LAB 5

Naive Bayes :

In this method we have considered every pixel or element of an image as independent.By this assumption, the covariance matrix will 0 value at non diagonal elements.

Observation:

This method has similar approach as lda and qda , difference is that here ,every pixel is considered independent.So, covariance matrix is calculated using cov function of numpy, then it is multiplied element wise with identity matrix.Accuracy by this approach obtained is 81.67% which is printed by code.

```
In [ ]: import numpy as np
        import matplotlib.pyplot as plt
        import pandas as pd
        import scipy.linalg as la
        import math
        import random
        from scipy.fftpack import fft,fftfreq
        from scipy.linalg import toeplitz
        from itertools import count
        from sklearn.model_selection import train_test_split
        data1 = pd.read_csv('mnist_train.csv')
        data2 = pd.read_csv('mnist_test.csv')
        Y_train = np.asfarray(data1.iloc[:,0])
        X_train = np.asfarray(data1.iloc[:,1:])
        Y_test = np.asfarray(data2.iloc[:,0])
        X_test = np.asfarray(data2.iloc[:,1:])
        Y_test=Y_test.reshape((len(Y_test),))
        Y_train=Y_train.reshape((len(Y_train),))

        cov_all = []
        mean_all = []
        po=np.zeros((10))
        for k in range(10):
            tmp = X_train[[i for i, j in zip(count(), Y_train) if
        j == k],:]
            cov_all.append((np.cov(tmp.T,bias=True)*np.identity(78
        4)))
            mean_all.append(np.mean(tmp,axis = 0))
            po[k] = np.log(len(tmp)/Y_train.shape[0])
        coin=np.zeros((10,784,784))
        mean_all=np.array(mean_all)
        cov_all=np.asfarray(cov_all)
        for k in range(10):
            coin[k]=np.linalg.pinv(cov_all[k])
        ypred=[]
        for i in range(len(Y_test)):
            max=[]
            for k in range(10):
                temp =X_test[i]-mean_all[k]
                temp1 = (-1/2)*np.dot(temp,np.dot(coin[k],(temp.
        T)))
                max.append(temp1 + po[k])
            ypred.append(np.argmax(max))
        ypred=np.array(ypred)
        ac = (ypred[:]==Y_test[:]).mean()
        print((ac)*100)

        81.67816781678168
```