
西北大学

# 智能信息系统综合实践

# 实验报告

题　　目：　贝叶斯分类+PCA

年　　级：　2020

专　　业：　软件工程

姓　　名：　张琬琪

# 一、题目

## 作业一鸢尾花分类

**作业1：** 数据集内包含 3 类共 150 条记录，每类各 50 个数据，每条记录都有 4 项特征：花萼长度、花萼宽度、花瓣长度、花瓣宽度，可以通过这4个特征预测鸢尾花卉属于（iris-setosa，iris-versicolour，iris-virginica）中的哪一品种。

***要求：**

① 在鸢尾花数据集上，尽量手动编写朴素贝叶斯分类代码，完成分类实验。（可参考李航统计学习方法）

② 在①的基础上，对鸢尾花中特征部分进行划分，分别利用1个，2个，3个，4个特征进行分类，比较不同特征对分类结果的影响，画图比较。

**作业2：** 选用的MNIST数据集，每条记录都有 28*28 项特征，随机选择70%训练，30%测试。

***要求：**

① 在数据集上，利用朴素贝叶斯分类代码，完成分类实验，并画图给出不同数字的准确率，并尝试分析其原因。

② 利用PCA将28*28维度降维到某个维度（如10，20等），完成分类并计算其准确率。

③ 贝叶斯中唯一的参数—平滑系统，可尝试调整参数，比较其对分类的影响。

## 二、 解题步骤

### 作业 1：

### ①：【解题思路】：

1）导入所需数据集，参考以往实验方法随机划分测试集训练集

**2）利用 sklearn 中提供的朴素贝叶斯分类算法，实验中选用 GaussianNB**

（注：在 sklearn 中，一共有 3 个朴素贝叶斯的分类算法类：

GaussianNB（先验是高斯分布的朴素贝叶斯）；

BernoulliNB（先验为伯努利分布的朴素贝叶斯）；

MultinomialNB（先验是多项式分布的朴素贝叶斯）。

**高斯 NB 用于连续值**；多项式 NB 用于离散的多值；伯努利 NB 用于离散的二值。）

3）利用训练集对于模型进行训练

4）对测试集进行预测，并输出分类相关指标：precision 准确率，recall 召回率，f1-score，混淆矩阵，决定系数$R^2$对模型分类情况进行评估

### ②：【解题思路】：

1)因题目要求需要使用不同数目的分类指标进行分类，因此在①的基础上，**在数据集划分时进行函数封装，将想要选择的特征数量，设置为变量**，最后在**主函数**中，**依次输入不同的特征数量**，达到最终目的。

（注:这里由于本身数据集四列即为四种不同特征，因此在具体

实验时，自己选取的 1 至 4 个特征，皆为顺序选取，即可简单理解为：第一次取所有列即选了所有四个特征，第二次取前三列即选了前三个特征，以此类推。）

2）进行绘制图像以类比不同特征对其分类精度的影响，本次实验经过衡量决定使用 ROC 曲线进行比较，选取 **scikitplot 库中 skplt.metrics.plot_roc(y_test, predicted_probas) 进行 ROC 曲线的绘制。**

1.ROC曲线能很容易地查出任意界限值时的对性能的识别能力。

2.选择最佳的诊断界限值。ROC曲线越靠近左上角,试验的准确性就越高。最靠近左上角的ROC曲线的点是错误最少的最好阈值，其假阳性和假阴性的总数最少。

3.两种或两种以上不同诊断试验对算法性能的比较。在对同一种算法的两种或两种以上诊断方法进行比较时，可将各试验的ROC曲线绘制到同一坐标中，以直观地鉴别优劣，靠近左上角的ROC曲线所代表的受试者工作最准确。亦可通过分别计算各个试验的ROC曲线下的面积(AUC)进行比较，哪一种试验的 AUC最大，则哪一种试验的诊断价值最佳。

**①、②【解题思路】代码：**

```
iris = load_iris()

def nb_select_feature(n):
    x = iris.data[:, : n]
    y = iris.target
    x_train,x_test,y_train,y_test=model_selection.train_test_split(x,y,test_size=0.3, random_state=42)
    model = GaussianNB()
    model.fit(x_train, y_train)
    predict = model.predict(x_test)
    predicted_probas = model.predict_proba(x_test)
    print(classification_report(y_test, predict))
    print(metrics.confusion_matrix(y_test, predict))#混淆矩阵

    print('特征数量为{}的朴素贝叶斯模型的预测准确率：{:.2f}%'.format(n,model.score(x_test, y_test) * 100))
    skplt.metrics.plot_roc(y_test, predicted_probas)#绘制ROC曲线
    plt.rcParams['font.sans-serif'] = ['SimHei']
    plt.rcParams['axes.unicode_minus'] = False
    plt.title('特征数量为{}的ROC curve'.format(n))
    plt.show()


if __name__ == '__main__':
    n=[1,2,3,4]
    while len(n)>0:
        a=n.pop()
        nb_select_feature(a)
```

