1)

A) The output after analysis dataset xeasy:

After 100 iteration of EM algorithm, the parameter pi is:

```
[[ 0.5911577]
  [ 0.4088423]]
```

The average of the two cluster(mu) are:

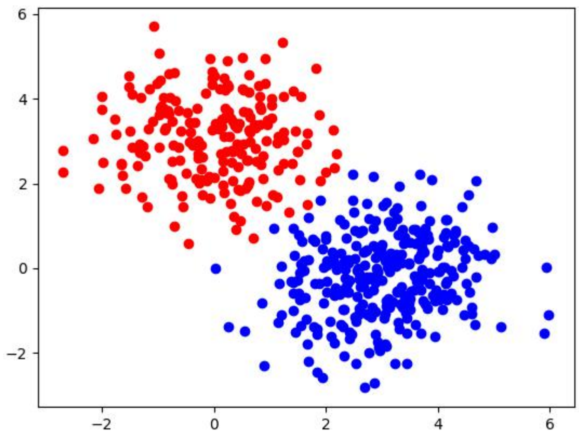
```
[[ 3.01886057 -0.17711179]
  [ 0.02847666    3.07051221]]
```

The covariance matrix of the two clusters are:

```
[[[ 1.00528764    0.16007171]
    [ 0.16007171    0.94210869]]

  [[ 1.0184948    -0.05861037]
    [-0.05861037    0.95457458]]]
```

The first row and second row of the output are for the first cluster and second cluster separately(which means axis=0 gives the two different value of k), the cluster result are shown as following:



B)The output after analysis dataset x1:

After 100 iteration of EM algorithm, the parameter pi is:

```
[[ 0.65187136]
  [ 0.34812864]]
```

The average of the two cluster(mu) are:

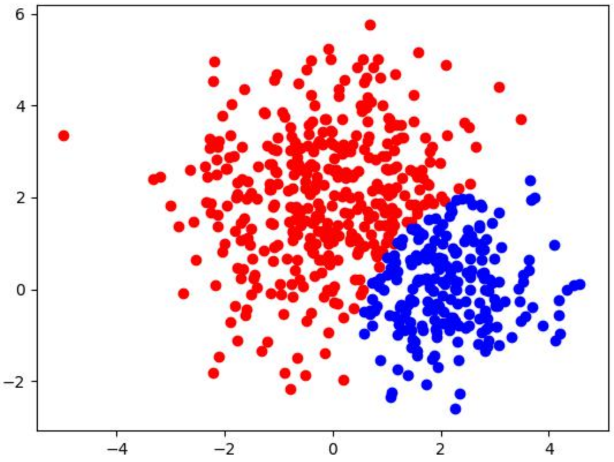
```
[[-0.10245865    1.97455142]
  [ 2.08271849    0.1263725 ]]
```

The covariance matrix of the two clusters are:

```
[[[ 1.70226018    0.1507016 ]
    [ 0.1507016      2.19963592]]

  [[ 0.86366148 -0.03966277]
    [-0.03966277    1.0604482 ]]]
```

The first row and second row of the output are for the first cluster and second cluster separately, the cluster result are shown as following:

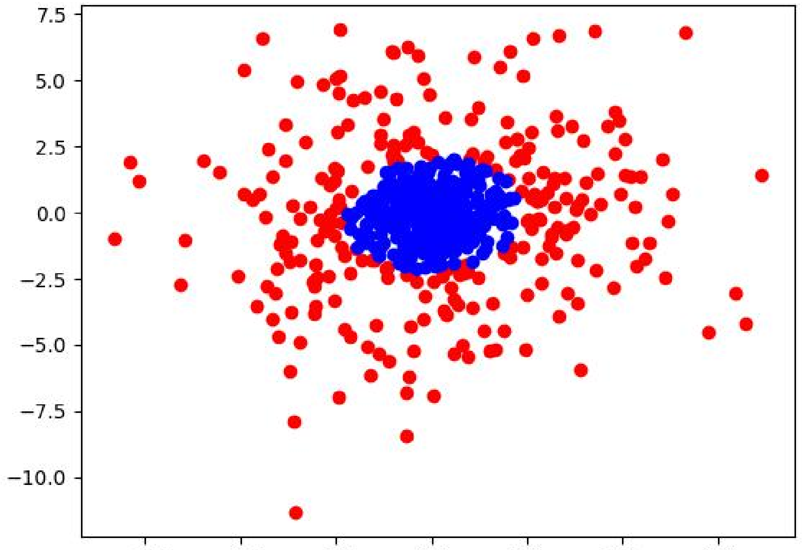


C)The output after analysis dataset x2:

```
After 100 iteration of EM algorithm, the parameter pi is:
[[ 0.48803811]
  [ 0.51196189]]
The average of the two cluster(mu) are:
[[ 0.1780945    -0.12519353]
  [ 0.01424334 -0.04651479]]
The covariance matrix of the two clusters are:
[[[ 9.25803313    0.76781765]
    [ 0.76781765    9.41423546]]

  [[ 1.06249481    0.06633439]
    [ 0.06633439    0.88825803]]]
```

