Application 13
Unsupervised Machine Learning

Clustering using K Mean Algorithm

Consider below Machine Learning application which implements K mean Algorithm for randomly generated dataset.

```
import numpy as np
import pandas as pd
from copy import deepcopy
from matplotlib import pyplot as plt

def MarvellousKMean():
    # Set three centers, the model should predict similar results
    center_1 = np.array([1,1])
    print(center_1)

    center_2 = np.array([5,5])
    print(center_2)

    center_3 = np.array([8,1])
    print(center_3)

    # Generate random data and center it to the three centers
    data_1 = np.random.randn(7, 2) + center_1
    print("Elements of first cluster with size"+str(len(data_1)))
    print(data_1)

    data_2 = np.random.randn(7,2) + center_2
    print("Elements of second cluster with size"+str(len(data_2)))
    print(data_2)

    data_3 = np.random.randn(7,2) + center_3
    print("Elements of third cluster with size"+str(len(data_3)))
    print(data_3)

    data = np.concatenate((data_1, data_2, data_3), axis = 0)
    print("Size of complete data set"+str(len(data)))

    plt.scatter(data[:,0], data[:,1], s=7)
    plt.title('Marvellous Infosystems : Input Dataset')
    plt.show()

    # Number of clusters
    k = 3

    # Number of training data
    n = data.shape[0]
    print("Total number of elements are",n)

    # Number of features in the data
    c = data.shape[1]
    print("Total number of features are",c)

    # Generate random centers, here we use sigma and mean to ensure it represent the whole data
    mean = np.mean(data, axis = 0)
    print("Value of mean",mean)

    # Calculate standard deviation
    std = np.std(data, axis = 0)
    print("Value of std",std)

    centers = np.random.randn(k,c)*std + mean
```

