**Answer 1** :

Code --------

# x1 & x2 in range (0,10)

import math

import numpy as np

import matplotlib.pyplot as plt

def multivariate\_gaussian\_dist(item, V\_inv,log\_det):

k = 2 # Rank of matrix

item = item.reshape(2,1)

mgd = ((-k/2)\*np.log(2\*(np.pi))) - log\_det/2 - ((np.matmul(np.matmul(item.T, V\_inv), item))/2)

return mgd

def get\_log\_det(eigenvalues,eigenvectors):

zero\_vec = np.zeros([1,2])

count = 0

full\_rank\_cov\_mat = np.empty([0,2])

non\_zero\_eigenvalues = np.empty([0])

# Reducing the p eigenvectors of dim p, to k eigen.Vecs of dim p.

for i in range(2):

# We take 1 as eigen-values very close to 0 need to be ignored. The rank is also satisfied when we take this condition.

if eigenvalues[i]>1:

non\_zero\_eigenvalues = np.append(non\_zero\_eigenvalues,eigenvalues[i])

full\_rank\_cov\_mat = np.append(full\_rank\_cov\_mat,eigenvectors[i].reshape(1,2),axis=0)

D = np.diag(non\_zero\_eigenvalues)

log\_det = 0

for eigenval in non\_zero\_eigenvalues:

log\_det += np.log(eigenval)

V = full\_rank\_cov\_mat.T.dot(D.dot(full\_rank\_cov\_mat))

V\_inv = np.linalg.pinv(V)

return log\_det, V\_inv

def case\_1():

u1 = np.array([3,3])

u2 = np.array([7,7])

cov = np.array([[3,0],[0,3]])

classes = []

class\_A = []

class\_B = []

while len(class\_A)<1000:

rand\_val = np.random.multivariate\_normal(u1, cov)

if rand\_val[0]>=0 and rand\_val[0]<=10 and rand\_val[1]>=0 and rand\_val[1]<=10:

class\_A.append(rand\_val)

class\_A = np.asarray(class\_A)

classes.append(class\_A)

while len(class\_B)<1000:

rand\_val = np.random.multivariate\_normal(u2, cov)

if rand\_val[0]>=0 and rand\_val[0]<=10 and rand\_val[1]>=0 and rand\_val[1]<=10:

class\_B.append(rand\_val)

class\_B = np.asarray(class\_B)

classes.append(class\_B)

classes = np.asarray(classes)

# Plot training data

plt.scatter(class\_A[:,0], class\_A[:,1], color="black")

plt.scatter(class\_B[:,0], class\_B[:,1], color="gray")

# 100 samples

test\_data\_x = []

test\_data\_y = []

test\_data\_x = np.random.uniform(0,10,100)

test\_data\_y = np.random.uniform(0,10,100)

# plt.scatter(test\_data\_x,test\_data\_y)

# plt.show()

max\_likelihood = np.zeros(shape=(100,2))

for j in range(2):

class\_label=float(j)

each\_class\_mean = np.mean(classes[j], axis=0).reshape([2,1])

eigenvectors, eigenvalues, V = np.linalg.svd(cov, full\_matrices=False)

eigenvalues = eigenvalues.reshape(eigenvalues.shape[0],1)

norm\_constant = math.sqrt(np.sum((eigenvalues.T).dot(eigenvalues)))

log\_det, V\_inv = get\_log\_det(eigenvalues, eigenvectors)

for k in range(len(test\_data\_x)):

item = np.asarray([test\_data\_x[k],test\_data\_y[k]]).reshape([2,1])

multivariate\_dist = multivariate\_gaussian\_dist(item-each\_class\_mean, V\_inv, log\_det)

max\_likelihood[k][j] = multivariate\_dist

max\_likelihood = np.asarray(max\_likelihood)

max\_likelihood\_indices = np.unravel\_index(np.argmax(max\_likelihood, axis=1), max\_likelihood.shape)

max\_likelihood\_indices = np.asarray(max\_likelihood\_indices)

classA\_plot\_x = []

classA\_plot\_y = []

classB\_plot\_x = []

classB\_plot\_y = []

for i in range(len(test\_data\_x)):

if(0==max\_likelihood\_indices[1][i]):

classA\_plot\_x.append(test\_data\_x[i])

classA\_plot\_y.append(test\_data\_y[i])

# plt.plot(test\_data\_x[i],test\_data\_y[i] ,color="green")

else:

classB\_plot\_x.append(test\_data\_x[i])

classB\_plot\_y.append(test\_data\_y[i])