The first row and second row of the output are for the first cluster and second cluster separately, the cluster result are shown as following:



The code are shown as following, which is implemented all by myself:

```python
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from scipy.stats import multivariate_normal

data=pd.read_csv('x2.txt', sep=',', header=None).dropna(1).as_matrix()
m=data.shape[0]
n=data.shape[1]


def distance(a, b):
    return np.linalg.norm(a-b)

def kmean(k, data, dist, m, n):
    iteration=1000
    times=100
    cost=10000000
    result=np.zeros([m])
    for s in range(times):
        centerlist = np.zeros([k, n])
        index = np.zeros([m])
        for i in range(k):
            index[i] = 1
        np.random.shuffle(index)
        counter = 0
        for i in range(m):
            if index[i] == 1:
                centerlist[counter] = data[i]
                counter += 1

        for _ in range(iteration):
            old = np.array(centerlist)
            for i in range(m):
                ind = 0
                dis = 10000000
                for j in range(k):
                    d = dist(centerlist[j], data[i])
                    if d < dis:
                        dis = d
                        ind = j
                index[i] = ind

            for i in range(k):
                counter = 0
                sum = np.zeros([n])
                for j in range(m):
                    if index[j] == i:
                        sum += data[j]
                        counter += 1
                sum /= counter
                centerlist[i] = np.array(sum)
            if np.all(old == centerlist):
                break


        c=costfunc(index,centerlist, data, dist, m)
        if c<cost:
            cost=c
            result=np.array(index)
    return result

def costfunc(index,centerlist, data, dist, m):
    rez=0
    for i in range(m):
        rez+=dist(centerlist[int(index[i])], data[i])
    return rez/m
```

```
def showrez(index, list, k, m):
    colorlist=['red', 'blue', 'yellow', 'green', 'pink', 'gray',
'black','magenta', 'aqua','gold','navy','orangered']
    y=np.zeros([m,2])
    steplist=np.zeros([k])
    counter=0
    for i in range(k):
        for j in range(m):
            if index[j]==i:
                y[counter]=list[j]
                counter+=1
        steplist[i]=counter

    for i in range(k):
        if i==0:
            px, py = y[0:int(steplist[i])].T
        else:
            px, py = y[int(steplist[i-1]):int(steplist[i])].T
        plt.scatter(px, py, color=colorlist[i])
    plt.show()


k=2
#k means clustering
index=kmean(k, data, distance, m, n)


#Gauss model learning
y=np.zeros([m, k])
pi=np.zeros([k,1])
mu=np.zeros([k,2])
cov=np.zeros([k,2,2])
e=np.ones([1,m])
for i in range(m):
    y[i][int(index[i])]=1

for i in range(100):
    #M stage
    pi=np.transpose(e.dot(y))/m
    mu = np.zeros([k, 2])
    for s in range(k):
        for j in range(m):
            mu[s]=mu[s]+y[j][s]*data[j]
    mu= mu/(pi*m)
    #print('1')
    cov = np.zeros([k, 2, 2])
    for s in range(k):
        for j in range(m):
            vec=data[j]-mu[s]
            temp=np.zeros([2,2])
            for st in range(2):
                for jt in range(2):
                    temp[st][jt]=vec[st]*vec[jt]
            cov[s]=cov[s]+y[j][s]*(temp)

    for s in range(k):
        cov[s]=cov[s]/pi[s][0]/m
#E stage
    y=np.zeros([m,k])
    for s in range(k):
        y[:,s]=multivariate_normal.pdf(data, mu[s], cov[s])*pi[s][0]
    ys=np.sum(y, axis=1)
    for j in range(m):
        y[j]=y[j]/ys[j]

print("After 100 iteration of EM algorithm, the parameter pi is:")
print(pi)
print("The average of the two cluster(mu) are:")
print(mu)
print ("The covariance matrix of the two clusters are:")
print(cov)
p = np.argmax(y, axis=1)
index=np.expand_dims(p, axis=1)
showrez(index, data, k, m)
```