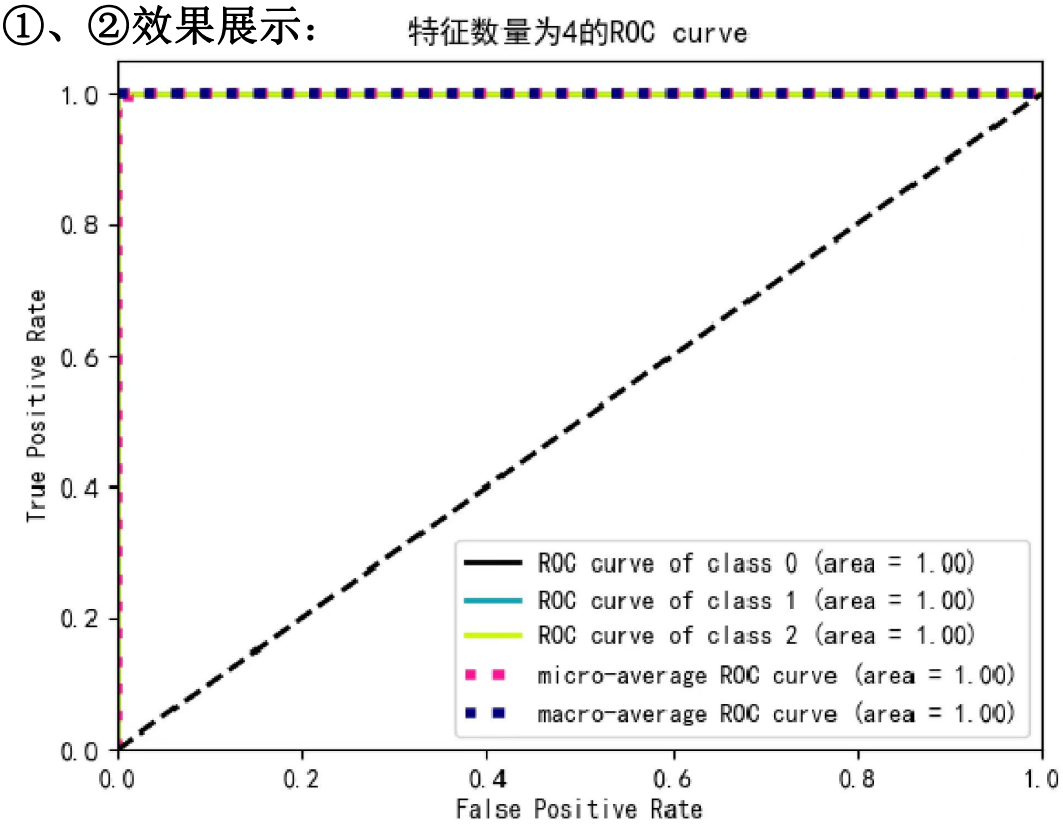
图 1

## 代码所需库与包见图 2：

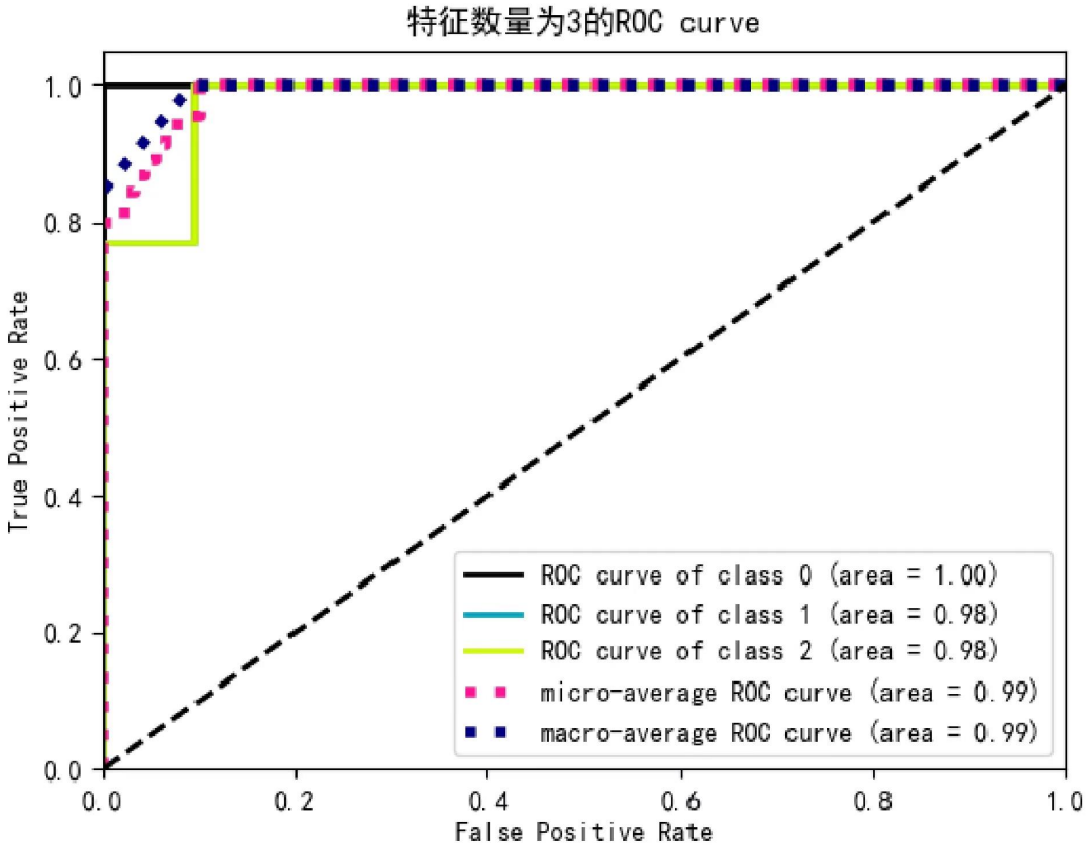
```
from sklearn.datasets import load_iris
from sklearn import model_selection
from sklearn.naive_bayes import GaussianNB #高斯分布的朴素贝叶斯
from sklearn import metrics
from sklearn.metrics import classification_report

import matplotlib.pyplot as plt
import scikitplot as skplt
```