# plt.plot(test\_data\_x[i],test\_data\_y[i],color="red")

plt.scatter(classA\_plot\_x, classA\_plot\_y, color="red")

plt.scatter(classB\_plot\_x, classB\_plot\_y, color="green")

plt.show()

def case\_2():

u1 = np.array([3,3])

u2 = np.array([7,7])

cov = []

cov.append(np.array([[3,1],[2,3]]))

cov.append(np.array([[7,2],[1,7]]))

classes = []

class\_A = []

class\_B = []

while len(class\_A)<1000:

rand\_val = np.random.multivariate\_normal(u1, cov[0])

if rand\_val[0]>=0 and rand\_val[0]<=10 and rand\_val[1]>=0 and rand\_val[1]<=10:

class\_A.append(rand\_val)

class\_A = np.asarray(class\_A)

classes.append(class\_A)

while len(class\_B)<1000:

rand\_val = np.random.multivariate\_normal(u2, cov[1])

if rand\_val[0]>=0 and rand\_val[0]<=10 and rand\_val[1]>=0 and rand\_val[1]<=10:

class\_B.append(rand\_val)

class\_B = np.asarray(class\_B)

classes.append(class\_B)

classes = np.asarray(classes)

# Plot training data

plt.scatter(class\_A[:,0], class\_A[:,1], color="black")

plt.scatter(class\_B[:,0], class\_B[:,1], color="gray")

# 100 samples

test\_data\_x = []

test\_data\_y = []

test\_data\_x = np.random.uniform(0,10,100)

test\_data\_y = np.random.uniform(0,10,100)

# plt.scatter(test\_data\_x,test\_data\_y)

# plt.show()

max\_likelihood = np.zeros(shape=(100,2))

for j in range(2):

class\_label=float(j)

each\_class\_mean = np.mean(classes[j], axis=0).reshape([2,1])

eigenvectors, eigenvalues, V = np.linalg.svd(cov[j], full\_matrices=False)

eigenvalues = eigenvalues.reshape(eigenvalues.shape[0],1)

norm\_constant = math.sqrt(np.sum((eigenvalues.T).dot(eigenvalues)))

log\_det, V\_inv = get\_log\_det(eigenvalues, eigenvectors)

for k in range(len(test\_data\_x)):

item = np.asarray([test\_data\_x[k],test\_data\_y[k]]).reshape([2,1])

multivariate\_dist = multivariate\_gaussian\_dist(item-each\_class\_mean, V\_inv, log\_det)

max\_likelihood[k][j] = multivariate\_dist

max\_likelihood = np.asarray(max\_likelihood)

max\_likelihood\_indices = np.unravel\_index(np.argmax(max\_likelihood, axis=1), max\_likelihood.shape)

max\_likelihood\_indices = np.asarray(max\_likelihood\_indices)

classA\_plot\_x = []

classA\_plot\_y = []

classB\_plot\_x = []

classB\_plot\_y = []

for i in range(len(test\_data\_x)):

if(0==max\_likelihood\_indices[1][i]):

classA\_plot\_x.append(test\_data\_x[i])

classA\_plot\_y.append(test\_data\_y[i])

# plt.plot(test\_data\_x[i],test\_data\_y[i] ,color="green")

else:

classB\_plot\_x.append(test\_data\_x[i])

classB\_plot\_y.append(test\_data\_y[i])

# plt.plot(test\_data\_x[i],test\_data\_y[i],color="red")

plt.scatter(classA\_plot\_x, classA\_plot\_y, color="red")

plt.scatter(classB\_plot\_x, classB\_plot\_y, color="green")