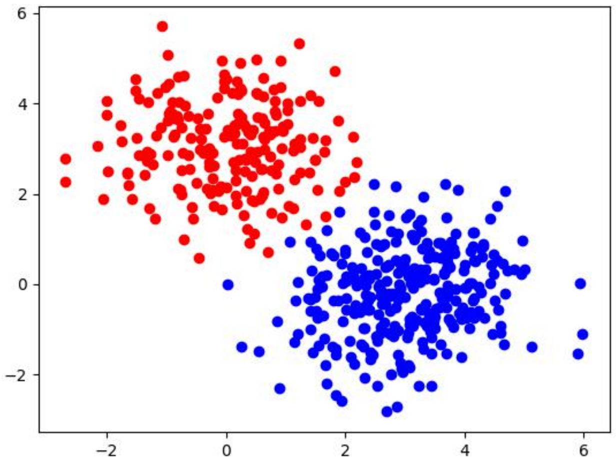
2)Repeat with the Provable EM

A)The output after analysis dataset xeasy:

```
After the two step EM algorithm, the parameter pi is:
[[ 0.4088423]
 [ 0.5911577]]
The average of the two cluster(mu) are:
[[ 0.02847666  3.07051221]
 [ 3.01886057 -0.17711179]]
The covariance matrix of the two clusters are:
[[[ 1.0184948  -0.05861037]
  [-0.05861037  0.95457458]]

 [[ 1.00528764  0.16007171]
  [ 0.16007171  0.94210869]]]
```

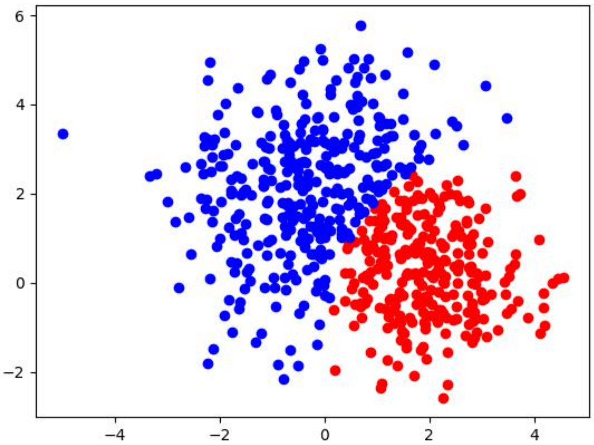
The first row and second row of the output are for the first cluster and second cluster separately(which means axis=0 gives the two different value of k), the cluster result are shown as following:



B)The output after analysis dataset x1:

```
After the two step EM algorithm, the parameter pi is:
[[ 0.45245168]
 [ 0.54754832]]
The average of the two cluster(mu) are:
[[ 1.86065036  0.37827939]
 [-0.3352951   2.11852399]]
The covariance matrix of the two clusters are:
[[[ 1.04157824 -0.16761685]
  [-0.16761685  1.35831271]]
 [[ 1.51224836    0.3479144 ]
   [ 0.3479144      2.21600941]]]
```

The first row and second row of the output are for the first cluster and second cluster separately, the cluster result are shown as following:



C)The output after analysis dataset x2:

After the two step EM algorithm, the parameter pi is:

```
[[ 0.51286941]
  [ 0.48713059]]
```

The average of the two cluster(mu) are:

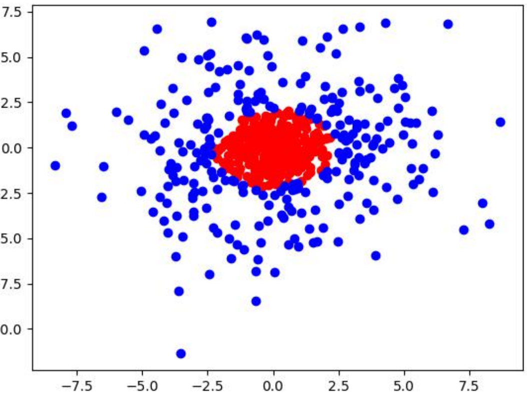
```
[[ 0.01453492 -0.04638625]
  [ 0.17809277 -0.12547543]]
```

The covariance matrix of the two clusters are:

```
[[[ 1.06649564    0.06699466]
   [ 0.06699466    0.88985815]]

  [[ 9.26913956    0.76845133]
   [ 0.76845133    9.42840164]]]
```

The first row and second row of the output are for the first cluster and second cluster separately, the cluster result are shown as following:



The code is as following:

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from scipy.stats import multivariate_normal

data=pd.read_csv('x2.txt', sep=',', header=None).dropna(1).as_matrix()
m=data.shape[0]
n=data.shape[1]


def distance(a, b):
    return np.linalg.norm(a-b)


def showrez(index, list, k, m):
    colorlist=['red', 'blue', 'yellow', 'green', 'pink', 'gray',
'black','magenta', 'aqua','gold','navy','orangered']
    y=np.zeros([m,2])
    steplist=np.zeros([k])
    counter=0
    for i in range(k):
        for j in range(m):
            if index[j]==i:
                y[counter]=list[j]
                counter+=1
        steplist[i]=counter

    for i in range(k):
        if i==0:
            px, py = y[0:int(steplist[i])].T
        else:
            px, py = y[int(steplist[i-1]):int(steplist[i])].T
        plt.scatter(px, py, color=colorlist[i])
    plt.show()

def pruning(pi, mu, I): #for convinience only work for k=2
    mask=np.ones(I)
    for i in range(I):
        if pi[i][0]<1/(4*I):
            mask[i]=0
    dist=0
    x=0
    y=0
    for i in range(I):
        for j in range(I):
            if(mask[i]==1 and mask[j]==1 and distance(mu[i],mu[j])>dist):
                dist=distance(mu[i],mu[j])
                x=i
                y=j
    return (x,y)
k=2
I=10
#Gauss model learning
pi=np.zeros([I,1])
mu=np.zeros([I,2])
cov=np.zeros([I,2,2])
e=np.ones([1,m])
y=np.zeros([m,I])

for i in range(I):
    pi[i][0]=1/I
    mu[i][0]=np.random.uniform(-5,5)
    mu[i][1]=np.random.uniform(-5,5)

for i in range(I):
    mi=100000
    for j in range(I):
        if distance(mu[i], mu[j])<mi and i!=j:
```

```
        mi=distance(mu[i], mu[j])
    cov[i][0][0]=cov[i][1][1]=mi

for _ in range(50):
    #E stage
    y=np.zeros([m,I])
    for s in range(I):
        y[:,s]=multivariate_normal.pdf(data, mu[s], cov[s])*pi[s][0]
    ys=np.sum(y, axis=1)
    for j in range(m):
        y[j]=y[j]/ys[j]

    #M stage
    pi=np.transpose(e.dot(y))/m
    mu = np.zeros([I, 2])
    for s in range(I):
        for j in range(m):
            mu[s]=mu[s]+y[j][s]*data[j]
    mu= mu/(pi*m)
    #print('1')
    cov = np.zeros([I, 2, 2])
    for s in range(I):
        for j in range(m):
            vec=data[j]-mu[s]
            temp=np.zeros([2,2])
            for st in range(2):
                for jt in range(2):
                    temp[st][jt]=vec[st]*vec[jt]
            cov[s]=cov[s]+y[j][s]*(temp)

    for s in range(I):
        cov[s]=cov[s]/pi[s][0]/m

#pruing
a, b=pruning(pi, mu, I)
pi=pi[[a,b],:]
mu=mu[[a,b],:]
cov=cov[[a,b],:,:]

for _ in range(50):
    #E stage
    y=np.zeros([m,k])
    for s in range(k):
        y[:,s]=multivariate_normal.pdf(data, mu[s], cov[s])*pi[s][0]
    ys=np.sum(y, axis=1)
    for j in range(m):
        y[j]=y[j]/ys[j]

    #M stage
    pi=np.transpose(e.dot(y))/m
    mu = np.zeros([k, 2])
    for s in range(k):
        for j in range(m):
            mu[s]=mu[s]+y[j][s]*data[j]
    mu= mu/(pi*m)
    #print('1')
    cov = np.zeros([k, 2, 2])
    for s in range(k):
        for j in range(m):
            vec=data[j]-mu[s]
            temp=np.zeros([2,2])
            for st in range(2):
                for jt in range(2):
                    temp[st][jt]=vec[st]*vec[jt]
            cov[s]=cov[s]+y[j][s]*(temp)

    for s in range(k):
        cov[s]=cov[s]/pi[s][0]/m
```

```
print("After the two step EM algorithm, the parameter pi is:")
print(pi)
print("The average of the two cluster(mu) are:")
print(mu)
print ("The covariance matrix of the two clusters are:")
print(cov)
p = np.argmax(y, axis=1)
index=np.expand_dims(p, axis=1)
showrez(index, data, k, m)
```