```
# import the packages
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import multivariate_normal as mvn
from sklearn.model_selection import train_test_split


data_train = pd.read_csv('/content/MNIST_train.csv')
data_train.head()
```

	Unnamed: 0	index	labels	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	1
0	0	0	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	2	2	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	3	3	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	4	4	9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

5 rows × 787 columns

```
data_train.shape
```

```
(60000, 787)
```

```
data1_train = data_train.drop(data_train.columns[[0, 1]], axis=1)
data1_train.head()
```

	labels	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

5 rows × 785 columns

```
data1_train.isnull().sum()
```

```
labels    0
0         0
1         0
2         0
3         0
          ..
779       0
780       0
781       0
782       0
783       0
Length: 785, dtype: int64
```

```
data1_train.columns
```

```
Index(['labels', '0', '1', '2', '3', '4', '5', '6', '7', '8',
       ...
       '774', '775', '776', '777', '778', '779', '780', '781', '782', '783'],
      dtype='object', length=785)
```

```
data1_train['labels'].value_counts()
```

```
1    6742
7    6265
3    6131
2    5958
9    5949
0    5923
6    5918
8    5851
4    5842
5    5421
Name: labels, dtype: int64
```

```
import seaborn as sns
count=sns.countplot(data1_train["labels"])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: P
  FutureWarning
```

from the above histogram it is obvious than the number image 1 has the highest instances and number image 5 has the lowest, but the difference between them is not overwhelming and not skewed toward a class and can therefore be used in training a model

```
X = data1_train.to_numpy()
X
```

```
array([[5, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [4, 0, 0, ..., 0, 0, 0],
       ...,
       [5, 0, 0, ..., 0, 0, 0],
       [6, 0, 0, ..., 0, 0, 0],
       [8, 0, 0, ..., 0, 0, 0]])
```

```
y=X[:,0]
y
```

```
array([5, 0, 4, ..., 5, 6, 8])
```

```
X1=X[:,1:]
X1
```

```
array([[0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       ...,
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0]])
```