图 2

## ①、②效果展示：



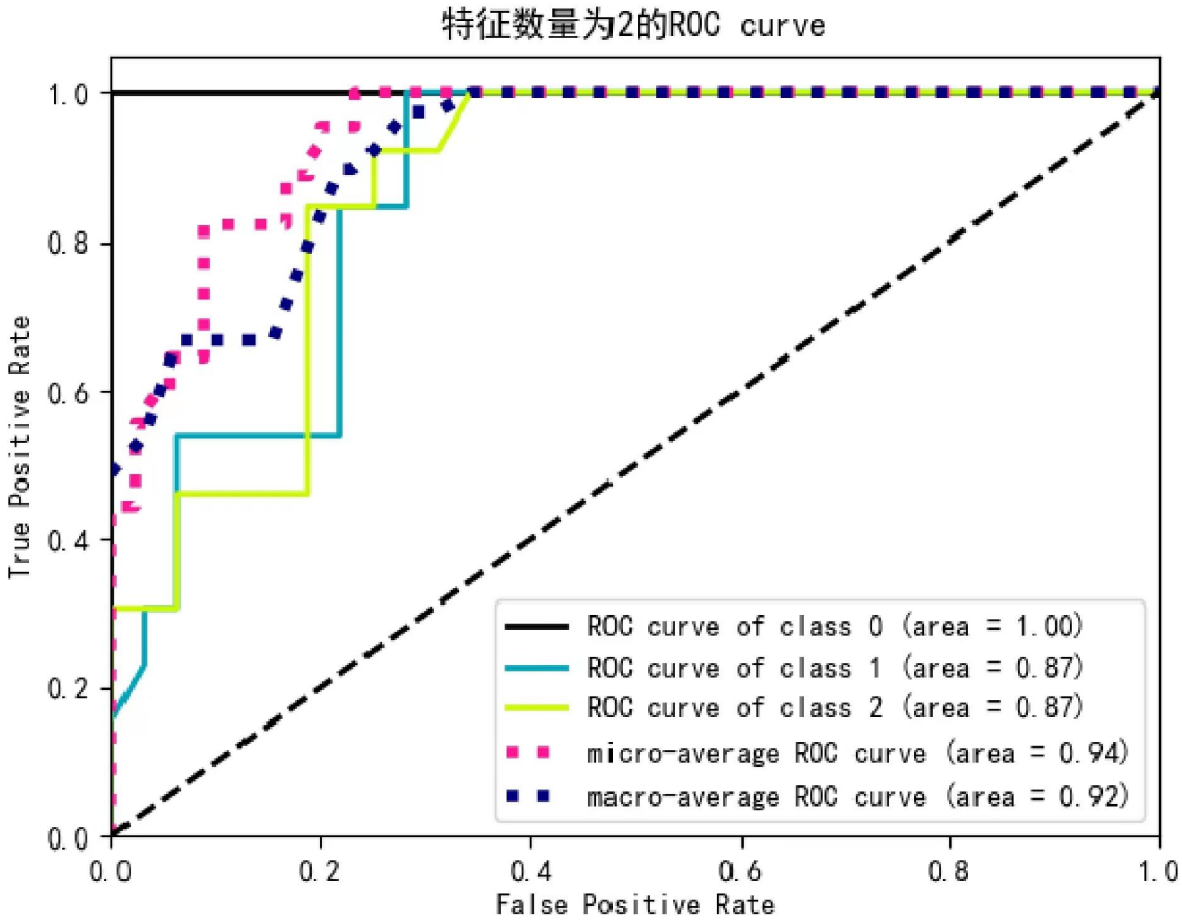

```
D:\developlAPP\python3.10.4\python.exe D:\科目学习类\pythonhomeworkCODE\exercise5\01.py
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        19
           1       1.00      0.92      0.96        13
           2       0.93      1.00      0.96        13

    accuracy                           0.98        45
   macro avg       0.98      0.97      0.97        45
weighted avg       0.98      0.98      0.98        45

[[19  0  0]
 [ 0 12  1]
 [ 0  0 13]]
特征数量为4的朴素贝叶斯模型的预测准确率：97.78%
```




| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 1.00 | 1.00 | 1.00 | 19 |
| 1 | 0.83 | 0.77 | 0.80 | 13 |
| 2 | 0.79 | 0.85 | 0.81 | 13 |
| accuracy | | | 0.89 | 45 |
| macro avg | 0.87 | 0.87 | 0.87 | 45 |
| weighted avg | 0.89 | 0.89 | 0.89 | 45 |

```
[[19  0  0]
 [ 0 10  3]
 [ 0  2 11]]
```

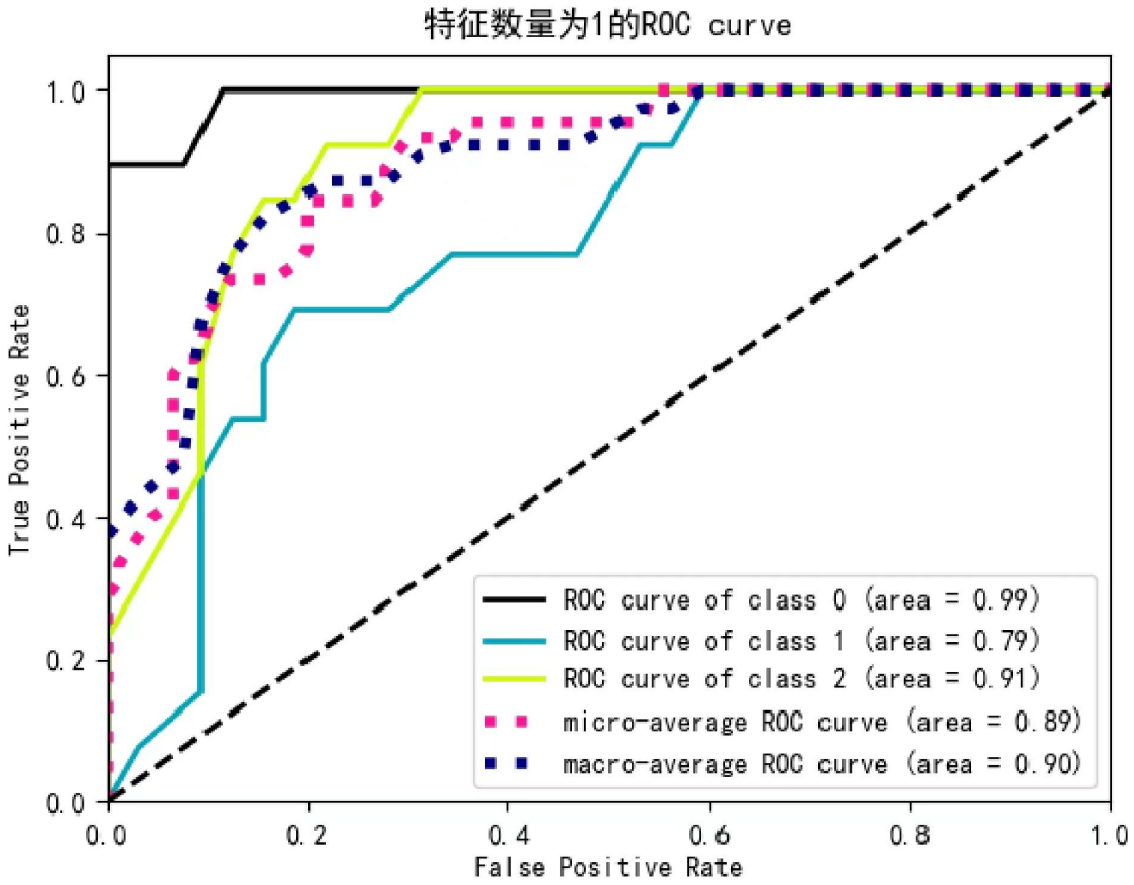
特征数量为3的朴素贝叶斯模型的预测准确率：88.89%




| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 1.00 | 1.00 | 1.00 | 19 |
| 1 | 0.78 | 0.54 | 0.64 | 13 |
| 2 | 0.65 | 0.85 | 0.73 | 13 |
| accuracy | | | 0.82 | 45 |
| macro avg | 0.81 | 0.79 | 0.79 | 45 |
| weighted avg | 0.83 | 0.82 | 0.82 | 45 |

```
[[19  0  0]
 [ 0  7  6]
 [ 0  2 11]]
```