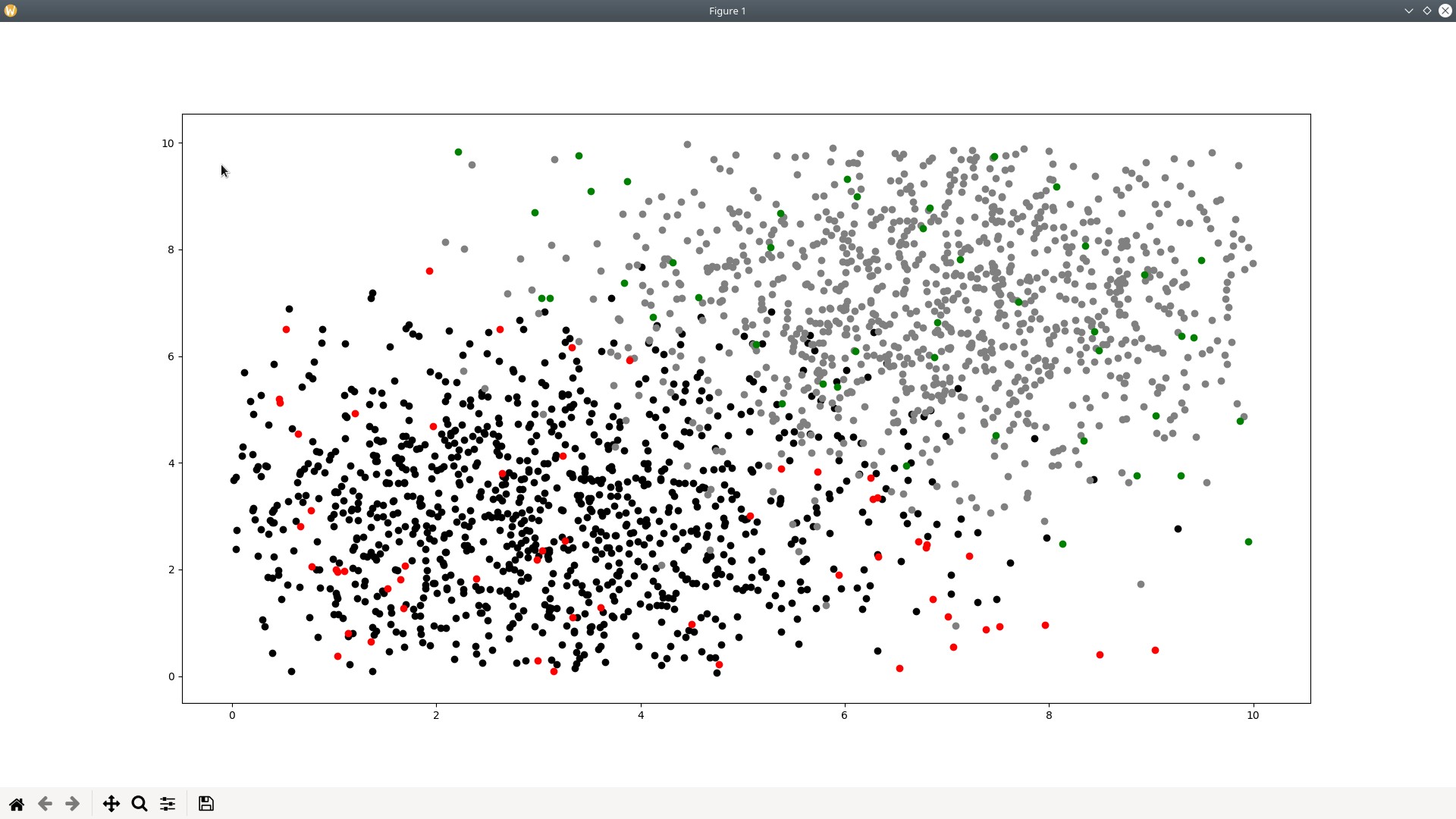
plt.show()

case\_1()

