Checklist:

F1:	1. Successfully load the dataset.	√
	2. Display the dataset information.	√
F2:	1. Successfully call library functions to train a model.	√
F3:	1. You have an implementation of an algorithm that is able to train a model.	√
F4:	1. Output the train and test errors for f2's model.	√
	2. Output the train and test errors for f3's model.	√
F5:	1. Allow users to query two models by changing the input.	√

0. Introduction

Dataset : Optical recognition of handwritten digits dataset

Algorithm : Naive bayes

Since Naive bayes is a supervised, non-modeled classification algorithm, the implement f2 and f3 can be achieved by running the algorithm directly without save models.

1. Detailing how to run your program, including the software dependencies

Software Dependencies:

- Python 3.8
- NumPy 1.19.2
- Scikit-learn 0.23.2
- Matplotlib 3.3.2

How to run program:

Clone to your development environment and run *main.py* with python 3.8 interpreter: python main.py

User interface of python program:

```
1. F1 The details of the dataset
2. F2 Implement Scikit-learn navie GaussianNB algorithm
3. F3 Implement my navie bayes algorithm
4. F4 Compare the train error and test error of the two algorithm
5. F5 Query two datasets
6. End
Enter number from 1 to 6:
```

The user selects numbers between 1 and 6 to run different implement, other numbers are not accepted.

2. Explaining how the functionalities and additional requirements are implemented

The dataset is loaded from the scikit-learn library and put into the matrix, and use `train_test_split` function to divide the dataset in to train dataset which is 30% and test dataset which is 70%

```
from sklearn import datasets
tra_x,tes_x,tra_y,tes_y = train_test_split(all_x,all_y,test_size=0.7)
```

def my_naive_bayes(x, y, mean, variance):

Transfer samples and labels of the dataset, mean and variance of the training set

Bayes formula $P(Y|\vec{X}) = \frac{P(\vec{X}|Y)*P(Y)}{P(\vec{X})}$ is used to calculate the probability of sample vector of each digits

```
for i in range(x.shape[0]):
    lists = []
    for j in range(len(classes)):
        numerator = np.exp(-((x[i]-mean[j]) ** 2) / ( *
variance[j]))
        denominator = np.sqrt(2 * np.pi * (variance[j]))
        prob_xc = numerator / denominator
        ratio = np.sum(np.log(prob_xc))
        lists.append(ratio)
    pred = lists.index(max(lists))
    if pred == y[i]:
        my_t_count = my_t_count + 1
        confusion_matrix[int(y[i])][int(y[i])] = _
confusion_matrix[int(y[i])][int(y[i])] + 1
    else:
        for k in range(len(classes)):
            if pred == k:
                confusion_matrix[int(y[k])][int(y[i])] =
confusion_matrix[int(y[k])][int(y[i])] + 1
```

def f1(): Provide the details of the dataset

Call the variable from the header: **all_x, all_y, a_x, tes_x, tra_y, tes_y**

Print the number of datasets: **all_x.shape[0], tra_x.shape[0], tes_x.shape[0]**

Use a loop statement to calculate the number of each digits

Print the maximum and minimum values for each feature:

```
print(np.max(all_x, axis=0))
print(np.min(all_x, axis=0), '\n')
```

def f2():

Call scikit-learn library GaussianNB function/algorithm to process the datasets

```
from sklearn.naive_bayes import GaussianNB
sk_nb = GaussianNB()
sk_nb.fit(tem_tra_x, tem_tra_y)
sk_nb_tra_y = (sk_nb.predict(tem_tra_x))
sk_nb_tes_y = (sk_nb.predict(tem_tes_x))
```

def f3():

Call the method **my_naive_bayes(x, y, mean, variance)** to compute the datasets
Assign two matrixes to store the mean and variance of the train dataset

```
for i in classes:
    tra_x_c = tra_x[tra_y == i]
    mean[int(i), :] = tra_x_c.mean(axis=0)
    variance[int(i), :] = tra_x_c.var(axis=0)
```

Use loop statements to filter samples for each category Assign two matrixes to store the mean and variance of the train dataset

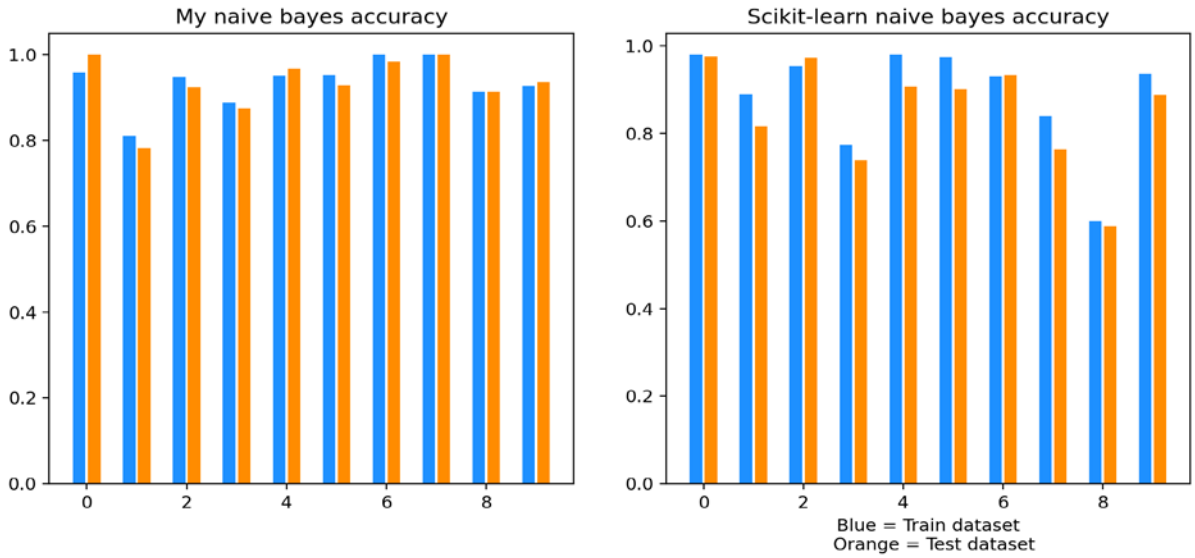
Call the method **`my_naive_bayes(x, y, mean, variance)`** to return the accuracy rate and number of train and test dataset respectively

