+ Code + Text

```
import pandas as pd
import math
import numpy as np
import matplotlib.pyplot as plt
```

```
from google.colab import drive
drive.mount("/content/gdrive")
```

```
     Mounted at /content/gdrive
```

```
%cd /content/gdrive/My Drive/NNFL/Assignment1
```

```
     /content/gdrive/My Drive/NNFL/Assignment1
```

```
# function for normalising data
def norm(data):
  # norm_data = data
  mean = np.mean(data)
  std = np.std(data)
  norm_data = (data-mean)/std
  return norm_data
```

```
def likelihood(x, meanmat, covariance):
  n = len(x)
  coeff = 1 /((( 2 * np.pi )** (7/2) )*np.linalg.det(covariance)** 0.5 )
  l = coeff*np.exp(- 0.5 * np.dot(np.dot((x - meanmat),np.linalg.inv(covariance)),(x - mea
  return l
```

```
def MLrule(x_testing, x, y):
  # splitting the input data into it's different classes
  x1 = np.array([x[i] for (i, val) in enumerate(y) if val == 1 ])
  x2 = np.array([x[i] for (i, val) in enumerate(y) if val == 2 ])
  x3 = np.array([x[i] for (i, val) in enumerate(y) if val == 3 ])

  m1 = np.mean(x1, axis = 0)
  m2 = np.mean(x2, axis = 0)
  m3 = np.mean(x3, axis = 0)
  cov1 = np.cov(np.transpose(x1.astype(float)))
  cov2 = np.cov(x2.astype(float).T)
  cov3 = np.cov(x3.astype(float).T)

  # likelihood
  l1 = likelihood(x_testing, m1, cov1)
  l2 = likelihood(x_testing, m2, cov2)
  l3 = likelihood(x_testing, m3, cov3)

  # output
  if max(l1, l2, l3) == l1:
    return 1
  elif max(l1, l2, l3) == l2:
    return 2
    return 2
  else:
    return 3


def confusion_mat(y_predicted, y_testing):
  conf_mat = np.zeros((3,3))
  for i in range(len(y_testing)):
    if y_testing[i] == 1:
      if y_predicted[i] == 1:
        conf_mat [0][0] += 1
      if y_predicted[i] == 2:
        conf_mat [0][1] += 1
      if y_predicted[i] == 3:
        conf_mat [0][2] += 1
    if y_testing[i] == 2:
      if y_predicted[i] == 1:
        conf_mat [1][0] += 1
      if y_predicted[i] == 2:
        conf_mat [1][1] += 1
      if y_predicted[i] == 3:
        conf_mat [1][2] += 1
    if y_testing[i] == 3:
      if y_predicted[i] == 1:
        conf_mat [2][0] += 1
      if y_predicted[i] == 2:
        conf_mat [2][1] += 1
      if y_predicted[i] == 3:
        conf_mat [2][2] += 1
  return conf_mat


# extracting the data
data = pd.read_excel("data_q6_q7.xlsx")
data = np.asarray(data)
data = np.random.permutation(data)
print(data)


# splitting into input and output
X = data[:, :-1]  #input
y = data[:,-1]    #output

# normalizing X and y
X = norm(X)


rows = X.shape[0]
X_training = X[0 : int(rows*0.7)]
X_testing = X[int(rows*0.7) : rows+1]
y_training = y[0 : int(rows*0.7)]
y_testing = y[int(rows*0.7) : rows+1]


ind_acc1 = []
ind_acc2 = []
ind_acc3 = []
```

```
overall_acc = []
y_predicted = []
for i in range(len(X_testing)):
  y_predicted.append(MLrule(X_testing[i], X_training, y_training))
# print(y_predicted, y_testing)
conf_mat = confusion_mat(y_predicted, y_testing)
# individual accuracy
ind_acc1 = conf_mat[ 0 ][ 0 ]/sum(conf_mat[ 0 ])
#  ind_acc1.append(acc1)
ind_acc2 = conf_mat[ 1 ][ 1 ]/sum(conf_mat[ 1 ])
#ind_acc2.append(acc2)
ind_acc3 = conf_mat[ 2 ][ 2 ]/sum(conf_mat[ 2 ])
# ind_acc3.append(acc3)
  # overall accuracy
overall_acc = (conf_mat[ 0 ][ 0 ] + conf_mat[ 1 ][ 1 ] + conf_mat[ 2 ][ 2 ])/np.sum(conf_m
# overall_acc.append(acc)
# avg_ind_acc1 = sum(ind_acc1)/len(ind_acc1)
# avg_ind_acc2 = sum(ind_acc2)/len(ind_acc2)
# avg_ind_acc3 = sum(ind_acc3)/len(ind_acc3)
# avg_overall_acc = sum(overall_acc)/len(overall_acc)
print("Individual accuracy of class 1:", ind_acc1)
print("Individual accuracy of class 2:", ind_acc2)
print("Individual accuracy of class 3:", ind_acc3)
print("Overall accuracy:", overall_acc)
```

```
Individual accuracy of class 1: 0.8333333333333334
Individual accuracy of class 2: 0.9444444444444444
Individual accuracy of class 3: 0.9047619047619048
Overall accuracy: 0.8888888888888888
```