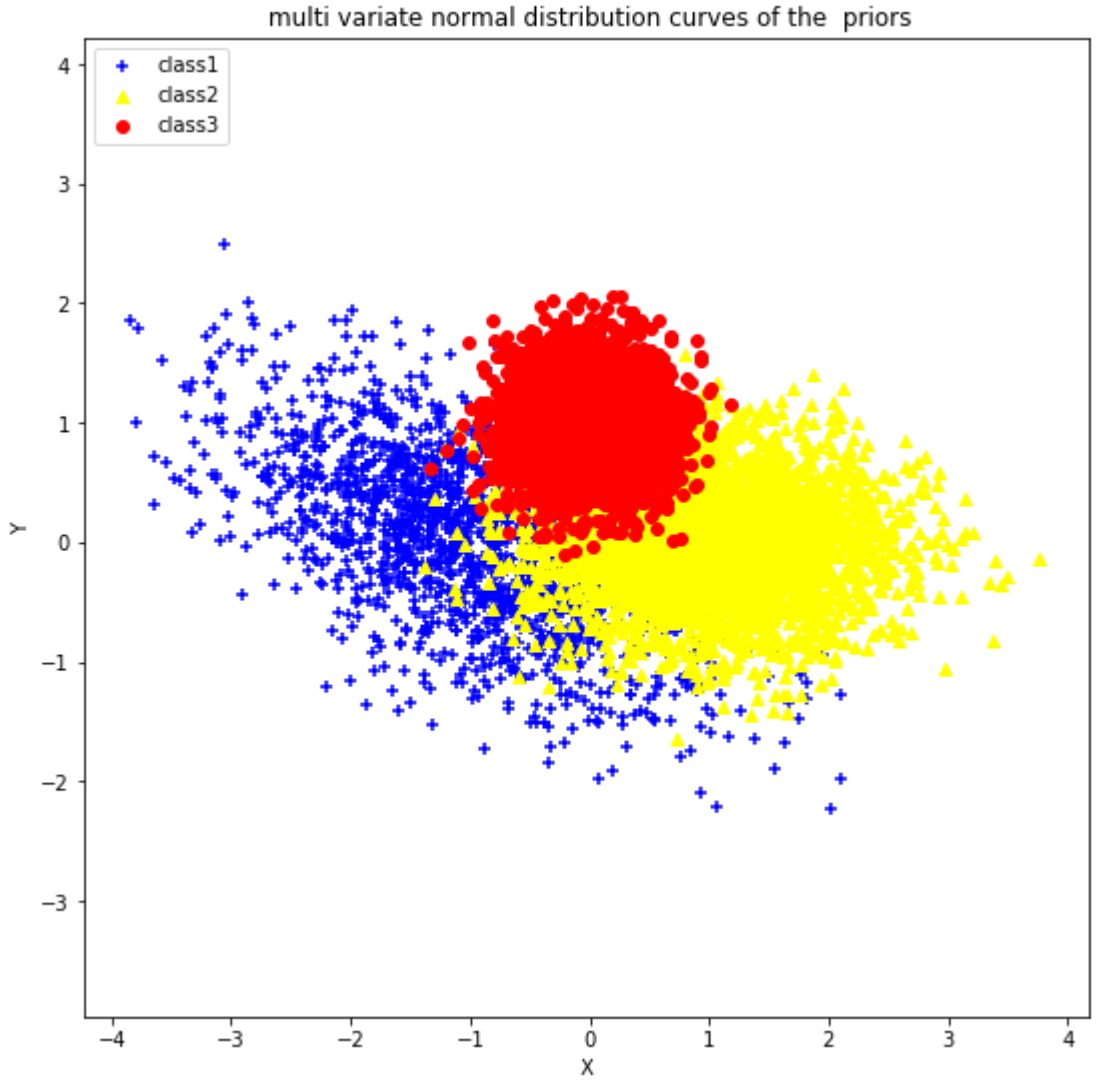
```
  import numpy as np
import pandas as pd
import matplotlib as mpl
import matplotlib.pyplot as plt
import seaborn as sns
import scipy.linalg as scln
from scipy.linalg import inv
from scipy.stats import multivariate_normal as mvn
#r=mvn.rvs(mean=mu,cov=cov,size=400)
# #r1=mvn.rvs(mean=mu1,cov=cov,size=400)


def samplegenerator(size,p):
  t=0
  t1=0
  b=np.random.choice(3,size,p=[p[0],p[1],p[2]])
  for i in b:
    if i==1:
      t=t+1
    if i==2:
      t1=t1+1
  size1=[10000-t-t1,t,t1]
  return size1


def multivariate_gaussian(mean,cov,sample):
  x=scln.cholesky(cov)
  Z=np.random.normal(loc=0, scale=1,size=(sample,cov.shape[0]))
  return (Z.dot(x)+mean)


def plot(r1,r2,r3):
  plt.figure(figsize=(9,9))
  plt.scatter(r1[0,:],r1[1,:],marker='+',label='class1',color='blue')
  plt.scatter(r2[0,:],r2[1,:],label='class2', marker = "^",color='YELLOW')
  plt.scatter(r3[0,:],r3[1,:],label='class3', marker = "o",color='red')
  plt.axis('equal')
  plt.title(' multi variate normal distribution curves of the  priors')
  plt.xlabel('X')
  plt.ylabel('Y')
  plt.legend(loc='upper left')
  plt.show()


p=[0.15,0.35,0.5]

x=samplegenerator(10000,p)

mu1=np.array([-1,0])
cov1=0.1*(np.array([[10,-4],[-4,5]]))
mu2=np.array([1,0])
cov2=0.1*(np.array([[5,0],[0,2]]))
mu3=np.array([0,1])
cov3=0.1*(np.eye(2, dtype=int))
r1= multivariate_gaussian(mu1,cov1,x[0]).T
r2=multivariate_gaussian(mu2,cov2,x[1]).T
r3=multivariate_gaussian(mu3,cov3,x[2]).T
plot(r1,r2,r3)
```




```
x[0] # number of samples in class 1
```

```
1595
```

```