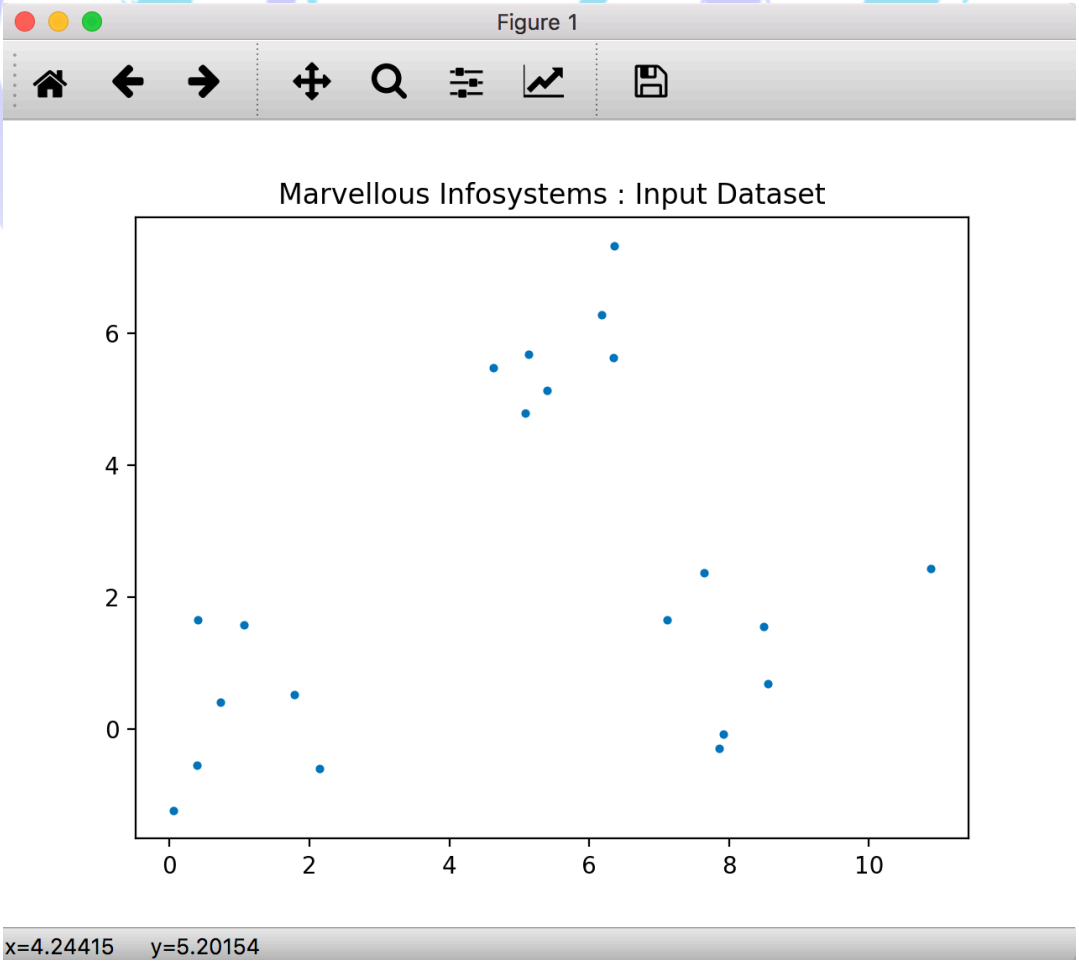
```
    print("Random points are",centers)

    # Plot the data and the centers generated as random
    plt.scatter(data[:,0], data[:,1],c='r', s=7)
    plt.scatter(centers[:,0], centers[:,1], marker='*', c='g', s=150)
    plt.title('Marvellous Infosystems : Input Datase with random centroid *')
    plt.show()

    centers_old = np.zeros(centers.shape)   # to store old centers
    centers_new = deepcopy(centers)        # Store new centers

    print("Values of old centroids")
    print(centers_old)

    print("Values of new centroids")
    print(centers_new)

    data.shape
    clusters = np.zeros(n)
    distances = np.zeros((n,k))

    print("Initial distances are")
    print(distances)

    error = np.linalg.norm(centers_new - centers_old)

    # When, after an update, the estimate of that center stays the same, exit loop
    while error != 0:

        # Measure the distance to every center
        print("Measure the distance to every center")
        for i in range(k):
            print("Iteration number ",i)
            distances[:,i] = np.linalg.norm(data - centers[i], axis=1)

        # Assign all training data to closest center
        clusters = np.argmin(distances, axis = 1)

        centers_old = deepcopy(centers_new)

        # Calculate mean for every cluster and update the center
        for i in range(k):
            centers_new[i] = np.mean(data[clusters == i], axis=0)
        error = np.linalg.norm(centers_new - centers_old)
    # end of while
    centers_new

    # Plot the data and the centers generated as random
    plt.scatter(data[:,0], data[:,1], s=7)
    plt.scatter(centers_new[:,0], centers_new[:,1], marker='*', c='g', s=150)
    plt.title('Marvellous Infosystems : Final data with Centroid')
    plt.show()

def main():
    print("---- Marvellous Infosystems by Piyush Khairnar-----")

    print("Unsuervised Machine Learning")

    print("Clustering using K Mean Algorithm")

    MarvellousKMean()

if __name__ == "__main__":
    main()