特征数量为2的朴素贝叶斯模型的预测准确率：82.22%




| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 1.00 | 0.68 | 0.81 | 19 |
| 1 | 0.50 | 0.62 | 0.55 | 13 |
| 2 | 0.69 | 0.85 | 0.76 | 13 |
| accuracy | | | 0.71 | 45 |
| macro avg | 0.73 | 0.72 | 0.71 | 45 |
| weighted avg | 0.77 | 0.71 | 0.72 | 45 |

```
[[13  6  0]
 [ 0  8  5]
 [ 0  2 11]]
```

特征数量为1的朴素贝叶斯模型的预测准确率：71.11%

```
Process finished with exit code 0
```

作业 2：

①：【解题思路】：

1）数据处理：MNIST 数据集是机器学习领域中非常经典的一个数据集，由 60000 个训练样本和 10000 个测试样本组成，每个样本都是一张 28 * 28 像素的灰度手写数字图片。为了方便实验，将数据集进行整理合并成 70000 个数据，将代表数字放在文件第一列，后面从第二列到 785 列代表 784 个不同的像素值。

2）进行文件读取，将标签和像素值按照列的位置分布进行划分，如图 3 所示：

```
raw_data = pd.read_csv('mnist_data_70000.csv', header=0)
data = raw_data.values

imgs = data[0::, 1::]
labels = data[::, 0]
```