x[1] #number of samples in class2
```

```
3477
```

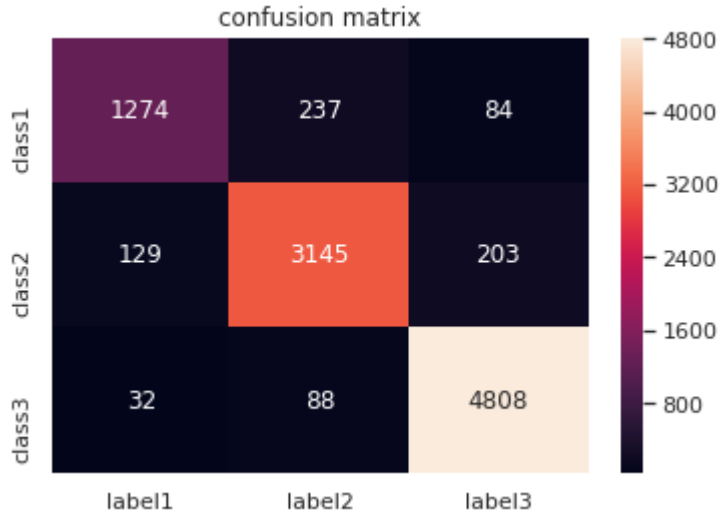
```
x[2] #number of samples in class3
```

```
4928
```

```
px1=0.15
px2=0.35
px3=0.50
sigma1=0.1*(np.array([[10,-4],[-4,5]]))
sigmainv1=np.linalg.inv(sigma1)
sigma2=0.1*(np.array([[5,0],[0,2]]))
sigmainv2=np.linalg.inv(sigma2)
sigma3=0.1*(np.eye(2, dtype=int))
sigmainv3=np.linalg.inv(sigma3)
```

```
W1=(-1/2)*sigmainv1
w1=sigmainv1@(mu1.reshape(2,1))
w10=(-1/2)*(mu1.reshape(2,1).T)@sigmainv1@mu1.reshape(2,1)-0.5*np.log(np.linalg.det(sigma1))+np.log(
W2=-0.5*sigmainv2
w2=sigmainv2@(mu2.reshape(2,1))
w20=(-1/2)*(mu2.reshape(2,1).T)@sigmainv2@mu2.reshape(2,1)-0.5*np.log(np.linalg.det(sigma2))+np.log(
W3=-0.5*sigmainv3
w3=sigmainv3@(mu3.reshape(2,1))
w30=(-1/2)*(mu3.reshape(2,1).T)@sigmainv3@mu3.reshape(2,1)-0.5*np.log(np.linalg.det(sigma3))+np.log(

g11=np.diag(r1.T@W1@r1)+w1.T@r1+w10
g12=np.diag(r2.T@W1@r2)+w1.T@r2+w10
g13=np.diag(r3.T@W1@r3)+w1.T@r3+w10
g23=np.diag(r3.T@W2@r3)+w2.T@r3+w20
g22=np.diag(r2.T@W2@r2)+w2.T@r2+w20
g21=np.diag(r1.T@W2@r1)+w2.T@r1+w20
g33=np.diag(r3.T@W3@r3)+w3.T@r3+w30
g32=np.diag(r2.T@W3@r2)+w3.T@r2+w30
g31=np.diag(r1.T@W3@r1)+w3.T@r1+w30


k11=((g11>g21)&(g11>g31))
k21=((g21>g11)&(g21>g31))
k31=((g31>g21)&(g31>g11))

k12=((g12>g22)&(g12>g32))
k22=((g22>g12)&(g22>g32))
k32=((g32>g22)&(g32>g12))

k13=((g13>g23)&(g13>g33))
k23=((g23>g13)&(g23>g33))
k33=((g33>g23)&(g33>g13))


a=np.array([[k11.sum(),k21.sum(),k31.sum()],[k12.sum(),k22.sum(),k32.sum()],[k13.sum(),k23.sum(),k33
import seaborn as sns; sns.set()
data=a
col = ['label1','label2','label3']