Return the correct rate for each digit, and the correct quantity counter

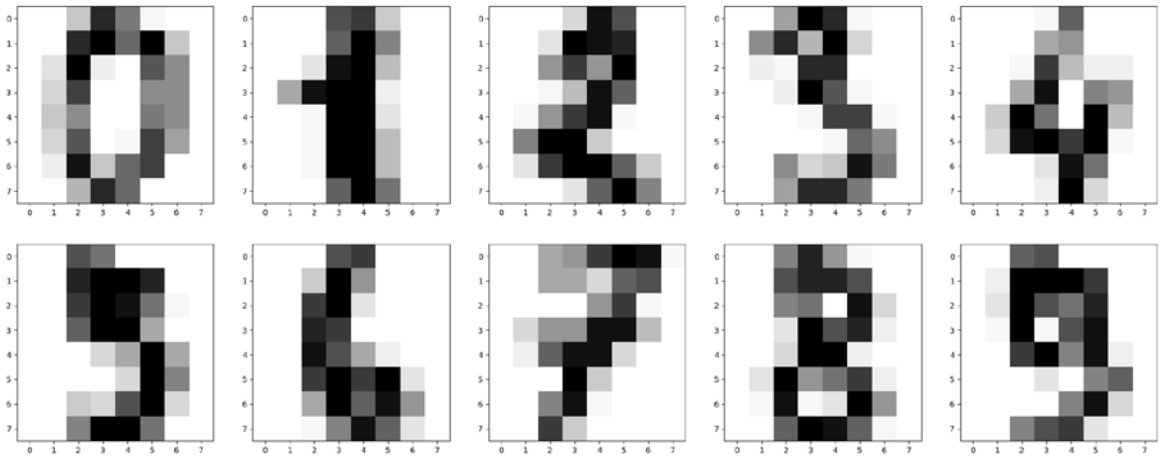
`def f4():`

Load and print the return values of both algorithms(My naive bayes algorithms and s scikit-learn GaussianNB algorithms) by call f2 and f3 function. And use matplotlib library to create a visualization

```
Sklearn training accuracy: 86.83 %  and the error number is  71
Sklearn testing accuracy: 82.51 %  and the error number is  220
My training accuracy: 93.14 %  and the error number is  37
My testing accuracy: 91.73 %  and the error number is  104
```



`def f5():` Detect the input number and returns the aim image and the detail of data



Additional requirements:

Design interactive interface, users can directly run F1 – F6 programs by input instructions. Bar chart is used in the F5 program to visually compare the train dataset and test dataset accuracy under different algorithms.

3. Providing the details of your implementation

1. The idea of my algorithm

The naive bayes algorithm is used to classify the Optical recognition of handwritten digits dataset. The basic method is to calculate the probability that the current feature samples belong to a certain class based on statistical data and according to the conditional probability formula, and select the category with the highest probability

Each picture of the handwritten digits' dataset consists of 8*8 pixels, each pixel is represented by 0 - 16 gray level and has a label to indicate the class. Data is the array 1 * 64, which can be regarded as vector $\vec{X}$. Import the bayes formula

$$P(Y|\vec{X}) = \frac{P(\vec{X}|Y) * P(Y)}{P(\vec{X})}$$

Since $P(\vec{X})$ is constant, so formula into looking for the biggest $P(\vec{X}|Y) * P(Y)$

$$\left(\prod_i P(x^i|y)\right) * P(y)$$

2. The meaning of parameter and variable

`all_x:` all the data from datasets

`all_y:` all the target of each data from dataset

`tra_x,tes_x,tra_y,tes_y:` Divide the datasets into train and test dataset

`sk_tra_digit_accuracy:` The digital accuracy of train dataset returned by scikit-learn GaussianNB algorithm

`sk_tra_true_count:` The correct number of train dataset returned by scikit-learn GaussianNB algorithm

`my_tes_digit_accuracy:` The digital accuracy of test dataset returned by my naive bayes algorithm

`my_tes_true_count:` The correct number of test dataset returned by my naive bayes algorithm

`mean:` The mean of train dataset features classes

`variance:` The variance of train dataset features classes