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)
```

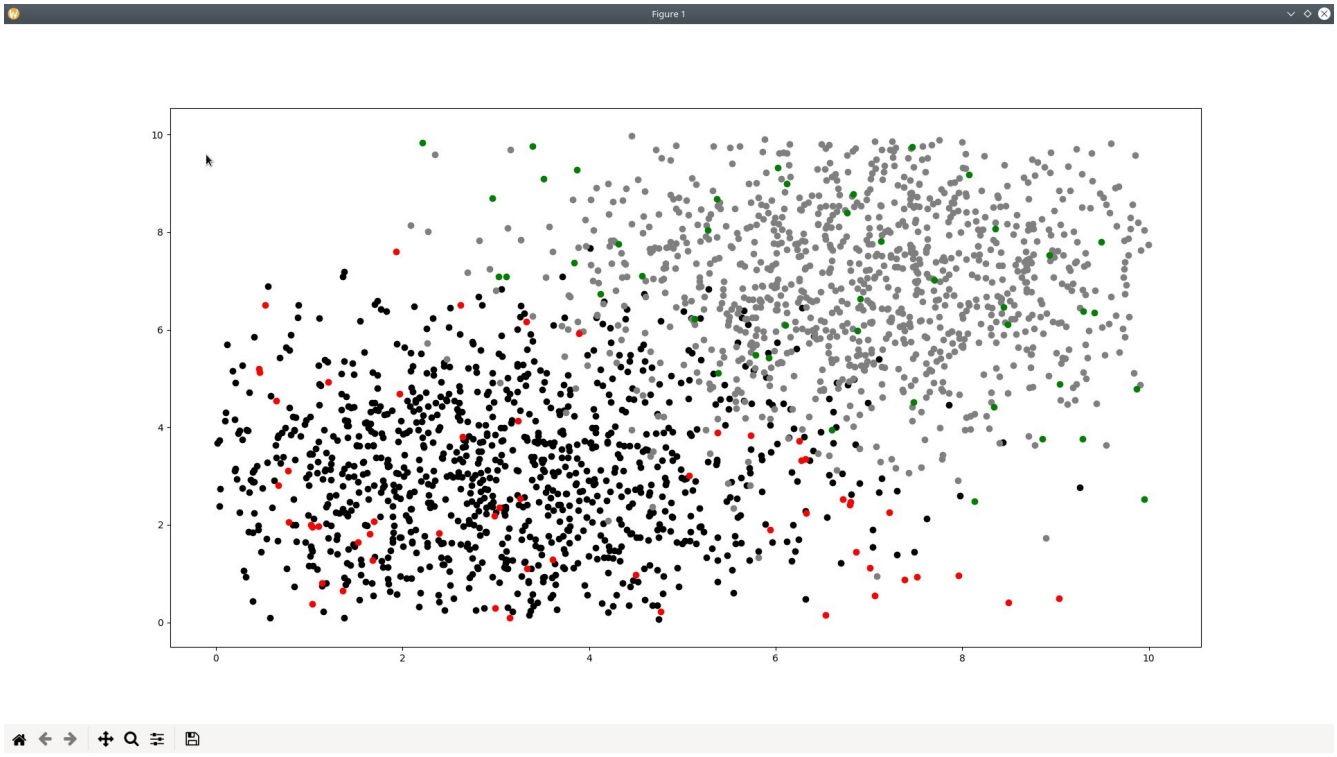
```
    classA_plot_x = []
    classA_plot_y = []
    classB_plot_x = []
    classB_plot_y = []
    for i in range(len(test_data_x)):
        if(0==max_likelihood_indices[1][i]):
            classA_plot_x.append(test_data_x[i])
            classA_plot_y.append(test_data_y[i])
            # plt.plot(test_data_x[i],test_data_y[i] ,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()
```

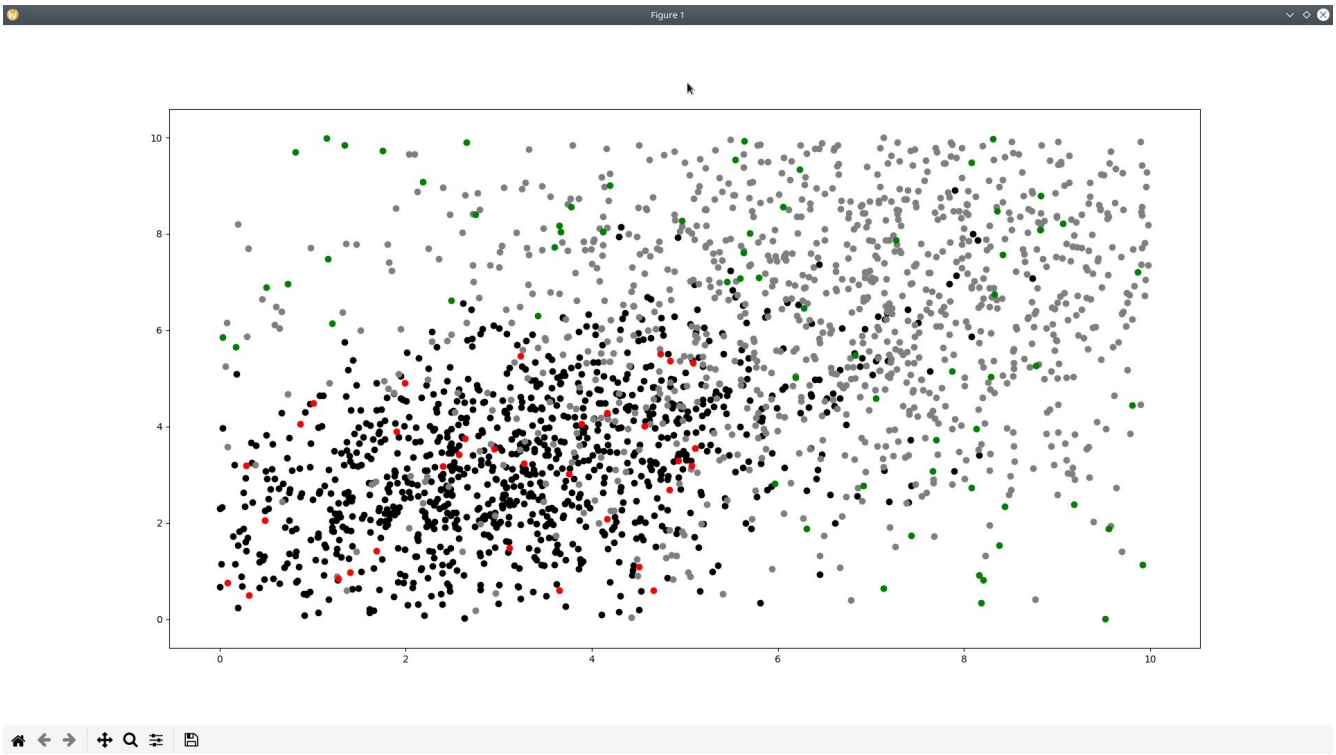
case_1()
case_2()

PLOTS ----

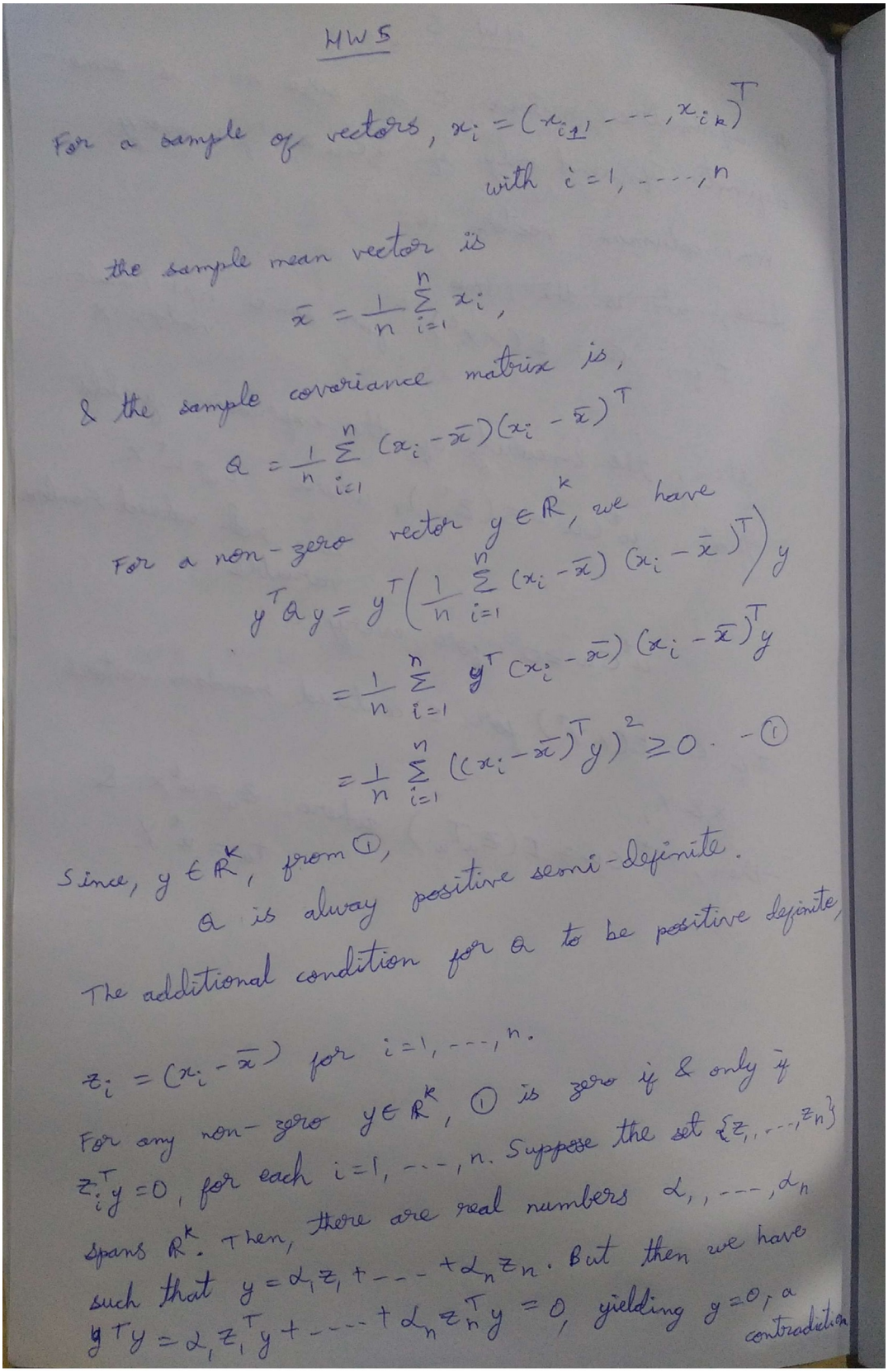
Case 1 :



Case 2:



Answer 3 :