row=['class1','class2','class3']
ax = sns.heatmap(data,xticklabels=col,yticklabels=row,annot=True, fmt="d")

plt.title('confusion matrix')
```

```
Text(0.5, 1.0, 'confusion matrix')
```

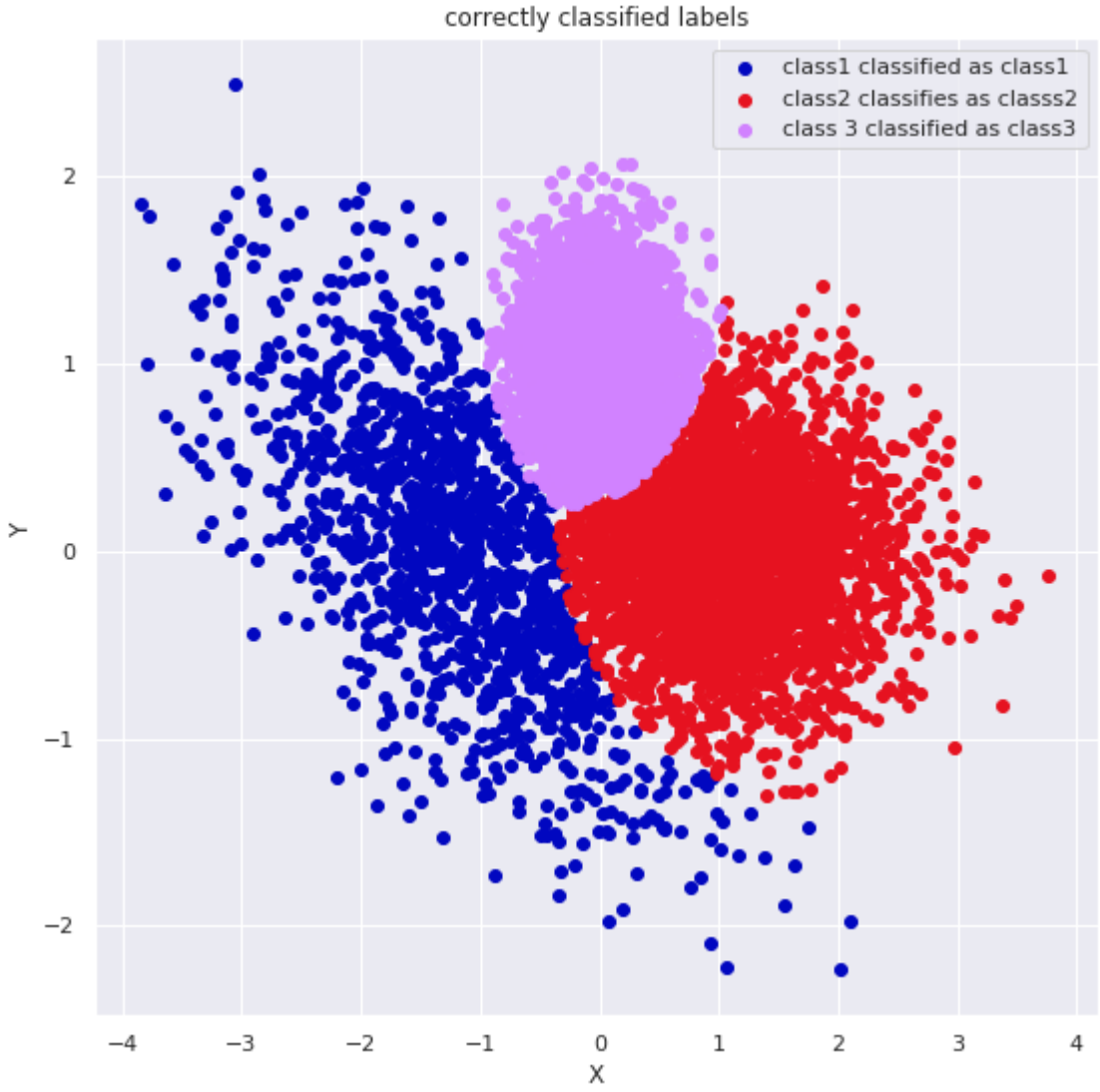



```
plt.figure(figsize=(9,9))
plt.scatter(r1[0,np.where((g11>g21)&(g11>g31))[1]],r1[1,np.where((g11>g21)&(g11>g31))[1]],label="cla
plt.scatter(r1[0,np.where((g21>g11)&(g21>g31))[1]],r1[1,np.where((g21>g11)&(g21>g31))[1]],label="cla
```

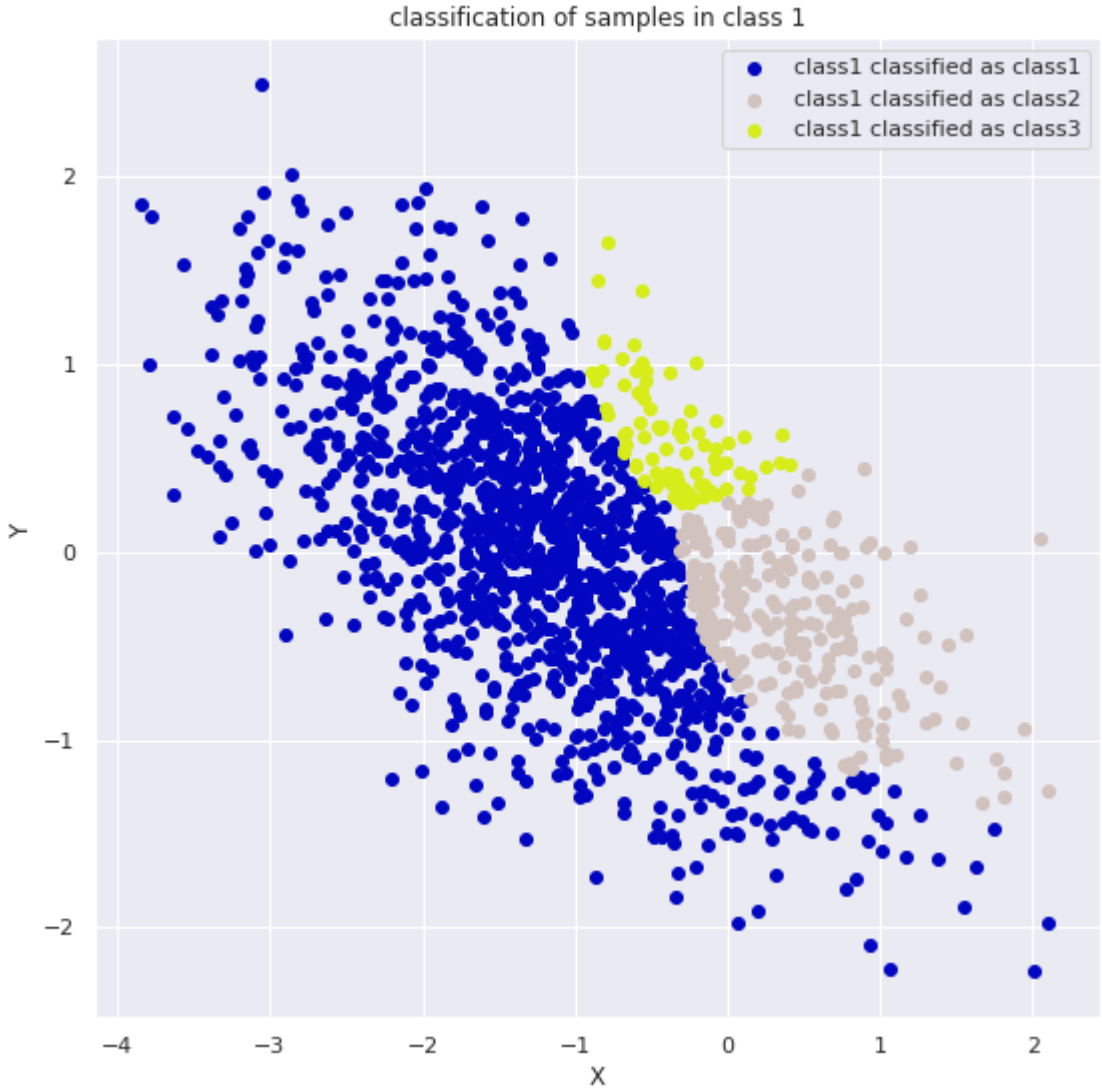
```
plt.scatter(r1[0,np.where((g31>g21)&(g31>g11))[1]],r1[1,np.where((g31>g21)&(g31>g11))[1]],label="cla
plt.scatter(r2[0,np.where((g12>g22)&(g12>g32))[1]],r2[1,np.where((g12>g22)&(g12>g32))[1]],label="cla
plt.scatter(r2[0,np.where((g22>g12)&(g22>g32))[1]],r2[1,np.where((g22>g12)&(g22>g32))[1]],label="cla
plt.scatter(r2[0,np.where((g32>g22)&(g32>g12))[1]],r2[1,np.where((g32>g22)&(g32>g12))[1]],label="cla
plt.scatter(r3[0,np.where((g13>g23)&(g13>g33))[1]],r3[1,np.where((g13>g23)&(g13>g33))[1]],label="cla
plt.scatter(r3[0,np.where((g23>g13)&(g23>g33))[1]],r3[1,np.where((g23>g13)&(g23>g33))[1]],label="cla
plt.scatter(r3[0,np.where((g33>g23)&(g33>g13))[1]],r3[1,np.where((g33>g23)&(g33>g13))[1]],label="cla
plt.xlabel('X')
plt.ylabel('Y')
plt.legend(loc='best')
plt.title('classification of the labels into different classes')
plt.show()
```


```
plt.figure(figsize=(9,9))
plt.scatter(r1[0,np.where((g21>g11)&(g21>g31))[1]],r1[1,np.where((g21>g11)&(g21>g31))[1]],label="cla
plt.scatter(r1[0,np.where((g31>g21)&(g31>g11))[1]],r1[1,np.where((g31>g21)&(g31>g11))[1]],label="cla
