Bharatiya Vidya Bhavan's

Sardar Patel Institute of Technology

Bhavan's Campus, Munshi Nagar, Andheri (West), Mumbai-400058-India

(Autonomous College Affiliated to University of Mumbai)

BE-ETRX B **Sub- AIML Lab**

Name: Shubham Sawant **UID :2019110050**

Name of the Experiment: **Clustering using K mean and EM algorithm**

Objective Apply EM algorithm to cluster a set of data stored in a .CSV file. Use the same dataset 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.

Outcomes:

1. Find the clustering using Estimation and maximization algorithm.
2. Apply K mean and EM algorithm for clustering.
3. Compare the results of these two algorithms
4. Find accuracy, precision, recall of the model for data set.

System Requirements: Windows

Theory:

EM algorithm

In applications of the EM algorithm in statistics, Y is a random vector taking values in RM and governed by the probability density function (pdf) or probability function (pf) fY (y|θtrue). The θ true is a parameter, or vector of parameters, to be estimated; the set Θ is the collection of all potential values of θ true. We have one realization, y, of Y , and we will estimate θtrue by maximizing the likelihood function of θ, given by L(θ) = fY (y|θ), over θ ∈ Θ, to get θ ML, a maximum-likelihood estimate of θ true. In the EM approach it is postulated that there is a second random vector, X, taking values in R N , such that, had we obtained an instance x of X, maximizing the function Lx(θ) = fX(x|θ) would have been computationally simpler than maximizing L(θ) = fY (y|θ). Clearly, maximizing Lx(θ) is equivalent to maximizing LLx(θ) = log fX(x|θ). In most discussions of the EM algorithm the vector y is called the "incomplete" data, while the x is the "complete" data and the situation is described by saying that there is "missing" data. In many applications of the EM algorithm this is suitable terminology. However, any data that we do not have but wish that we did have can be called "missing". I will call the vector y the "observed" data and the x the "preferred" data. It would be reasonable to estimate x, using the current estimate θ k−1 and the data y, and then to use this estimate of x to get the next estimate θ k . Since it is LLx(θ) that we want to maximize, we estimate log fX(x|θ), rather than x itself. The EM algorithm estimates LLx(θ) as

$$E(\log f_X(X|\theta)|y, \theta_{k-1}) = \int f_{X|Y}(x|y, \theta_{k-1}) \log f_X(x|\theta)dx, \quad (2.1)$$

the conditional expected value of the random function log fX(X|θ), conditioned on the data y and the current estimate θ k−1 . This is the so-called E-step of the EM algorithm. It is convenient to define

$$Q(\theta|\theta_{k-1}) := \int f_{X|Y}(x|y, \theta_{k-1}) \log f_X(x|\theta)dx. \quad (2.2)$$

The M-step is to maximize Q(θ|θ k−1) to get θ k . For the case of probability functions we replace the integral with summation. An EM algorithm generates a sequence {θ k} of estimates of θtrue.

K mean clustering:

k-means is one of the simplest unsupervised learning algorithms that solve the well-known clustering problem. The procedure follows a simple and easy way to classify a given data set through a certain number of clusters (assume k clusters) fixed apriori. The main idea is to define k centers, one for each cluster. These centers should be placed in a cunning way because of different location causes different result. So, the better choice is to place them as much as possible far away from each other. The next step is to take each point belonging to a given data set and associate it to the nearest center. When no point is pending, the first step is completed and an early group age is done. At this point we need to re-calculate k new centroids as barycenter of the clusters resulting from the previous step. After we have these k new centroids, a new binding has to be done between the same data set points and the nearest new center. A

loop has been generated. As a result of this loop we may notice that the k centers change their location step by step until no more changes are done or in other words centers do not move any more. Finally, this algorithm aims at minimizing an objective function know as squared error function given by:

Euclidean $\sqrt{\sum_{i=1}^{k}(x_i - y_i)^2}$

Manhattan $\sum_{i=1}^{k}|x_i - y_i|$

Minkowski $\left(\sum_{i=1}^{k}(|x_i - y_i|)^q\right)^{1/q}$

Procedure:

K-Means Algorithm Steps

1. Load data set
2. Clusters the data into k groups where k is predefined.
3. Select k points at random as cluster centers.
4. Assign objects to their closest cluster center according to the Euclidean distance function.
5. Calculate the centroid or mean of all objects in each cluster.
6. Repeat steps 3, 4 and 5 until the same points are assigned to each cluster in consecutive rounds.