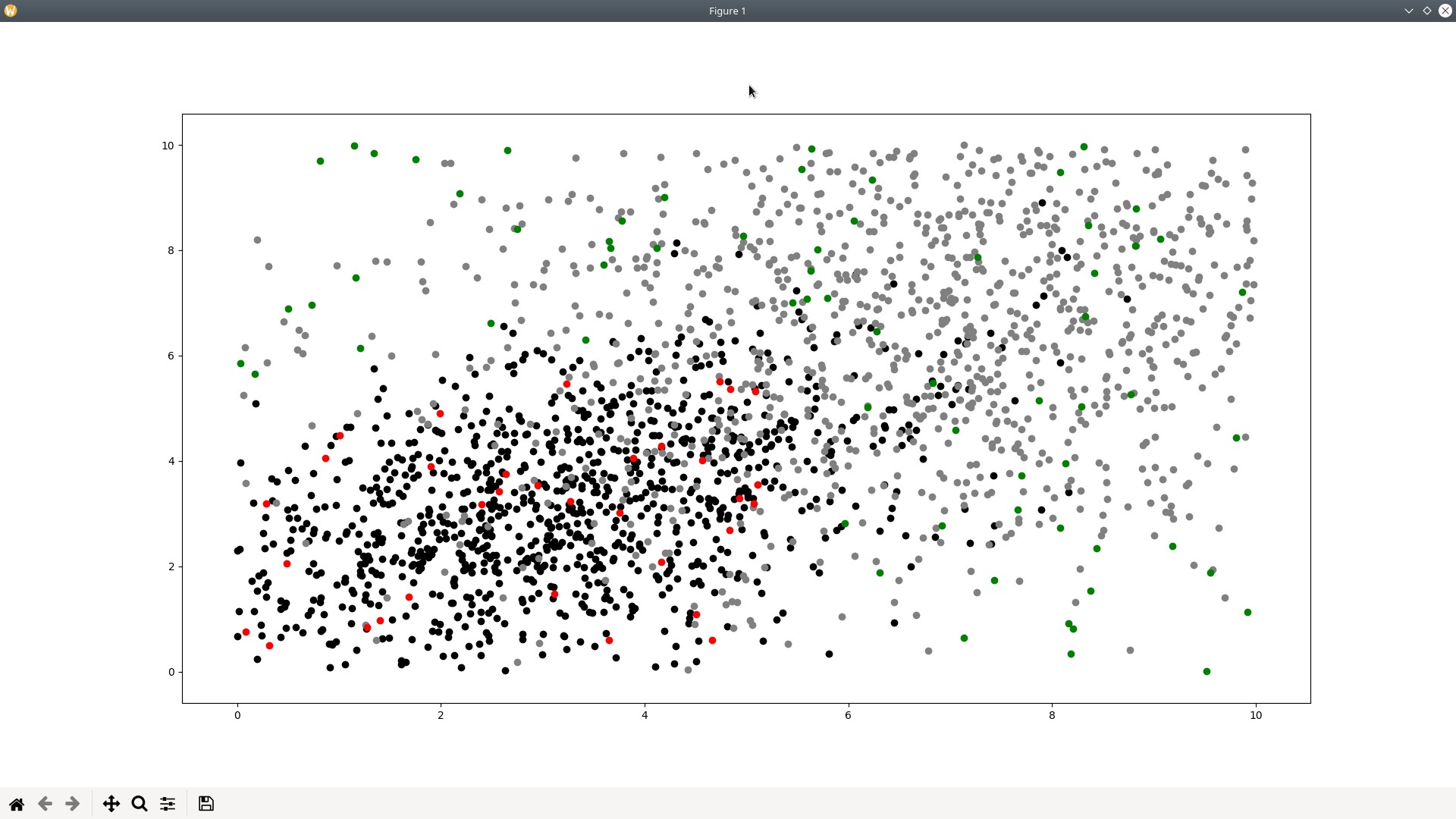
case\_2()

**PLOTS** ----

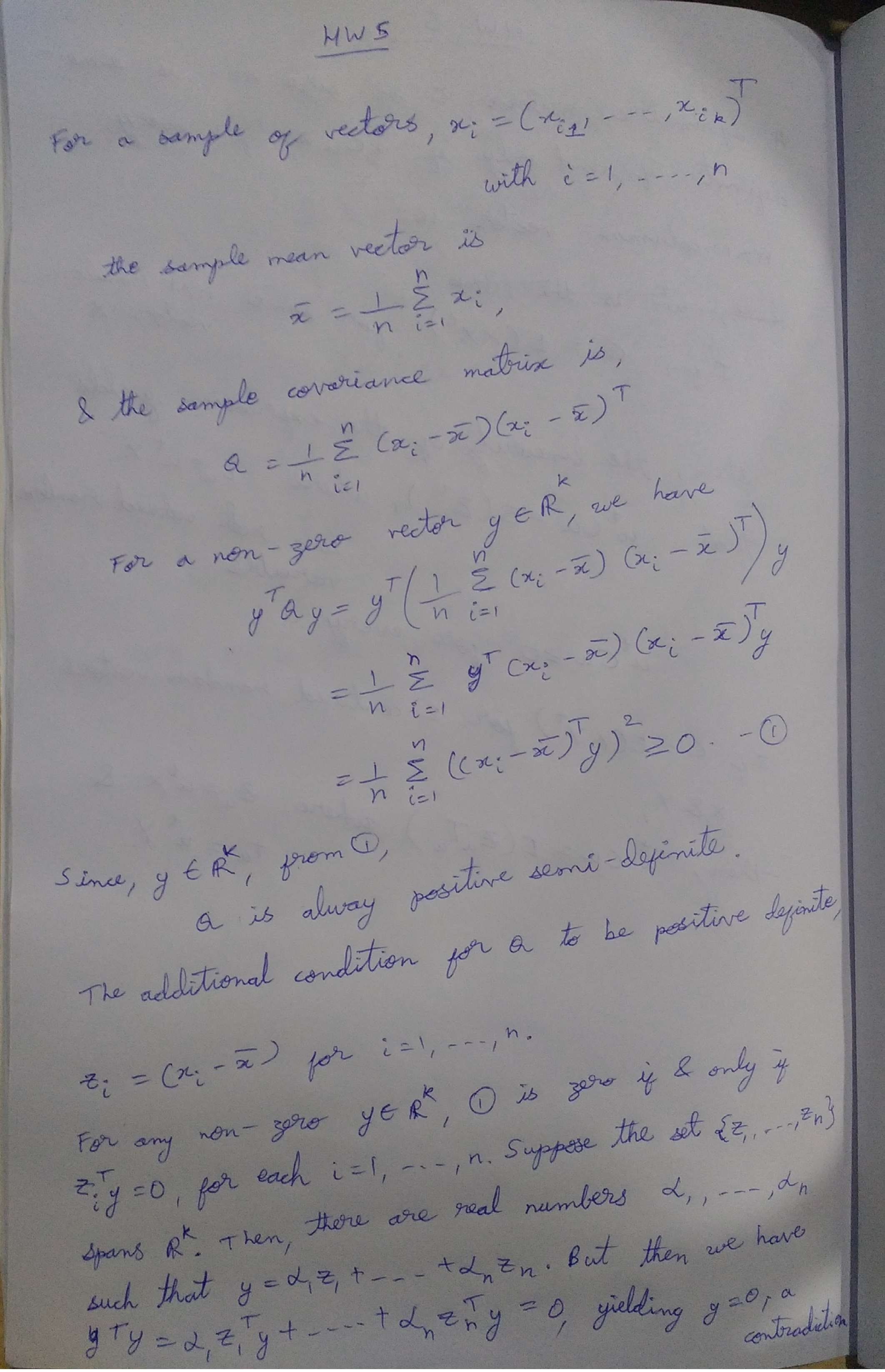
Case 1 :

