```python
import random
import math
import sys
import matplotlib.pyplot as plt

def cluster(centroids, datapoints):

    # for each point find the nearest Center
    for p in datapoints:
        minD = distance(p[0], centroids[0][0])
        p[1] = centroids[0][0]
        for c in centroids:
            d = distance(p[0], c[0])
            if(d < minD):
                minD = d
                p[1] = c[0]
    return datapoints

# Update the centers
def calculateCentroid(centroids, datapoints):

    # for each Center find it's points
    for c in centroids:
        x=0
        y=0
        count = 0
        for p in points:
            # if a Point's Label is center C
            if p[1] == c[0]:
                # include it in (x1+x2,.../points + y1+y2,.../points)
                x += p[0][0]
                y += p[0][1]
                count += 1
        c[0] = [x/count, y/count]

    return centroids

def distance(p, c):
    d = 0
    for i in range(2):
        d += ((p[i] - c[i]) ** 2)
    return math.sqrt(d)

def unshared_copy(inList):
    if isinstance(inList, list):
        return list( map(unshared_copy, inList) )
```

```
47      return inList
48  ################################################ Input
49
50  # get the DataSet, K, M
51  with open(sys.argv[1]) as f:
52      k, m = [int(x) for x in f.readline().split()]
53
54      # each point = [ [coordinates], [Label] ]
55      points = [list([list([float(x) for x in line.split()]),[]]) for line in
    f]
56
57  # get an Unshared Copy of Points ,
58  # so it doesn't have any reference to 'points'
59  # because we are picking up centers from points and then we change them
60  pointC = unshared_copy(points)
61
62  centers = []
63
64  ################################################ K-Mean
65  # Initialize First Centers Randomly
66  while len(centers) < m:
67      centers.append(random.choice(pointC))
68
69  # 5 Iterations for K-Mean
70  for i in range(5):
71      points = cluster(centers, points)
72      centers = calculateCentroid(centers, points)
73
74  ################################################ Plot
75  # Clusters
76  for p in points:
77      if p[1][0] == centers[0][0][0]:
78          plt.scatter(p[0][0], p[0][1], s=125, c='red')
79      else:
80          plt.scatter(p[0][0], p[0][1], s=125, c='blue')
81  # Centers
82  for c in centers:
83      plt.scatter(c[0][0], c[0][1], s=125, c='yellow')
84  plt.xlim(-1, 4)
85  plt.ylim(-1, 4)
86  plt.show()
87
88  # Final Centers : [[1.8, 2.867], [1.032, 1.212]]
89  ######################
90
```