plt.scatter(r2[0,np.where((g12>g22)&(g12>g32))[1]],r2[1,np.where((g12>g22)&(g12>g32))[1]],label="cla
plt.scatter(r2[0,np.where((g32>g22)&(g32>g12))[1]],r2[1,np.where((g32>g22)&(g32>g12))[1]],label="cla
plt.scatter(r3[0,np.where((g13>g23)&(g13>g33))[1]],r3[1,np.where((g13>g23)&(g13>g33))[1]],label="cla
plt.scatter(r3[0,np.where((g23>g13)&(g23>g33))[1]],r3[1,np.where((g23>g13)&(g23>g33))[1]],label="cla
plt.xlabel('X')
plt.ylabel('Y')
plt.legend(loc='best')
plt.title('errors in classification')
plt.show()
```


```
plt.figure(figsize=(9,9))
plt.scatter(r1[0,np.where((g11>g21)&(g11>g31))[1]],r1[1,np.where((g11>g21)&(g11>g31))[1]],label="cla
plt.scatter(r2[0,np.where((g22>g12)&(g22>g32))[1]],r2[1,np.where((g22>g12)&(g22>g32))[1]],label="cla
plt.scatter(r3[0,np.where((g33>g23)&(g33>g13))[1]],r3[1,np.where((g33>g23)&(g33>g13))[1]],label="cla
plt.xlabel('X')
plt.ylabel('Y')
plt.legend(loc='best')
plt.title('correctly classified labels')
plt.show()
```

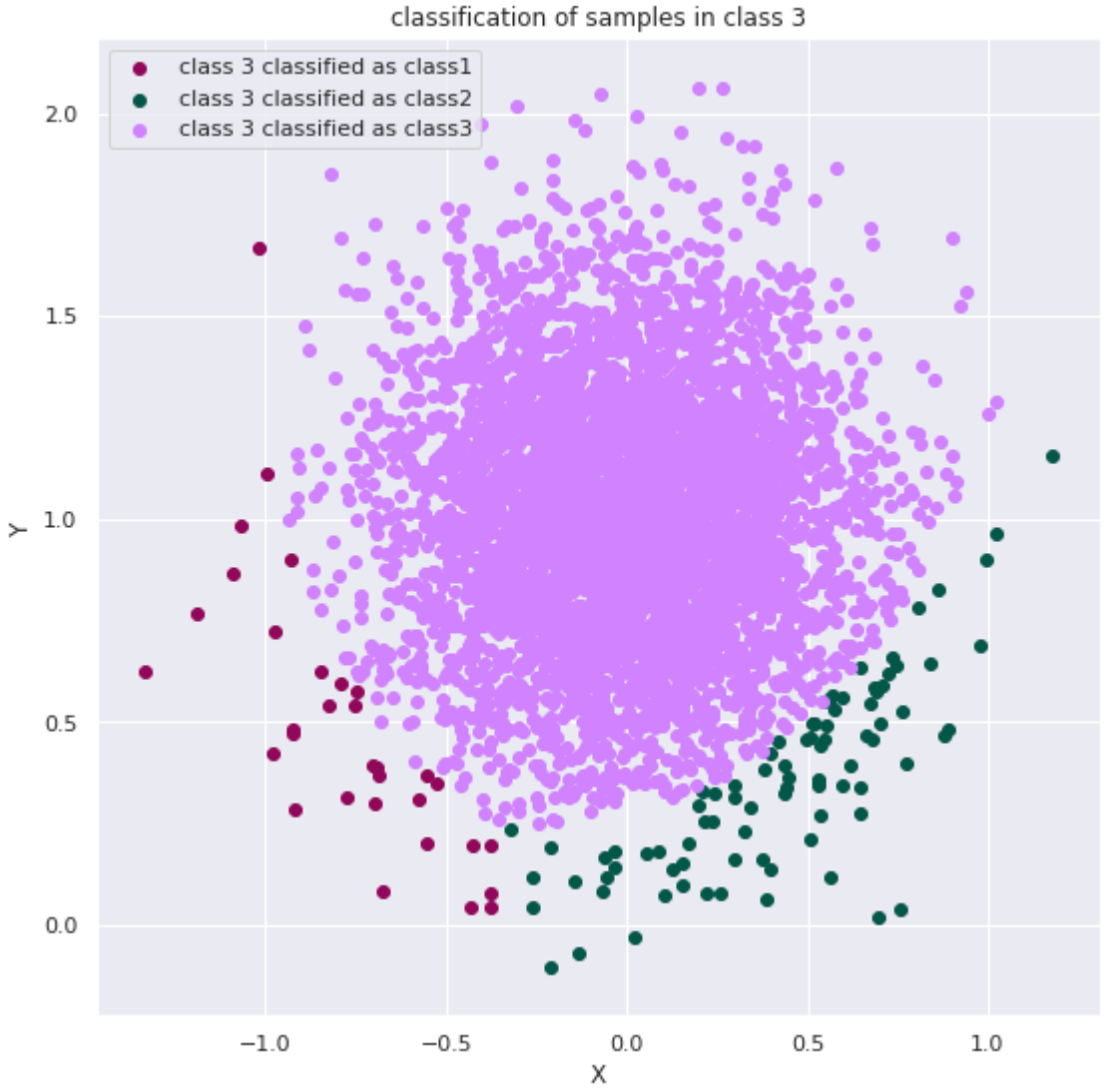



```
plt.figure(figsize=(9,9))
plt.scatter(r1[0,np.where((g11>g21)&(g11>g31))[1]],r1[1,np.where((g11>g21)&(g11>g31))[1]],label="cla
plt.scatter(r1[0,np.where((g21>g11)&(g21>g31))[1]],r1[1,np.where((g21>g11)&(g21>g31))[1]],label="cla
plt.scatter(r1[0,np.where((g31>g21)&(g31>g11))[1]],r1[1,np.where((g31>g21)&(g31>g11))[1]],label="cla
plt.xlabel('X')
plt.ylabel('Y')
plt.legend(loc='best')
plt.title('classification of samples in class 1')
plt.show()
```




```
plt.figure(figsize=(9,9))
plt.scatter(r2[0,np.where((g12>g22)&(g12>g32))[1]],r2[1,np.where((g12>g22)&(g12>g32))[1]],label="cla