Data Set Link: https://www.kaggle.com/datasets/fedesoriano/stroke-prediction-dataset

Dataset Description:

Number of Instances: 100
Number of Attributes (including the class attribute): 9
Attribute Information:

1. **age**: Age of an individual
2. **Gender**: Gender of an individual
3. **Hypertension**: Hypertension binary feature
4. **Work_Type:** Work type of the patient
5. **Residence_Type**: Residence type of the patient
6. **Avg_Glucose_Level**: Average glucose level in blood
7. **Bmi**: Body Mass Index
8. **Smoking_Status**: Smoking status of the patient
9. **Stroke**: Stroke event

Bharatiya Vidya Bhavan's
Sardar Patel Institute of Technology
Bhavan's Campus, Munshi Nagar, Andheri (West), Mumbai-400058-India
(Autonomous College Affiliated to University of Mumbai)

BE-ETRX B **Sub- AIML Lab**
Name: Shubham Sawant **UID :2019110050**

Code:

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
from scipy import stats
```
✓ 0.3s

```
from sklearn.cluster import KMeans
```
✓ 0.2s

```
df = pd.read_csv(r'C:\Users\Dell\Desktop\Shubham\SEM7\AIML\EXP8\healthcare-dataset-stroke-data.csv')
```
✓ 0.4s

```
df.columns
```
✓ 0.3s

```
Index(['id', 'gender', 'age', 'hypertension', 'work_type', 'Residence_type',
       'avg_glucose_level', 'bmi', 'smoking_status', 'stroke'],
      dtype='object')
```

```
df = df.dropna()
df.reset_index(inplace=True, drop=True)
```
✓ 0.4s

```
X_numerics = df[['age', 'avg_glucose_level', 'bmi']]
```

```
from yellowbrick.cluster import KElbowVisualizer

model = KMeans(random_state=1)
visualizer = KElbowVisualizer(model, k=(2,10))

visualizer.fit(X_numerics)
visualizer.show()
plt.show()
```
✓ 0.8s

```
model = KMeans(random_state=1)
visualizer = KElbowVisualizer(model, k=(2,10), metric='silhouette')

visualizer.fit(X_numerics)
visualizer.show()
plt.show()
```

BE-ETRX B **Sub- AIML Lab**

Name: Shubham Sawant **UID :2019110050**

```
KM_5_clusters = KMeans(n_clusters=2, init='k-means++').fit(X_numerics) # initialise and fit K-Means model

KM5_clustered = X_numerics.copy()
KM5_clustered.loc[:,'Cluster'] = KM_5_clusters.labels_ # append labels to points
```

✓ 0.8s

```
fig1, (axes) = plt.subplots(1,2,figsize=(12,5))


scat_1 = sns.scatterplot('avg_glucose_level', 'bmi', data=KM5_clustered,
                         hue='Cluster', ax=axes[0], palette='Set1', legend='full')

sns.scatterplot('age', 'avg_glucose_level', data=KM5_clustered,
                         hue='Cluster', palette='Set1', ax=axes[1], legend='full')

axes[0].scatter(KM_5_clusters.cluster_centers_[:,1],KM_5_clusters.cluster_centers_[:,2], marker='s', s=40, c="blue")
axes[1].scatter(KM_5_clusters.cluster_centers_[:,0],KM_5_clusters.cluster_centers_[:,2], marker='s', s=40, c="blue")
plt.show()
```

```
KM_clust_sizes = KM5_clustered.groupby('Cluster').size().to_frame()
KM_clust_sizes.columns = ["KM_size"]
KM_clust_sizes
```

✓ 0.8s

Cluster	KM_size
0	30
1	52

```
KM_5_clusters.labels_
```

✓ 0.3s

```
array([0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1,
       1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1,
       0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1])
```

```
(sum([1 if i==j else 0 for i,j in zip(np.array(df['stroke']), KM_5_clusters.labels_)])/len(df['stroke']))*100
```

Bharatiya Vidya Bhavan's
Sardar Patel Institute of Technology
Bhavan's Campus, Munshi Nagar, Andheri (West), Mumbai-400058-India
(Autonomous College Affiliated to University of Mumbai)

BE-ETRX B **Sub- AIML Lab**

Name: Shubham Sawant **UID :2019110050**