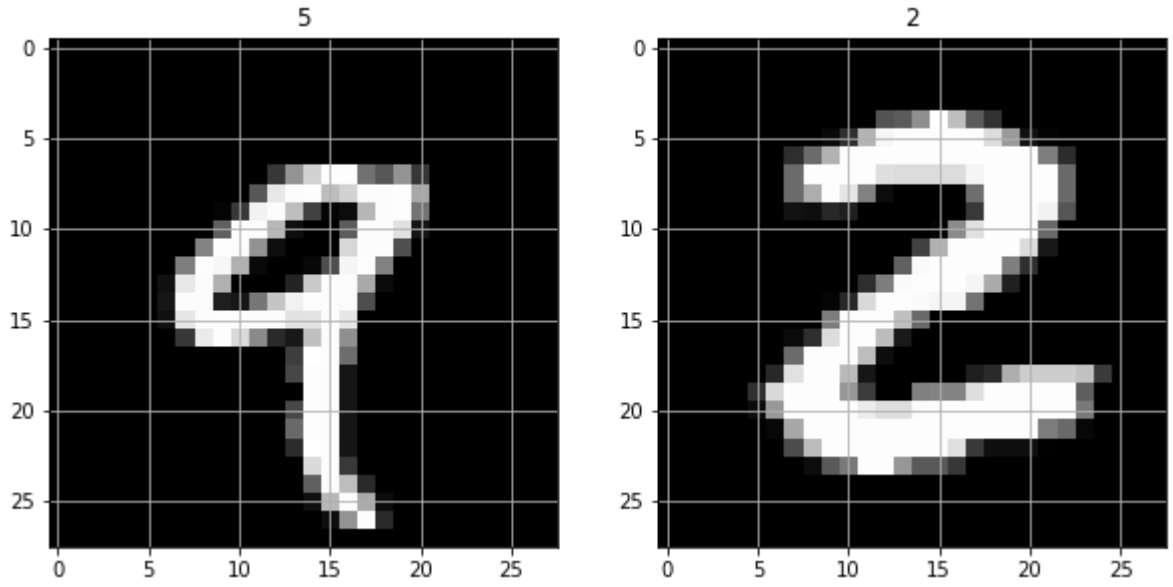
```
# split into train validation sets
X_train, X_val, y_train, y_val = train_test_split(X1, y, test_size=0.33)
print(X_train.shape, X_val.shape, y_train.shape, y_val.shape)
```

```
(40200, 784) (19800, 784) (40200,) (19800,)
```

```
plt.figure(figsize=(10,8))
img = X1[4]
img = img.reshape((28,28))
plt.subplot(1,2,1)
plt.imshow(img,cmap='gray')
plt.title(data1_train.iloc[244,0])
```

```
plt.grid()


img1 = X1[25]
img1 = img1.reshape((28,28))
plt.subplot(1,2,2)
plt.imshow(img1,cmap='gray')
plt.title(data1_train.iloc[25,0])
plt.grid()
plt.show()
```




```
class GaussNB():
    def fit(self, X, y, epsilon = 1e-3):
        self.likelihoods = dict()
        self.priors = dict()

        self.K = set(y.astype(int))

        for k in self.K:
            X_k =X[y==k,:]
            self.likelihoods[k] = {"mean":X_k.mean(axis=0), "cov":X_k.var(axis=0) + epsilon}
            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)


gnb = GaussNB()
```

```
gnb.fit(X_train,y_train)
```

```
y_hat=gnb.predict(X_val)
```

```
def accuracy(y,y_hat):
    return np.mean(y==y_hat)
```

```
accuracy(y_val,y_hat)
```

```
0.587020202020202
```

```
### Multiclasses classifier, works with any number of classes(categories)
class GaussBayes():

    def fit(self, X,y, epsilon=1e-3):
        self.likelihoods = dict()
        self.priors = dict()
        ###for categorical values use onehtencoder
        self.K = set(y.astype(int))

        for k in self.K:
            X_k = X[y==k,:]
            N_k,D = X_k.shape   ## N_k=total number of observations of that class
            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
            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)
```

```
gbayes = GaussBayes()
```

```
gbayes.fit(X_train,y_train)
```

```
y_hat=gbayes.predict(X_val)
```

```
accuracy(y_val,y_hat)
```

```
0.7564141414141414
```

```
data_test = pd.read_csv('/content/MNIST_test.csv')
data_test.head()
```

	Unnamed: 0	index	labels	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
0	0	0	7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	2	2	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	3	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	4	4	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

5 rows × 787 columns

```
data1_test = data_test.drop(data_test.columns[[0, 1]], axis=1)
data1_test.head()
```

	labels	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
0	7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

5 rows × 785 columns

```
data1_test.shape
```

```
(10000, 785)
```

```
data1_test.isnull().sum()
```

```
labels    0
0         0
1         0
2         0
3         0
         ..
779       0
```

```
780       0
781       0
782       0
783       0
Length: 785, dtype: int64
```

```
X_test = data1_test.to_numpy()
X_test
```

```
array([[7, 0, 0, ..., 0, 0, 0],
       [2, 0, 0, ..., 0, 0, 0],
       [1, 0, 0, ..., 0, 0, 0],
       ...,
       [4, 0, 0, ..., 0, 0, 0],
       [5, 0, 0, ..., 0, 0, 0],
       [6, 0, 0, ..., 0, 0, 0]])
```

```
y_test=X_test[:,0]
y_test
```

```
array([7, 2, 1, ..., 4, 5, 6])
```

```
X1_test=X_test[:,1:]
X1_test
```

```
array([[0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       ...,
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0]])
```

```
y_hat=gbayes.predict(X1_test)
```

```
accuracy(y_test,y_hat)
```

```
0.7542
```

```
plt.figure(figsize=(10,7))
y_actu = pd.Series(y_test, name='Actual')
y_pred = pd.Series(y_hat, name='Predicted')
cm = pd.crosstab(y_actu, y_pred)
ax = sns.heatmap(cm, annot=True, cmap = 'RdYlGn', fmt="d")
plt.ylabel('True label')
plt.xlabel('Predicted label')
```

```
Text(0.5, 42.0, 'Predicted label')
```