plt.scatter(r2[0,np.where((g22>g12)&(g22>g32))[1]],r2[1,np.where((g22>g12)&(g22>g32))[1]],label="cla
plt.scatter(r2[0,np.where((g32>g22)&(g32>g12))[1]],r2[1,np.where((g32>g22)&(g32>g12))[1]],label="cla
plt.xlabel('X')
plt.ylabel('Y')
plt.legend(loc='best')
plt.title('classification of samples in class 2')
plt.show()
```


```
plt.figure(figsize=(9,9))
plt.scatter(r3[0,np.where((g13>g23)&(g13>g33))[1]],r3[1,np.where((g13>g23)&(g13>g33))[1]],label="cla
plt.scatter(r3[0,np.where((g23>g13)&(g23>g33))[1]],r3[1,np.where((g23>g13)&(g23>g33))[1]],label="cla
plt.scatter(r3[0,np.where((g33>g23)&(g33>g13))[1]],r3[1,np.where((g33>g23)&(g33>g13))[1]],label="cla
plt.xlabel('X')
plt.ylabel('Y')
plt.legend(loc='best')
plt.title('classification of samples in class 3')
plt.show()
```




```
#total number of points misclassified by the classifier
misclassified_points=k21.sum()+k31.sum()+k12.sum()+k32.sum()+k13.sum()+k23.sum()
print("total number of points misclassified by the classifier is {}".format(misclassified_points))
```

```
total number of points misclassified by the classifier is 773
```

```
#probability of error estimate
p_err=misclassified_points/10000
print("probability of error of the classifier is {}".format(p_err))
```

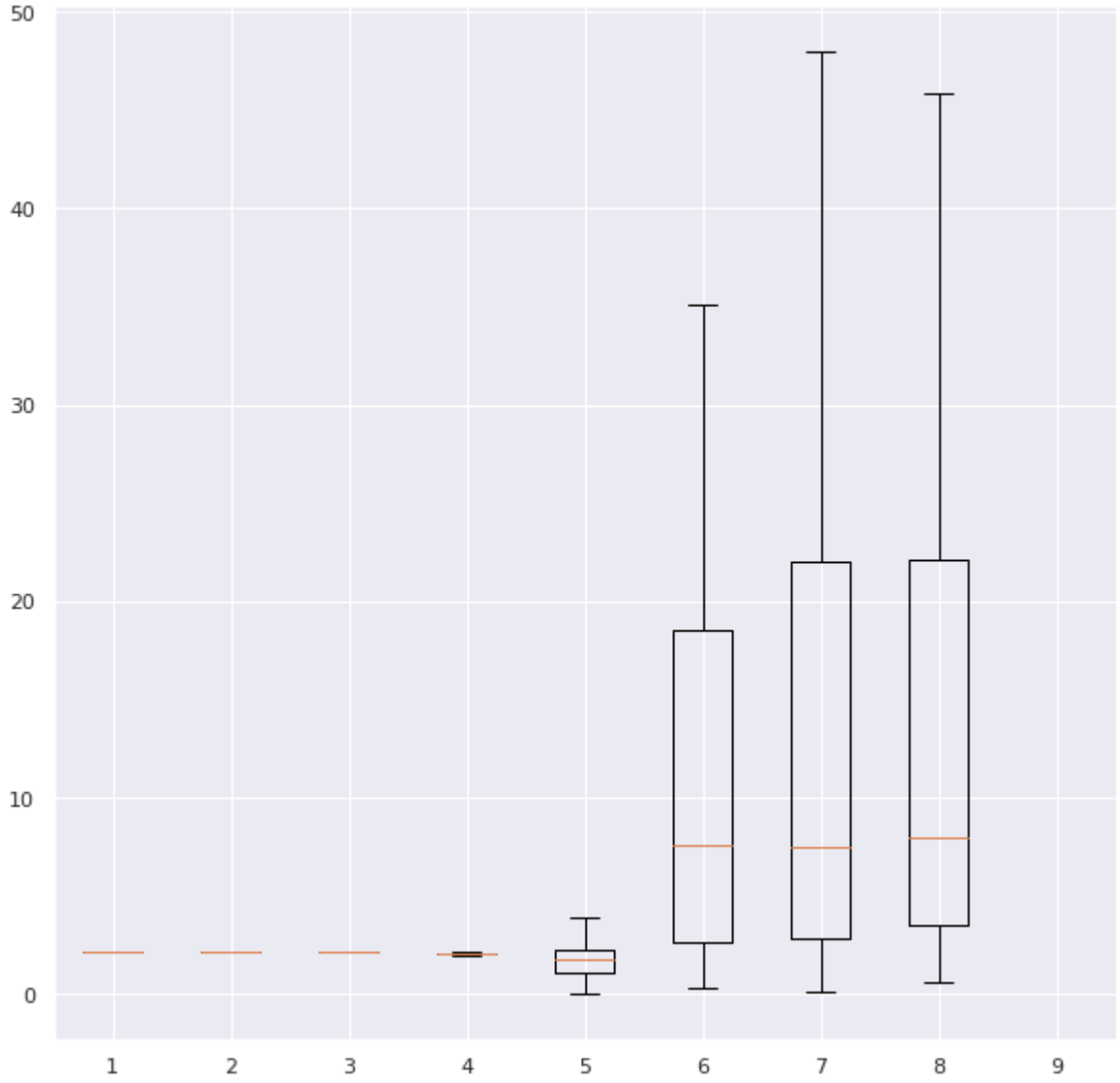
```
probability of error of the classifier is 0.0773
```

question3

```
%matplotlib inline
l2=[]
for i in range (-4,4):

  for j in range(100):
    N=10
    v=np.random.normal(loc=0,scale=1,size=10).reshape(10,1)
    x=np.random.uniform(-1,1,(10,4))
    wtrue=np.array([1,-1,-0.25,0.25]).reshape(4,1)
    ii=x[:,2]**3
