```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sbn
from scipy.stats import multivariate_normal as mvn

#dataset is values of light intensity for each pixel in the pictures
data=pd.read_csv("MNIST_train.csv") #training data
data_test=pd.read_csv("MNIST_test.csv") #testing data

X=data.to_numpy() #converts dataframe to a numpy array
X_test = data_test.to_numpy()


y=X[:,2] #grabs the third column for the labels
X=X[:,3:]  #grabs the rest of the data frame after the labels, which is light intesity in each pixel

#does the same for the test data
y_test = X_test[:,2]
X_test = X_test[:,3:]

X=X/255     #the highest value in the data is 255, so this step normalizes the arrays to values
between 0 and 1
X_test = X_test/255



class GaussBayes():
   def fit(self, X, y, epsilon = 1e-2):    #an error value of .01 gives the best results. smaller values
prove to be too small for the divsion step
       self.likelihoods = dict()
       self.priors = dict()
       self.K = set(y.astype(int))

       for k in self.K:
          X_k = X[y == k,:]
          N_k, D = X_k.shape
          mu_k = X_k.mean(axis=0)
          self.likelihoods[k] = {"mean":X_k.mean(axis=0), "cov":
(1/(N_k-1))*np.matmul((X_k-mu_k).T,X_k-mu_k)+epsilon*np.identity(D)}
          #covariance is a cov matrix
          self.priors[k] = len(X_k)/len(X)

   def predict(self, X):
       N, D = X.shape
```

```
        P_hat = np.zeros((N,len(self.K)))
        for k, l in self.likelihoods.items():
            P_hat[:,k] = mvn.logpdf(X, l["mean"], l["cov"])+np.log(self.priors[k])
        return P_hat.argmax(axis = 1)


gb=GaussBayes()
gb.fit(X,y) #uses the training dataset

y_hat=gb.predict(X_test) #uses the newly trained model to fit the test dataset

def accuracy(y,y_hat):     #tests the accuracy of the model. This gives an accuracy of 94.7%
    return np.mean(y == y_hat)

#visualize means
#confusion matrix (matrix of numbers that get confused for each other)
def show_me(X):
    plt.imshow(X.reshape(28,28))    #imshow makes it an image and 28 is the sqrt of 784
    #call as X[1,:] etc

#sets up a confusion matrix and applies it to a heatmap
confmatdata={"y_actual":y_test, "y_predicted":y_hat}
df = pd.DataFrame(confmatdata, columns=["y_actual","y_predicted"])  #convert back to pandas
dataframe form
confusion_matrix = pd.crosstab(df["y_actual"],df["y_predicted"], rownames=["Actual"],
colnames=["Predicted"])
ax = sbn.heatmap(confusion_matrix, linewidth = 0.5, annot = True, fmt = "d")
```