```

Output of above application

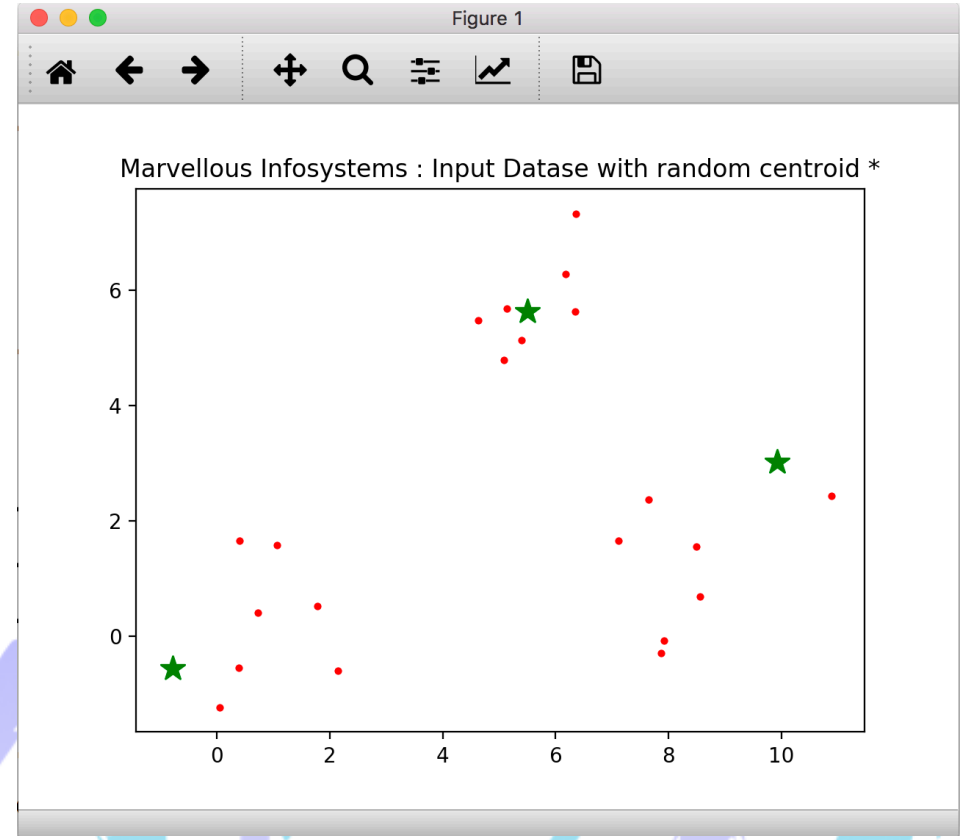
```
Clustering — -bash — 51×20
(base) MacBook-Pro-de-MARVELLOUS:Clustering marvell
ous$ python3 Clustering.py
---- Marvellous Infosystems by Piyush Khairnar-----
Unsuervised Machine Learning
Clustering using K Mean Algorithm
[1 1]
[5 5]
[8 1]
Elements of first cluster with size7
[[ 1.78591077  0.52204584]
 [ 1.06552455  1.58010131]
 [ 2.14878607 -0.60838652]
 [ 0.40137758  1.64781455]
 [ 0.39167985 -0.55167997]
 [ 0.05705409 -1.23997492]
 [ 0.72317674  0.3989142 ]]
Elements of second cluster with size7
[[5.13100873 5.68867927]
 [4.63122904 5.4776318 ]
 [5.08787673 4.79849352]]
Elements of third cluster with size7
[[ 7.91639025 -0.08093321]
 [ 7.6395752   2.36452259]
 [ 8.55221324  0.68666628]
 [10.88013951  2.43256217]
 [ 7.86321404 -0.29989975]
 [ 7.10880253  1.65059962]
 [ 8.49230784  1.55607116]]
Size of complete data set21
Total number of elements are 21
Total number of features are 2
Value of mean [4.95998816 2.40135806]
Value of std [3.17645752 2.5908486 ]
Random points are [[ 9.91497286  3.02323184]
 [ 5.5055685   5.63498908]
 [-0.77374946 -0.56501315]]
Values of old centroids
[[0. 0.]
 [0. 0.]
 [0. 0.]]
```

```
 [0.  0.  0.]
 [0.  0.  0.]
 [0.  0.  0.]
 [0.  0.  0.]
 [0.  0.  0.]
 [0.  0.  0.]
 [0.  0.  0.]
 [0.  0.  0.]
 [0.  0.  0.]
 [0.  0.  0.]]
Measure the distance to every center
Iteration number  0
Iteration number  1
Iteration number  2
Measure the distance to every center
Iteration number  0
Iteration number  1
Iteration number  2
(base) MacBook-Pro-de-MARVELLOUS:Clustering marvell
ous$
```

Plotting of input data set



Plotting of input data set with random centroids



Plotting of data elements after clustering