    j=x[:,2]**2
    k=x[:,2]
    x[:,3]=1
    l=x[:,3]
    m=np.vstack((ii,j,k,l))
    y=m.T@wtrue+v

    wmap=(np.linalg.inv((1/(10**(i)))**2*(np.identity(4))+np.matmul(m,m.T))@m@y)

    l2.append(np.sum((wmap-wtrue)**2))
  print("gamma = ",i)
  print("\n")
  print("\t Q2 quantile of l2 : ", np.quantile(l2, .50))
  print("\t Q1 quantile of l2 : ", np.quantile(l2, .25))
  print("\t Q3 quantile of l2 : ", np.quantile(l2, .75))
  print("\t 100th quantile of l2 : ", np.quantile(l2, .1))
  print("\n")

plt.figure(figsize=(10,10))
plt.boxplot([l2[0:100],l2[100:200],l2[200:300],l2[300:400],l2[400:500],l2[500:600],l2[600:700],l2[70
print
```

```
gamma =  -4


         Q2 quantile of l2 :  2.124999981795579
         Q1 quantile of l2 :  2.1249999617099133
         Q3 quantile of l2 :  2.124999994337321
         100th quantile of l2 :  2.12499992776657


gamma =  -3


         Q2 quantile of l2 :  2.1249999504670436
         Q1 quantile of l2 :  2.124997528372232
         Q3 quantile of l2 :  2.1249999911616024
         100th quantile of l2 :  2.1249942056983127


gamma =  -2


         Q2 quantile of l2 :  2.1249988749709203
         Q1 quantile of l2 :  2.124944943781433
         Q3 quantile of l2 :  2.1249999868756513
         100th quantile of l2 :  2.124613372804025


gamma =  -1


         Q2 quantile of l2 :  2.124996452073785
         Q1 quantile of l2 :  2.124430073999493
         Q3 quantile of l2 :  2.124999983261431
         100th quantile of l2 :  2.089958856800157


gamma =  0


         Q2 quantile of l2 :  2.1249951216478937
         Q1 quantile of l2 :  2.11065712218046
