap='viridis');
plt.show()
print("EM predictions")
probs = gmm.predict_proba(X)
print(probs)
print("MEAN:\n",gmm.means_)
print("COVARIANCES\n",gmm.covariances_)
```




```
EM predictions
[[3.47083669e-02 1.96126811e-09 2.54311161e-12 9.65291631e-01]
 [9.24887596e-01 3.13672627e-07 6.40339115e-06 7.51056872e-02]
 [1.09399842e-01 1.54995986e-10 4.71486097e-10 8.90600157e-01]
 ...
 [1.82607051e-08 9.99999982e-01 9.52208165e-13 2.69844692e-14]
 [1.12922599e-17 1.00000000e+00 1.72763295e-16 3.65862107e-27]
 [3.29230460e-09 9.99999997e-01 2.40160451e-17 2.90426561e-15]]
MEAN:
```

```
gmm = GaussianMixture(n_components=3).fit(X)
labels = gmm.predict(X)
plt.scatter(X[:, 0], X[:, 1], c=labels, s=50, cmap='viridis');
plt.show()
print("EM predictions")
probs = gmm.predict_proba(X)
print(probs)
print("MEAN:\n",gmm.means_)
print("COVARIANCES\n",gmm.covariances_)
```




```
EM predictions
[[2.47857809e-09 1.51560970e-12 9.99999998e-01]
 [3.50898583e-07 4.99388810e-06 9.99994655e-01]
 [1.79826264e-10 3.06364433e-10 1.00000000e+00]
 ...
 [9.99999967e-01 5.88532728e-13 3.31608645e-08]
 [1.00000000e+00 1.18181069e-16 3.40814080e-15]
 [9.99999999e-01 1.17363874e-17 7.83207859e-10]]
MEAN:
 [[ 69.89436808 -10.11243325]
 [ 40.68585643  59.70939105]
 [  9.46516999  10.71326558]]
COVARIANCES
 [[[110.42521599  -0.60038245]
  [ -0.60038245 106.47434137]]

 [[103.11540751  -1.82258688]
  [ -1.82258688  94.29478126]]

 [[103.72485985   3.83148201]
  [  3.83148201  96.80338694]]]
```