```python
def gen_data(k=3, dim=2, points_per_cluster=200, lim=[-10, 10]):
    x = []
    mean = random.rand(k, dim)*(lim[1]-lim[0]) + lim[0]
    for i in range(k):
        cov = random.rand(dim, dim+10)
        cov = np.matmul(cov, cov.T)
        _x = np.random.multivariate_normal(mean[i], cov, points_per_cluster)
        x += list(_x)
    x = np.array(x)
    if(dim == 2):
        fig = plt.figure()
        ax = fig.gca()
        ax.scatter(x[:,0], x[:,1], s=3, alpha=0.4)
        ax.autoscale(enable=True)
    return x
```

✓ 0.4s

```python
X_numerics.isna().sum()
```

```python
X = np.array(X_numerics[['age','avg_glucose_level']])
```

```python
def plot(title):
    fig = plt.figure(figsize=(8, 8))
    ax = fig.gca()
    ax.scatter(X[:, 0], X[:, 1], s=3, alpha=0.4)
    ax.scatter(gmm.mu[:, 0], gmm.mu[:, 1], c=gmm.colors)
    gmm.draw(ax, lw=3)
    ax.set_xlim((-12, 12))
    ax.set_ylim((-12, 12))

    plt.title(title)
    plt.savefig(title.replace(':', '_'))
    plt.show()
    plt.clf()
```

```python
X = gen_data(k=2, dim=2, points_per_cluster=1000)
```

BE-ETRX B **Sub- AIML Lab**

Name: Shubham Sawant **UID :2019110050**

```
gmm = GMM(2, 2)
```

```
# Training the GMM using EM

# Initialize EM algo with data
gmm.init_em(X)
num_iters = 3
# Saving log-likelihood
log_likelihood = [gmm.log_likelihood(X)]
# plotting
plot("Iteration:  0")
for e in range(num_iters):
    # E-step
    gmm.e_step()
    # M-step
    gmm.m_step()
    # Computing log-likelihood
    log_likelihood.append(gmm.log_likelihood(X))
    print("Iteration: {}, log-likelihood: {:.4f}".format(e+1, log_likelihood[-1]))
    # plotting
    plot(title="Iteration: " + str(e+1))
```