         Q3 quantile of l2 :  2.1249999886958033
         100th quantile of l2 :  1.7658252307674323


gamma =  1


         Q2 quantile of l2 :  2.124997418348534
         Q1 quantile of l2 :  2.111800420704065
         Q3 quantile of l2 :  2.125001294278727
         100th quantile of l2 :  1.6671538497497258


gamma =  2
```

```
        Q2 quantile of l2 :  2.1249998796979357
        Q1 quantile of l2 :  2.1190470964962227
        Q3 quantile of l2 :  2.5047760923741906
        100th quantile of l2 :  1.6362664027379024


gamma =  3


        Q2 quantile of l2 :  2.1249999686593934
        Q1 quantile of l2 :  2.124061134623456
        Q3 quantile of l2 :  4.25654532327811
        100th quantile of l2 :  1.6362664027379024


<function print>
```

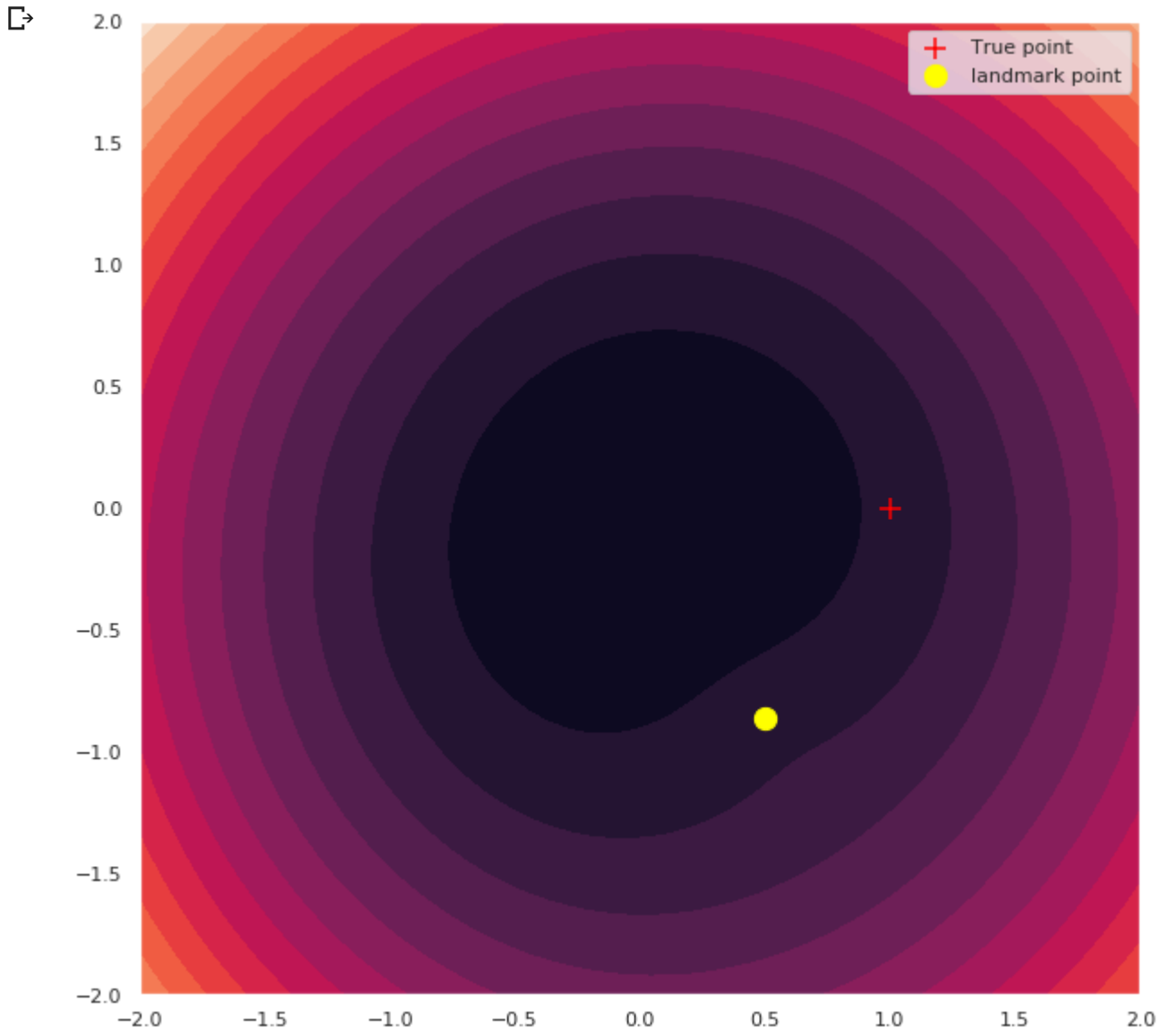


question 2

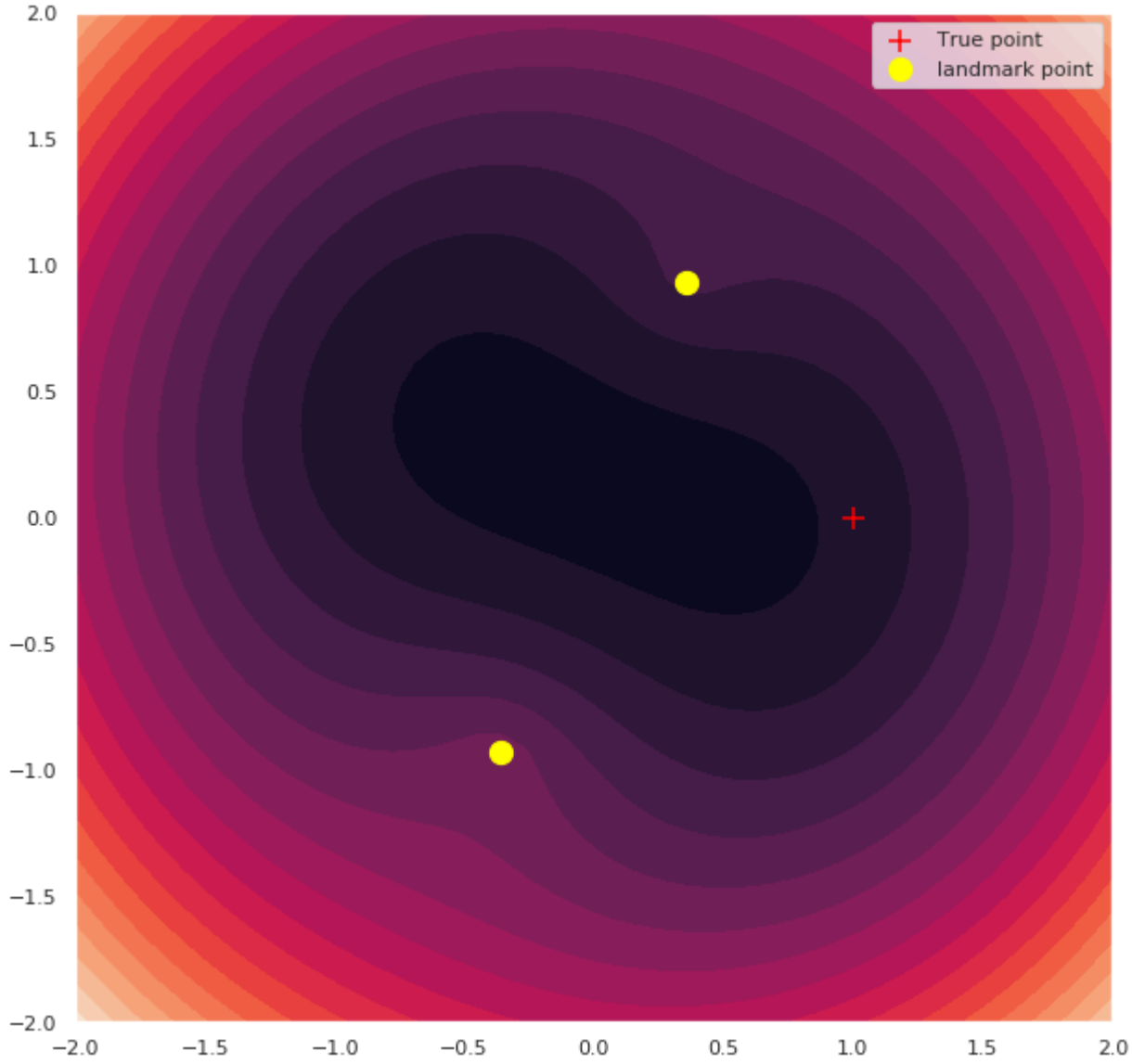
```
def question2(q):


  xx=np.linspace(-2,2,50)
  yy=np.linspace(-2,2,50)
  r = 1
  x = []
  y = []
  dtheta = np.random.randint(360) #random angle generation
  for i in range(q):
    x.append(math.cos(dtheta + 2*math.pi*i/q)) #appending the random x coordinate in a list
    y.append(math.sin(dtheta + 2*math.pi*i/q)) #appending the random y coordinate in a list

  # angle

  alpha = 0
  # random radius
  r = 1
  # calculating coordinates of the true point
  xd = r * math.cos(alpha)
  yd = r * math.sin(alpha)
  dx=[]
  dy=[]
  for i in range (len(x)):
    dx.append((x[i]-xd)**2)
    dy.append((y[i]-yd)**2)
  #range ri is given by
  noise=np.random.normal(0,0.3,1)

  r1=[]
  for i in range (len(x)):