HW 5

For a sample of vectors, $x_i = (x_{i1}, \dots, x_{ik})^T$ with $i = 1, \dots, n$

the sample mean vector is

$$\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i,$$

& the sample covariance matrix is,

$$Q = \frac{1}{n} \sum_{i=1}^{n} (x_i - \bar{x})(x_i - \bar{x})^T$$

For a non-zero vector $y \in \mathbb{R}^k$, we have

$$y^T Q y = y^T \left(\frac{1}{n} \sum_{i=1}^{n} (x_i - \bar{x})(x_i - \bar{x})^T \right) y$$

$$= \frac{1}{n} \sum_{i=1}^{n} y^T (x_i - \bar{x})(x_i - \bar{x})^T y$$

$$= \frac{1}{n} \sum_{i=1}^{n} ((x_i - \bar{x})^T y)^2 \geq 0 \quad \cdots (1)$$

Since, $y \in \mathbb{R}^k$, from (1),

Q is alway positive semi-definite.

The additional condition for Q to be positive definite,

$z_i = (x_i - \bar{x})$ for $i = 1, \dots, n$.

For any non-zero $y \in \mathbb{R}^k$, (1) is zero if & only if $z_i^T y = 0$, for each $i = 1, \dots, n$. Suppose the set $\{z_1, \dots, z_n\}$ spans $\mathbb{R}^k$. Then, there are real numbers $\alpha_1, \dots, \alpha_n$ such that $y = \alpha_1 z_1 + \dots + \alpha_n z_n$. But then we have $y^T y = \alpha_1 z_1^T y + \dots + \alpha_n z_n^T y = 0$, yielding $y = 0$, a contradiction

Hence if the z_i's span $\mathbb{R}^k$, then Q is positive definite. This condition is equivalent to rank $[z_1, \ldots z_n] = k$.