```
# Plotting for creating GIF of log-likehood graph
for i in range(1, len(log_likelihood)):
    plt.title("log-likelihood for iteration: " + str(i))
    plt.plot(log_likelihood[1:1+i], marker='.')
    axes = plt.axes()
    axes.set_ylim([min(log_likelihood[1:])-50, max(log_likelihood[1:])+50])
    axes.set_xlim([-2, 32])
    plt.savefig("ll_" + str(i))
    plt.clf()
```

BE-ETRX B **Sub- AIML Lab**

Name: Shubham Sawant **UID :2019110050**

Interpretation of output:

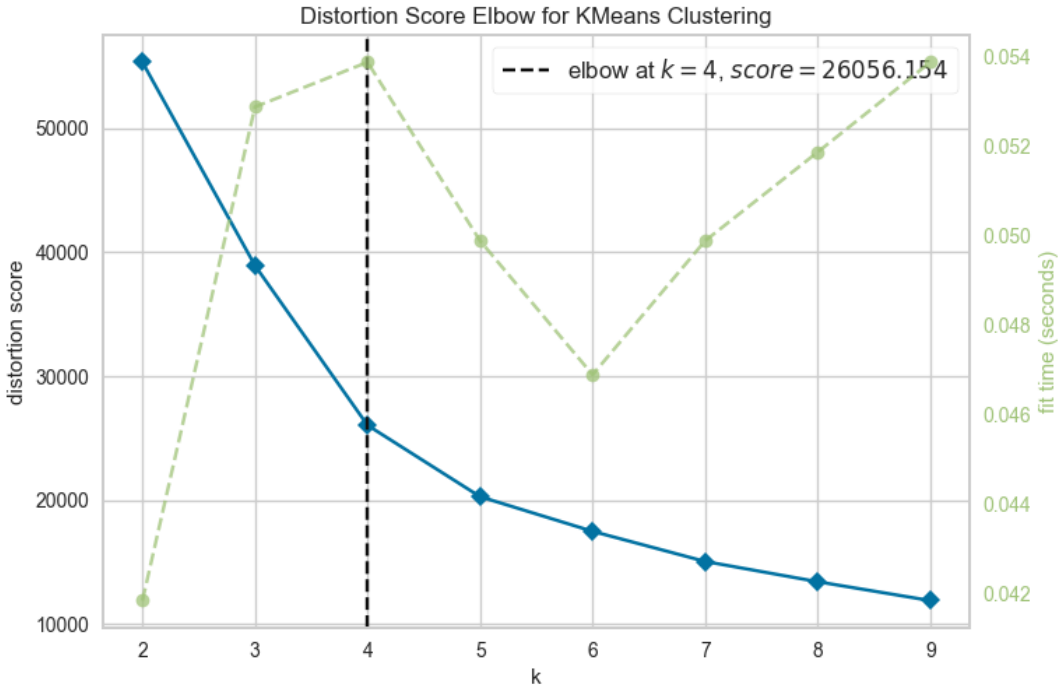

Fig 8.1: Distortion Score Elbow for KMeans Clustering