    r1.append(math.sqrt(dx[i]+dy[i])+noise)


  #calculating map estimate

  sigma=0.3
  sigmax=0.23 #according to the question
  sigmay=0.25 #according to the question
  mapest=[]#storing all the mapestimate values in this list
  for i in range (0,len(xx)):
    for j in range(0,len(yy)):
      kkk=0
      for k in range(len(r1)):
        kkk=kkk+((r1[k]-math.sqrt((x[k]-xx[i])**2+(y[k]-yy[j])**2))**2)
      mapest.append(((1/sigma)**2)*kkk+(xx[i])**2/(sigmax)**2+(yy[j])**2/(sigmay)**2)

  h=[] #creating this list for plotting data on

  for i in range(0,len(xx)):
    for j in range(0,len(yy)):
      h.append([xx[i],yy[j]])

  h=np.array(h) # array intialization

  #
  dd=np.array(h[:,0])
  dd=dd.reshape(50,50)

  tt=np.array(h[:,1])
```

```
  tt=tt.reshape(50,50)

  bb=np.array(mapest)
  bb=bb.reshape(50,50)

  fig,ax=plt.subplots(1,1,figsize=(10,10))
  cp=ax.contourf(dd,tt,bb,levels=20)
  plt.scatter(xd,yd,marker='+',color='red',label='True point', s=150)
  plt.scatter(x,y,marker="o",color='yellow',label='landmark point',s=150)
  plt.legend()
  plt.show()
```

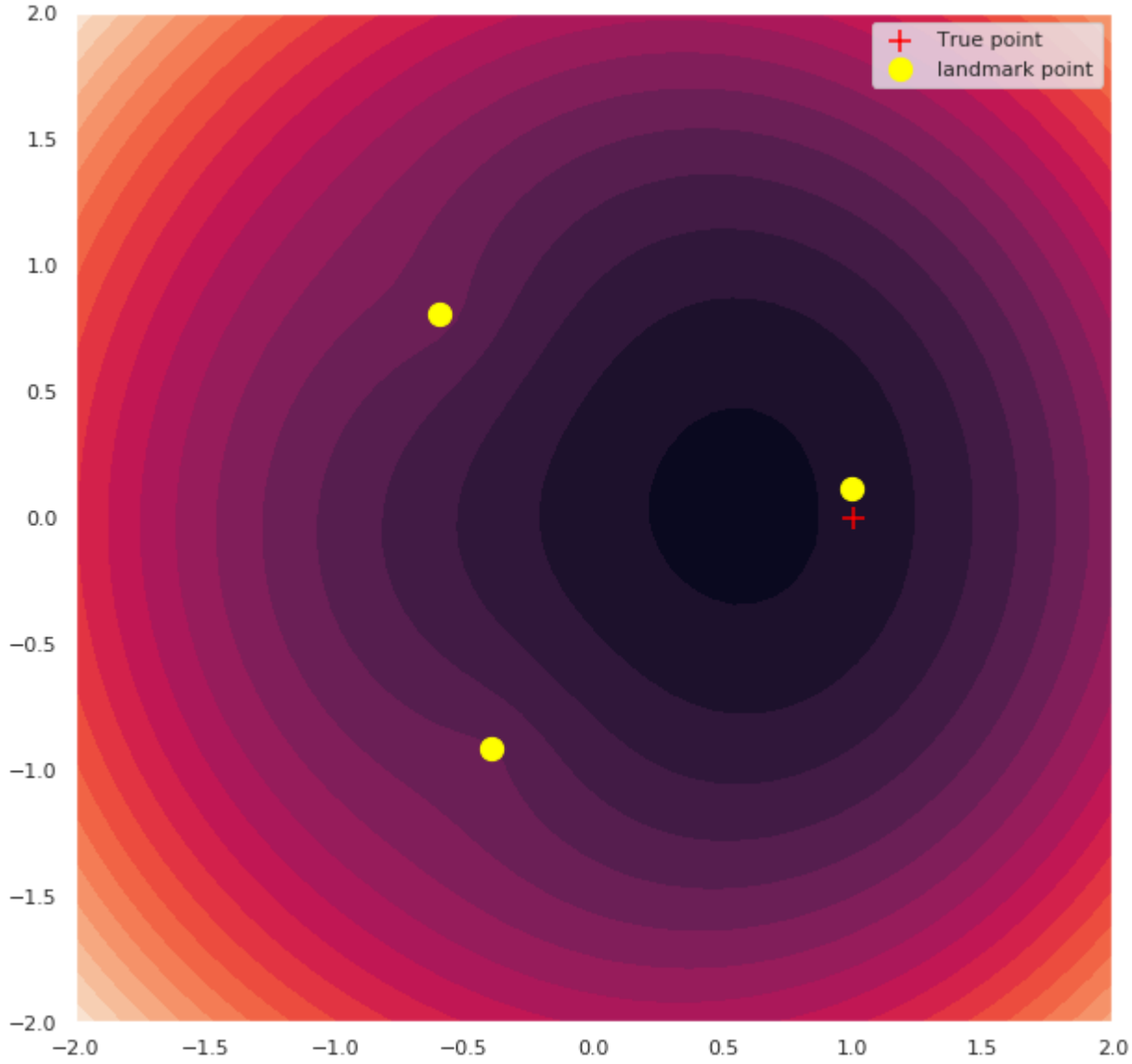
```
question2(1)  # when k is equal to 1
```



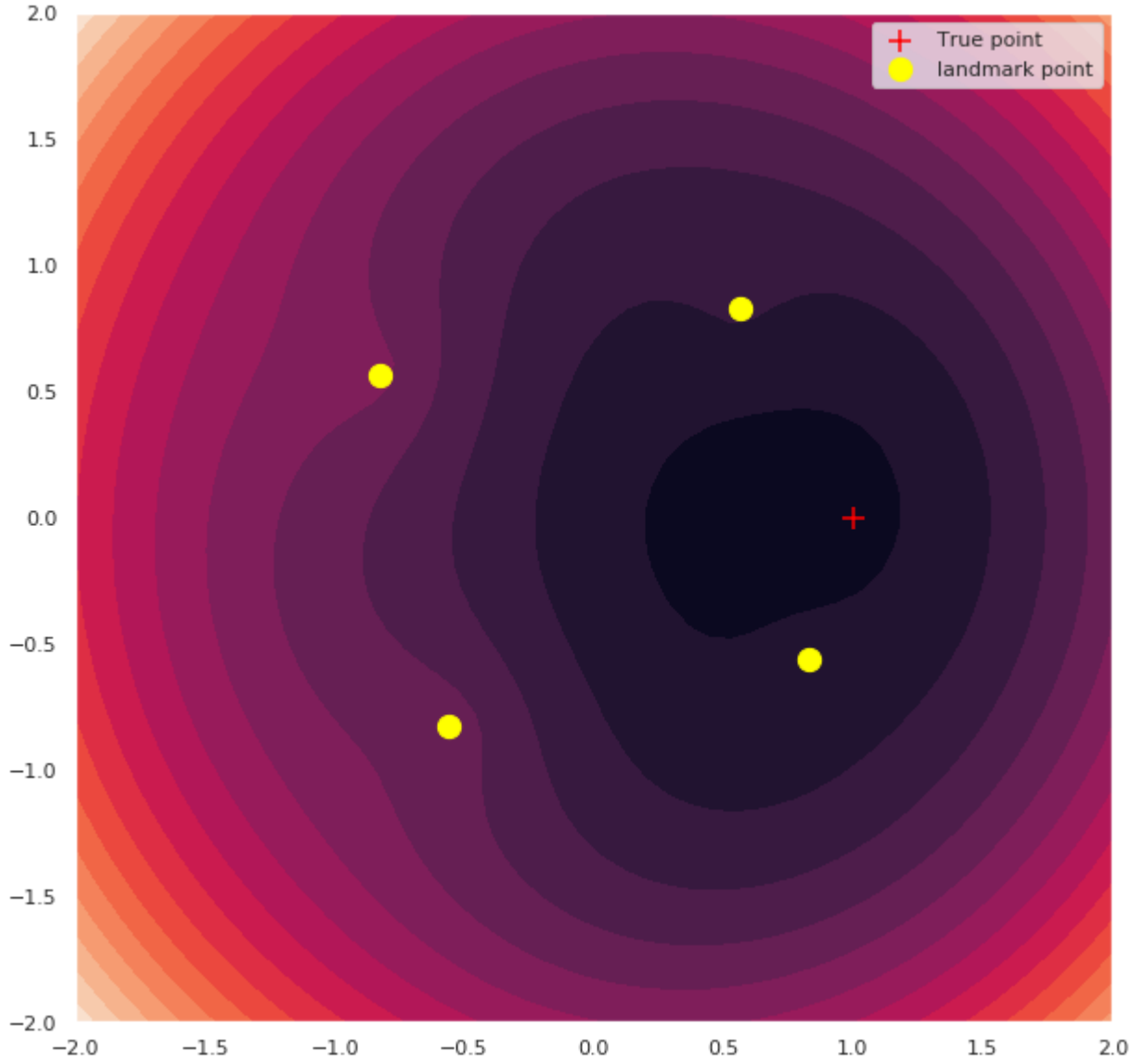

```
question2(2)
# number of landmarks =2
```




```
question2(3)
#number of landmarks is equal to 3
```




```
question2(4)
#number of landmarks is equal to 4
```




```
from google.colab import drive
drive.mount('/gdrive')
%cd /gdrive
```

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_id=9473189

Enter your authorization code:
··········
Mounted at /gdrive
/gdrive
```