课后作业：建立算法的管道模型

作者：欧新宇（Xinyu OU）

本文档所展示的测试结果，均运行于：Intel Core i7-7700K CPU 4.2GHz

【作业提交】

将分类结果保存到文本文档进行提交(写上每一题的题号和题目，然后再贴答案)，同时提交源代码。

1. 测试结果命名为: ex11-结果-你的学号-你的姓名.txt
2. 源代码命名为：ex1101-你的学号-你的姓名.py

结果文件，要求每小题标注题号，两题之间要求空一行

- 要求在 "乳腺癌" 数据集(breast_cancer)上完成以下任务，要求如下：
    1. 要求 `训练集`、`验证集` 的分割比例为：70%:30%
    2. 使用 `StratifiedKFold` 算法进行**分层3折交叉验证**
    3. 所有算法的 `随机参数` random_state=8
    4. 结果保留3位小数
    5. 输出训练集和测试集的预测结果
    6. 要求在网格搜索中测试两种预处理算法：StandardScaler(), MinMaxScaler()
    7. 要求在网格搜索中测试两种分类算法：神经网络MLPClassifier和高斯贝叶斯GaussianNB。两种算法要求测试以下参数：
        - MLP需要搜索的参数列表为：
            - alpha: [1e-6, 5e-5, 1e-5, 5e-4, 1e-4, 5e-3, 1e-3, 5e-2, 1e-2],
            - hidden_layer_sizes: [[20], [50], [100], [150], [200], [10, 10],[20, 20],[50, 50]]}
        - 高斯贝叶斯需要搜索的参数列表为：
            - var_smoothing: np.logspace(-15,10,26)

> **提示**：
>
> 1. 在训练集上训练模型，并使用 `交叉验证` 和 `网格搜索` 获取**最优参数**
> 2. 使用最优模型（或最优参数）在训练验证集上重新进行训练生成最终模型
> 3. 输出训练验证集上的评分和测试集上的评分

对于乳腺癌数据集有如下操作提示：

- 可以使用 `print(cancer.keys())` 查看肿瘤数据集包含哪些字段
- 使用 `print(cancer['字段'])` 观察该字段的内容，例如使用 `print(cancer['data'])` 查看数据的信息

1. 载入MNIST数据集

```python
# TODO: 1. 导入必须库 以及 定义必要的函数
# 导入机器学习数据集处理工具
from sklearn import datasets
from sklearn.model_selection import train_test_split

# TODO: 2. 创建/导入数据
cancer = datasets.load_breast_cancer()

# TODO: 3. 数据预处理，包括训练集、测试集划分，数据正则化，数据清洗等
X = cancer.data
y = cancer.target
X_trainval, X_test, y_trainval, y_test = train_test_split(X, y, train_size=0.7, random_state=8)
# X_train, X_val, y_train, y_val = train_test_split(X_trainval, y_trainval, train_size=0.7)
```

2. 使用管道模型和网格搜索选择最优模型

```python
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import MinMaxScaler

from sklearn.naive_bayes import GaussianNB
# 导入MLP神经网络包
from sklearn.neural_network import MLPClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.pipeline import Pipeline
from sklearn.model_selection import StratifiedKFold
import numpy as np

SKF = StratifiedKFold(n_splits=3, random_state=8, shuffle=False)

params = [{'scaler':[StandardScaler(),MinMaxScaler()],
           'cls':[MLPClassifier(random_state=8, max_iter=2000)],
           'cls__alpha':[1e-6, 5e-5, 1e-5, 5e-4, 1e-4, 5e-3, 1e-3, 5e-2, 1e-2],
           'cls__hidden_layer_sizes':[[20], [50], [100], [150], [200], [10, 10],[20, 20],[50, 50]]},
          {'scaler':[StandardScaler(),MinMaxScaler()],
           'cls':[GaussianNB()],
           'cls__var_smoothing':np.logspace(-15,10,26)}]

pipe = Pipeline([('scaler',StandardScaler()),('cls',MLPClassifier())])
grid = GridSearchCV(pipe, params, cv=SKF, verbose=1,n_jobs=-1)
grid.fit(X_trainval,y_trainval)

print('最佳模型是：\n{}'.format(grid.best_params_))
print('交叉验证平均分:{:.3f}'.format(grid.best_score_))
```

```
Fitting 3 folds for each of 196 candidates, totalling 588 fits
```

```
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 8 concurrent workers.
[Parallel(n_jobs=-1)]: Done  34 tasks      | elapsed:    7.6s
[Parallel(n_jobs=-1)]: Done 184 tasks      | elapsed:   26.5s
[Parallel(n_jobs=-1)]: Done 434 tasks      | elapsed:   56.2s
[Parallel(n_jobs=-1)]: Done 588 out of 588 | elapsed:   56.8s finished
```

```
最佳模型是:
{'cls': MLPClassifier(activation='relu', alpha=1e-06, batch_size='auto',
beta_1=0.9,
              beta_2=0.999, early_stopping=False, epsilon=1e-08,
              hidden_layer_sizes=[50], learning_rate='constant',
              learning_rate_init=0.001, max_iter=2000, momentum=0.9,
              n_iter_no_change=10, nesterovs_momentum=True, power_t=0.5,
              random_state=8, shuffle=True, solver='adam', tol=0.0001,
              validation_fraction=0.1, verbose=False, warm_start=False),
'cls__alpha': 1e-06, 'cls__hidden_layer_sizes': [50], 'scaler':
StandardScaler(copy=True, with_mean=True, with_std=True)}
交叉验证平均分:0.980
```

3. 保持参数不变，在训练验证集上重新训练，并输出结果

```
score_trainval = grid.score(X_trainval, y_trainval)
score_test = grid.score(X_test, y_test)

print('基于GridSearch方法的输出，训练集得分: {:.3f}，测试集得分:
{:.3f}'.format(score_trainval, score_test))
```

```
基于GridSearch方法的输出，训练集得分: 0.995，测试集得分: 0.977
```