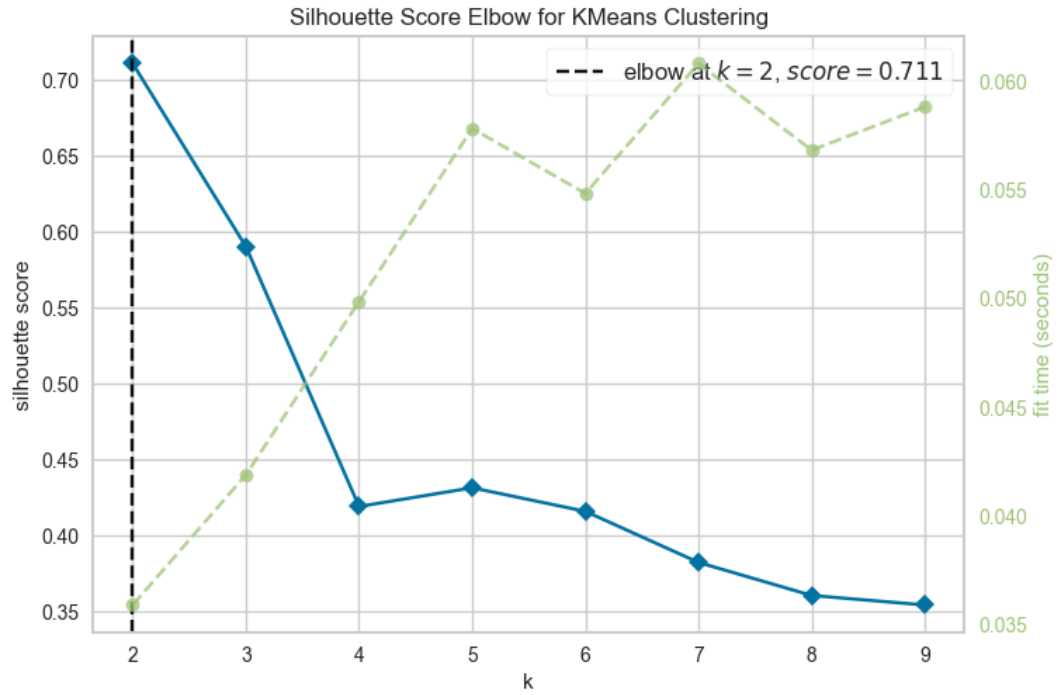

Fig 8.2: Silhouette Score Elbow for KMeans Clustering

BE-ETRX B **Sub- AIML Lab**
Name: Shubham Sawant **UID :2019110050**

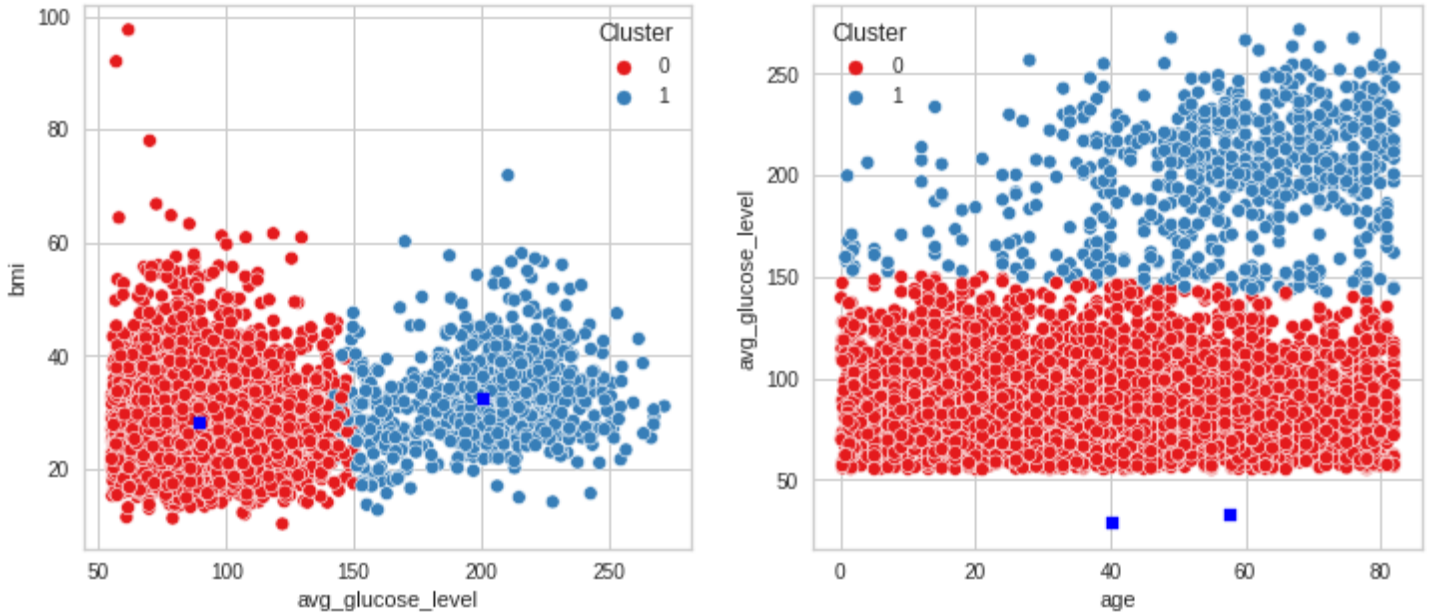

Fig 8.3: Cluster plots

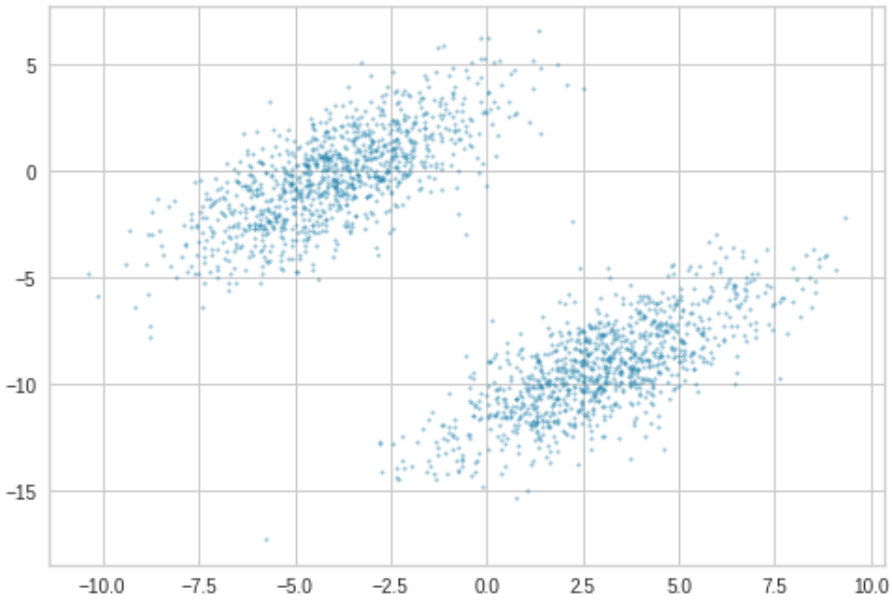

Fig 8.4: Random mixture of Gaussians in given range

BE-ETRX B **Sub- AIML Lab**
Name: Shubham Sawant **UID :2019110050**

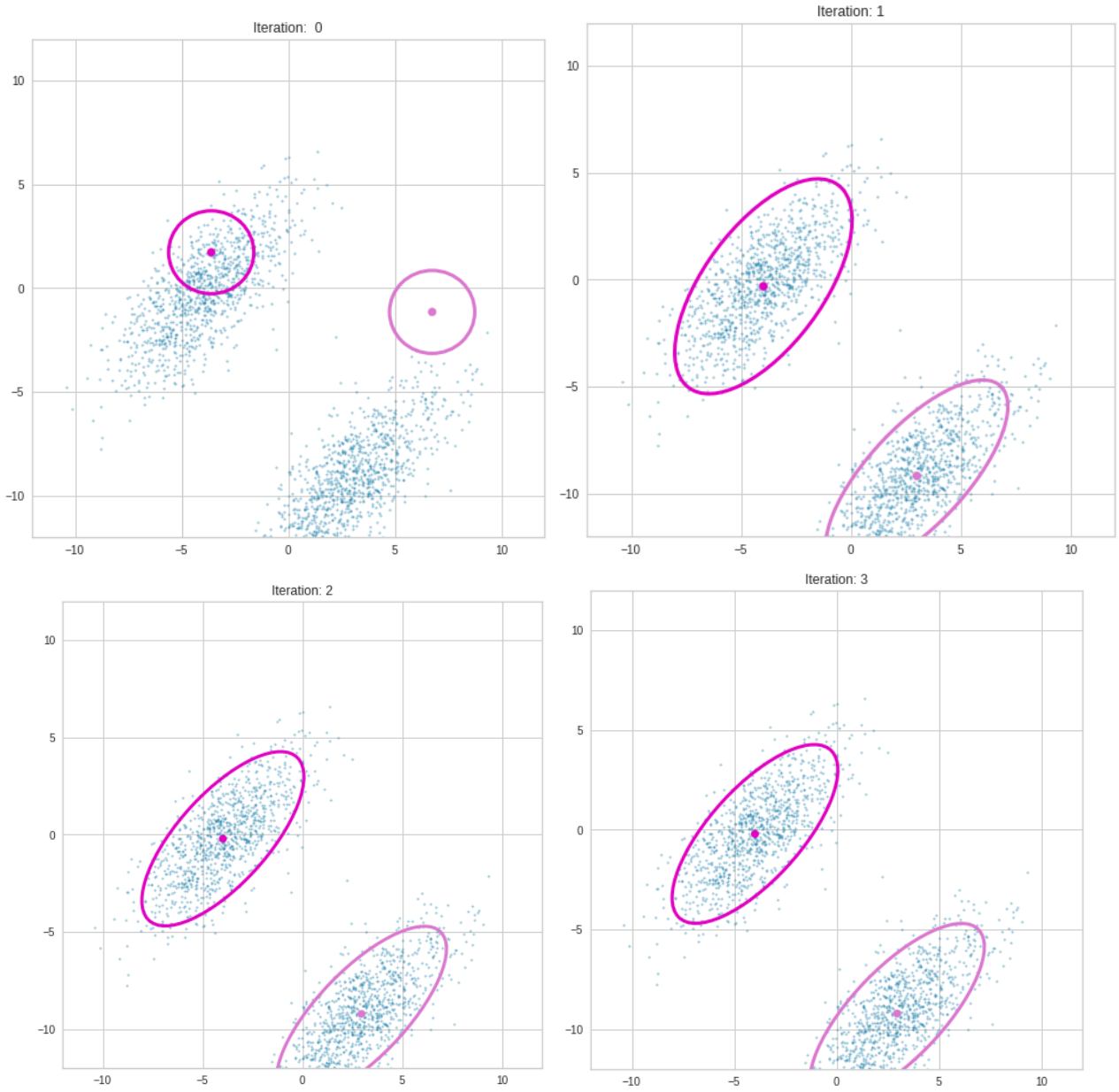

Fig 8.5: Plotting of each Iteration

Conclusion:

1. Therefore, we can conclude that we identified the problem and worked towards creating a solution for the problem.
2. We used sklearn's KNN inbuilt library. There are total 10 input features each with different units having values in different ranges. We visualised the generated clustered data and found out the centers of the clusters.
3. We also used the EM algorithm for clustering. The results were better than KNN algorithm. The results are better because we are estimating and maximising our results. Thus, we learned how clustering works in KNN and EM algorithm.
4. We also implemented the algorithm and calculated our results and plotted the graphs.