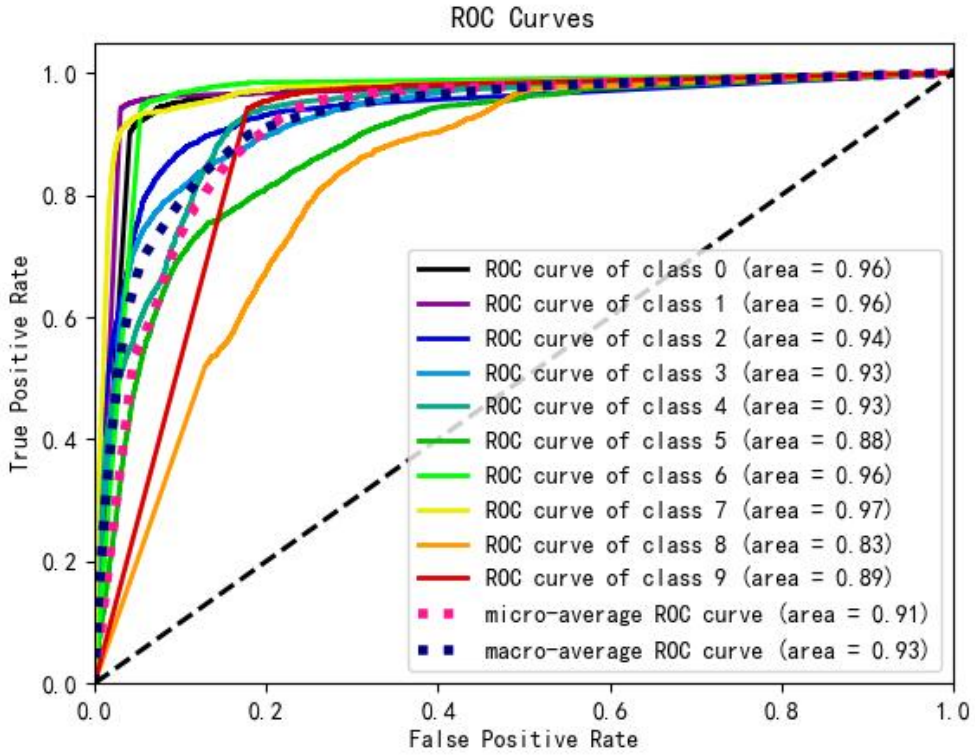
图 3

3）按照题目要求 3：7 进行数据集划分

4）利用训练集数据训练模型，并利用测试集进行测试，生成与作业 1 相同的相关数据与图像。

原因分析：MNIST 数据集数据本身会有些许错误样本，加之训练模型参数不同，或是不同的训练模型以及其他一些不可控制因素，都会造成数字识别准确率的下降或改变。

## 【结果展示】：




02

```
D:\developlAPP\python3.10.4\python.exe D:\科目学习类\pythonhomeworkCODE\exercise5\02.py
              precision    recall  f1-score   support

           0       0.67      0.92      0.77      2064
           1       0.79      0.94      0.86      2422
           2       0.87      0.29      0.43      2078
           3       0.76      0.31      0.44      2163
           4       0.83      0.14      0.24      2001
           5       0.53      0.04      0.08      1879
           6       0.63      0.95      0.76      2092
           7       0.92      0.29      0.44      2221
           8       0.28      0.55      0.37      2013
           9       0.36      0.95      0.52      2067

    accuracy                           0.55     21000
   macro avg       0.66      0.54      0.49     21000
weighted avg       0.67      0.55      0.50     21000

[[1896    2   13    6    4   10   55    1   39   38]
 [   4 2283    3    8    2    3   32    2   59   26]
 [ 259   69  596  121   11   11  529    2  436   44]
 [ 233  105   18  665    3   12  139   10  707  271]
 [  74    7   22   11  280   15  232    9  364  987]
 [ 277   61   11   30    6   80  109    3 1070  232]
 [  24   30    8    0    3    6 1985    0   33    3]
 [   9   17    3   17   13    3    3  648   64 1444]
 [  49  283   10   18    7    8   42    4 1105  487]
 [  14   22    4    2   10    3    1   26   27 1958]]
mnist数据集的朴素贝叶斯模型的预测准确率：54.74%
```

②：【解题思路】：

在①的基础上主要加入 PCA 降维步骤，基于 **sklearn.decomposition.PCA** 进行 PCA 降维。PCA 类基本不需要调参，一般来说，我们只需要指定我们需要降维到的维度。

主要参数：n_components：这个参数可以帮我们指定希望 PCA 降维后的特征维度数目。

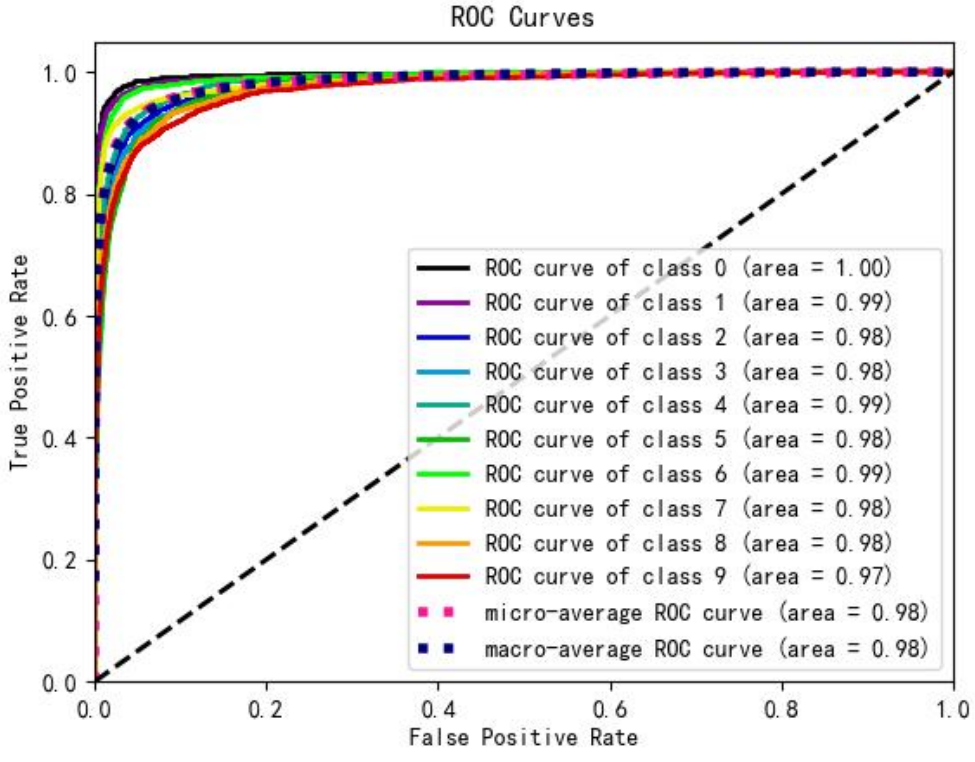
实现：如 **newX=pca.fit_transform(X)，newX** 就是降维后的数据。