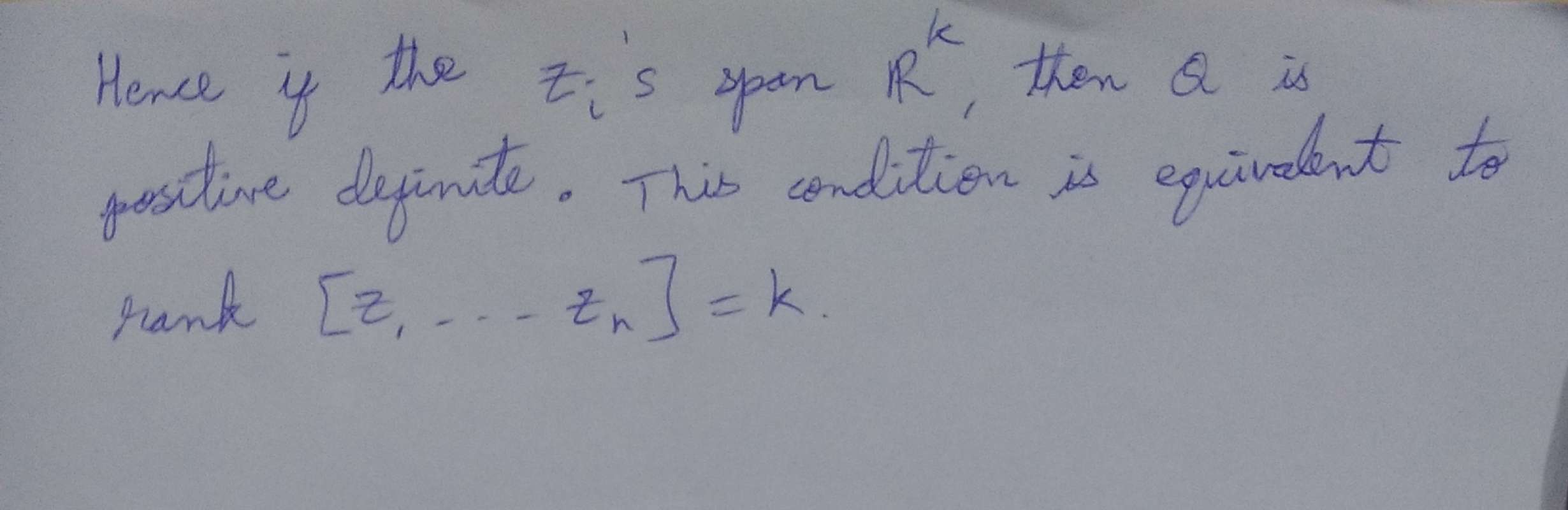
****

Case 2:



**Answer 3** :

****

****