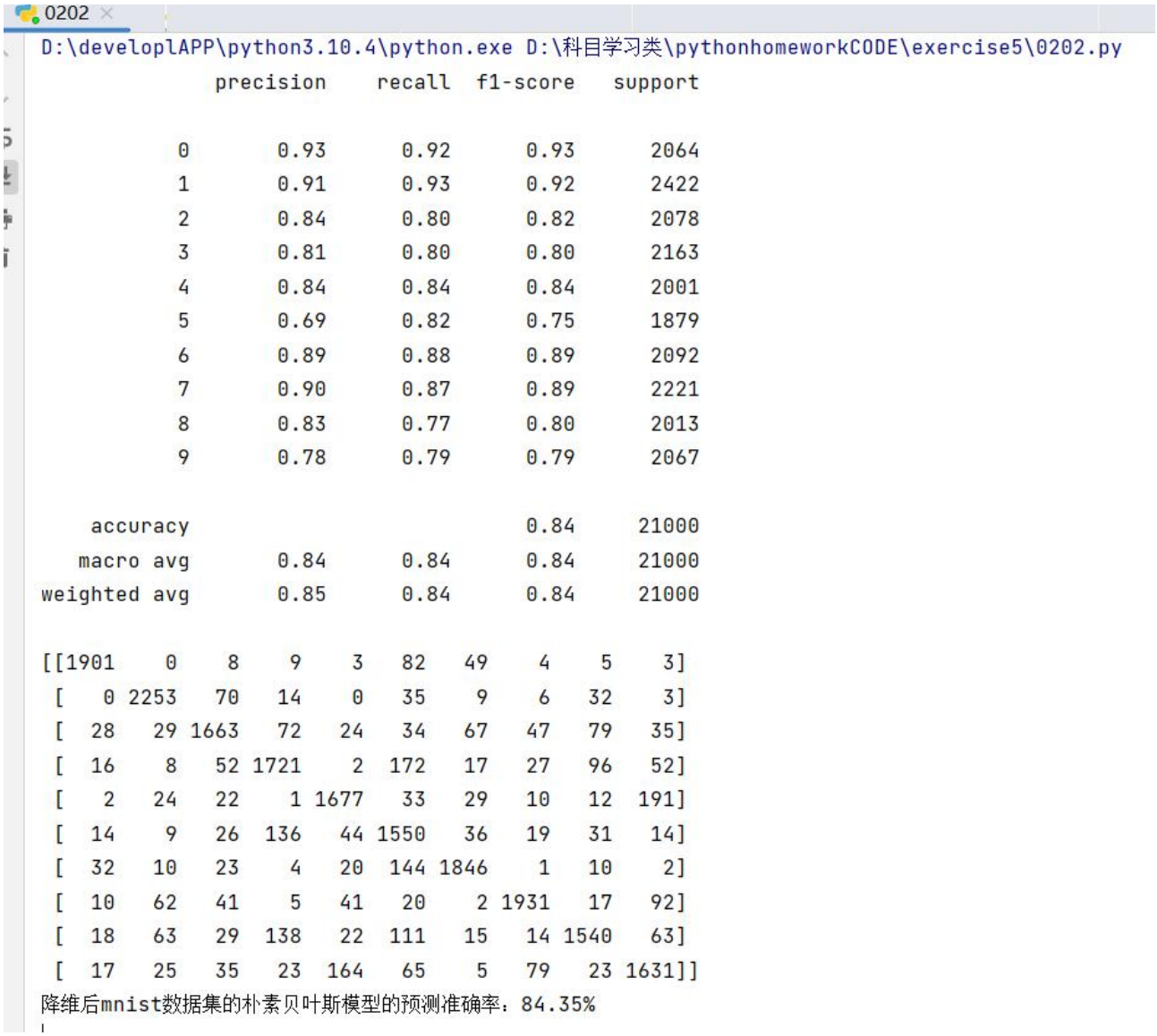
关键代码如图 **4** 所示：

```
raw_data = pd.read_csv('mnist_data_70000.csv', header=0)
data = raw_data.values
pca = decomposition.PCA(n_components=20)

imgs = data[0::, 1::]
labels = data[::, 0]

new_images =pca.fit_transform(imgs)
x_train, x_test, y_train, y_test = train_test_split(new_images, labels, test_size=0.3,random_state=42)
model = GaussianNB()
model.fit(x_train, y_train)
```

图 4

【结果展示】：






**降维后，实验后发现分类准确率明显提升。**

## ③：【解题思路】：

本次实验选取 GaussianNB()，进行简单的参数介绍

参数:

1-priors : array-like of shape (n_classes,),Prior probabilities of the classes. If specified the priors are not adjusted according to the data.

先验：数组-（n_classes,）每一类的先验概率。如果指定先验概

率，则不会根据数据调整先验概率。

**2-var_smoothing : float, default=1e-9**

**方差平滑：浮点数，默认=1e-9，为了计算稳定性而添加到方差中的所有特征的最大方差部分。**

**结合题意，对加粗的参数进行修改**

**默认值：GaussianNB(*, priors=None, var_smoothing=1e-09)**

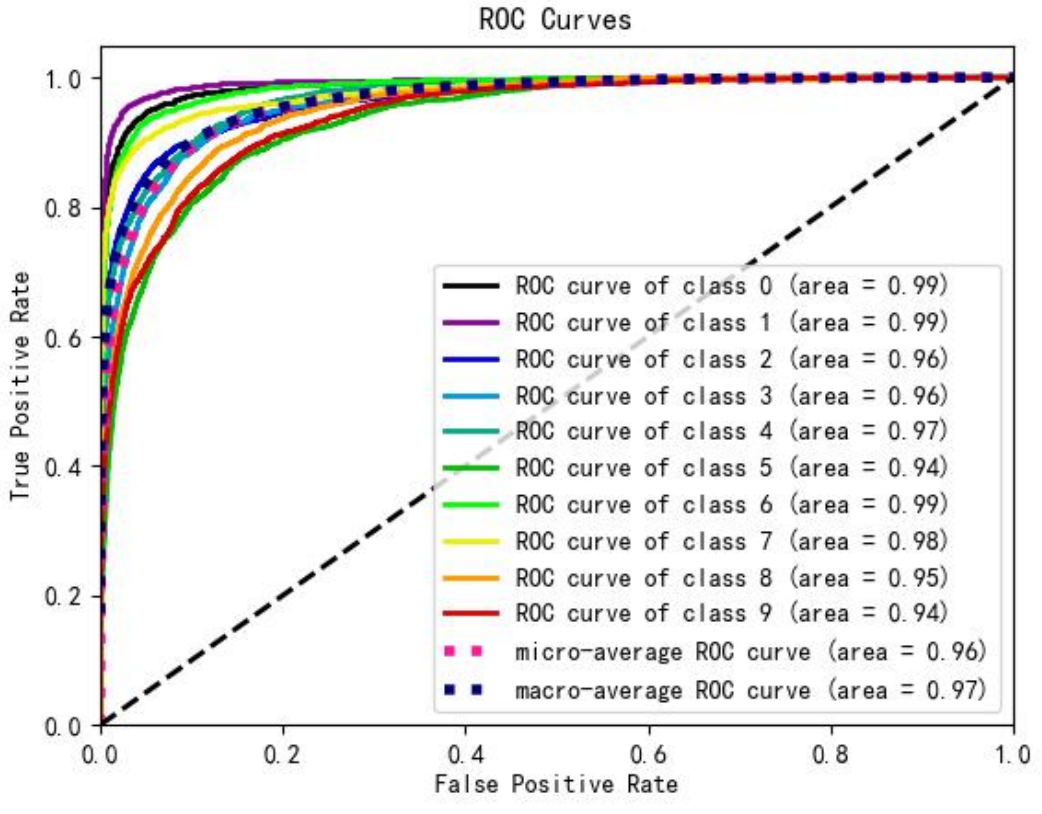
**(在第二问已经降维的基础上)修改数据，见图 5：**

```
new_images =pca.fit_transform(imgs)
x_train, x_test, y_train, y_test = train_test_split(new_images, labels, test_size=0.3,random_state=42)
model = GaussianNB(  priors=None, var_smoothing=1)
model.fit(x_train, y_train)
```

图 5

**实验发现在进行随机修改数据值时，基本上都没有默认值所能达到的准确率高，当然也存在实验数据与次数不够完备的情况。**

## 【结果展示】：



0203

```
D:\developlAPP\python3.10.4\python.exe D:\科目学习类\pythonhomeworkCODE\exercise5\0203.py
              precision    recall  f1-score   support

           0       0.95      0.83      0.89      2064
           1       0.53      0.99      0.69      2422
           2       0.91      0.64      0.75      2078
           3       0.74      0.72      0.73      2163
           4       0.81      0.77      0.79      2001
           5       0.81      0.43      0.56      1879
           6       0.86      0.83      0.84      2092
           7       0.86      0.83      0.84      2221
           8       0.75      0.66      0.70      2013
           9       0.69      0.75      0.72      2067

    accuracy                           0.75     21000
   macro avg       0.79      0.75      0.75     21000
weighted avg       0.79      0.75      0.75     21000

[[1723   10   18   27    8   96   89   27   48   18]
 [   0 2391    4    3    1    1    5    7    8    2]
 [  18  355 1328   55   53    5   69   59  103   33]
 [   7  240   41 1565    5   39   19   23  150   74]
 [   0  146    2    0 1541    0   26   15   12  259]
 [  15  407    1  314   49  804   49   42   82  116]
 [  16  213   34    3   51   29 1730    3   10    3]
 [   6  219    8    0   30    1    1 1852   15   89]
 [   9  374    7  127   23   14   15   17 1336   91]
 [  13  172   15   24  139    4    3  118   24 1555]]
降维后mnist数据集的朴素贝叶斯模型的预测准确率：75.36%

Process finished with exit code 0
```

## 三、总结

①在完成第一题手写朴素贝叶斯算法，由于部分数据在计算概率以及其他值方面出现错误率较高，故最后选取直接利用 sklearn 中的朴素贝叶斯分类算法。

②绘制 ROC 曲线时，实际需要的值与以往实验预测值有细微区别，也就是需要分清以下二者的区别：

**print(clf.predict(x_test)) #** 返回预测标签

**print(clf.predict_proba(x_test)) #** 返回预测属于某标签的概率

③本次实验由于时间仓促，没有做到使用更多的数值或是更加有实验价值的数据进行实验测试，使得实验结果欠缺更坚实的数据支撑。