ML LR Demo

机器学习一般我们基于 scikit-learn 框架来实现，机器学习的整个流程一般包括模型训练、模型推理部署两个主要阶段：

模型训练流程 纯文本

```
1．数据加载
2．数据转换处理(特征工程)
3．模型创建
4．模型训练
5．模型评估
6．模型持久化
```

模型推理部署流程 纯文本

```
1．模型恢复(一般和模型持久化对应)
2．数据转换处理：对待预测数据进行和训练相同的数据转换操作
3．模型预测
4．后处理：基于模型预测结果进行转换构造要求的输出格式数据

PS1：
#### 其中模型恢复仅需要恢复一次，其它操作需要针对每个待预测样本进行处理转换，
#### 故一般建议使用面向对象的方式来编写程序。

PS2：
#### 最终部署模型的时候，一般是对外提供一个类似HTTP的接口
#### 实际上和本地推理测试的逻辑相同，仅额外在外围增加一个接收HTTP请求参数以及处理结果返回的框架代码；
```

一、模型训练

import 导包 Python

```python
import os
import warnings
import numpy as np
from sklearn.datasets import make_circles
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LogisticRegression
from sklearn import metrics
import joblib
import json

warnings.filterwarnings('ignore')
```

1.1 数据准备

功能：从数据源加载数据到内存中；

PS: 当前准备构造一个双圆分类数据，内部是一个类别，外部是另外一个类别数据；

加载数据 Python

```python
# 1. 加载数据
X, Y = make_circles(
    n_samples=1000,  # 样本数目
    noise=0.1,  # 噪声样本比例
    factor=0.2,  # 内圈直径是外圈直径的factor倍
    random_state=24  # 随机数种子
)
x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=0.2, random_s
tate=24)
print(f"训练数据shape形状为: {type(x_train)} - {x_train.shape} -- {type(y_train)} -
{y_train.shape}")
print(f"评估数据shape形状为: {type(x_test)} - {x_test.shape} -- {type(y_test)} - {y_
test.shape}")
print(f"类别取值: {np.unique(y_train)} - {np.bincount(y_train)} -- {np.unique(y_tes
t)} - {np.bincount(y_test)}")
```

1.2 数据转换处理(特征工程)

功能：对原始数据进行特征处理，包括：异常值处理、标准化处理、多项式扩展等各种操作(可选)。

特征工程/数据转换 Markdown

```python
# 2. 特征工程
poly = PolynomialFeatures(degree=2)
x_train = poly.fit_transform(x_train)
x_test = poly.transform(x_test)
print(f"多项式扩展转换规则: {poly.get_feature_names_out(['x1', 'x2'])}")
print(f"转换后训练数据shape形状为: {type(x_train)} - {x_train.shape}")
print(f"转换后评估数据shape形状为: {type(x_test)} - {x_test.shape}")
```

1.3 模型创建

功能：使用 sklearn 提供封装好的算法模型对象，直接创建模型对象，对象内部包含了模型结构、损失函数、优化器等信息；

模型创建 Python

```python
# 3. 模型创建
algo = LogisticRegression(max_iter=10)
```

1.4 模型训练

功能：调用算法模型对应的训练方法，执行内部封装好的迭代训练过程；

模型训练 Python

```python
# 4. 模型训练
algo.fit(x_train, y_train)
```

1.5 模型评估

功能：针对某种特定数据集，使用训练过的模型进行推理预测得到预测结果，将预测结果和实际结果进行判断比较，从而得到最终的模型评估指标值，分类任务中常见评估指标有：准确率、F1、召回率、精确率等；

PS: 当评估指标不好的时候，一般需要回到最开始的时候进行数据转换的处理或者模型的更改等操作；

模型评估 Python

```python
# 5. 模型评估
pred_train = algo.predict(x_train)
pred_test = algo.predict(x_test)
print(f"预测结果类型: {type(pred_train)} - {pred_train.shape}")
print(f"训练数据上的准确率: {metrics.accuracy_score(y_train, pred_train)}")
print(f"评估数据上的准确率: {metrics.accuracy_score(y_test, pred_test)}")
print(f"训练数据上的分类报告: \n{metrics.classification_report(y_train, pred_train)}
```

```
\n")
print(f"评估数据上的分类报告: \n{metrics.classification_report(y_test, pred_test)}
\n")
```

1.6 模型持久化

功能：当训练好模型后(模型评估指标满足要求)，为了后续的模型评估部署，会将模型进行持久化保存到磁盘。针对不同的模型结构，可以采取不同的持久化方式。

▼ 模型持久化 Python

```python
# 6. 模型持久化
joblib_dump_file = "./output/01/ml.pkl"
os.makedirs(os.path.dirname(joblib_dump_file), exist_ok=True)
joblib.dump({
    'poly': poly,
    'algo': algo
}, joblib_dump_file)

json_dump_file = "./output/01/ml.json"
with open(json_dump_file, "w", encoding="utf-8") as writer:
    json.dump(
        {
            'poly': poly.get_feature_names_out(['x1', 'x2']).tolist(),  # 获取多项式的组合规则，并转换为list输出
            'algo': {
                'intercept': algo.intercept_.tolist(),  # 获取LR的截距项，并转换为list输出
                'coef': algo.coef_.tolist()  # 提取LR的参数项，并转换为list输出(json默认仅支持普通python类型)
            }
        },  # 持久化的对象
        writer,  # 输出文件对象
        indent=2,  # json格式化空格 -- 每个级别前面空2个空格
        ensure_ascii=False  # 中文不进行编码输出，直接输出中文
    )
```

二、模型推理

PS: 仅基于 joblib 持久化的模型进行推理预测

▼ import 导包 Python

```python
import joblib
```

2.1 模型恢复

功能：从持久化好的文件中恢复好模型对象，包括参数以及结构(执行逻辑)；

模型恢复 Python

```python
# 1. 模型恢复
obj = joblib.load("./output/01/ml.pkl")
poly = obj['poly']
algo = obj['algo']
print(f"模型恢复完成: {poly} -- {algo}")
```

2.2 数据转换

功能：使用和训练数据相同的数据处理流程，对待预测数据进行转换处理(如果训练时候没有做数据转换操作，那么推理的时候也不需要进行)；

数据转换 Python

```python
# 2. 数据转换
x = [
    [0.05, -0.01],
    [0.1, 0.3],
    [-0.4, 0.2],
    [1.0, 1.2],
    [0.0, 0.75],
    [0.0, -1.2]
]
x = poly.transform(x)
```

2.3 模型预测

功能：调用模型对应的预测方法，获取预测结果；根据不同的需要，可能需要调用不同的预测方法，比如：返回预测类别 id、返回预测属于各个类别的概率等等；

模型预测 Python

```python
# 3. 模型预测
y_pred_proba = algo.predict_proba(x)
print(f"获取预测概率对象为: {type(y_pred_proba)} - {y_pred_proba.shape}")
```

2.4 结果转换

功能：结合调用方的要求，对模型的预测结果进行转换输出；

结果转换 Python

```python
# 4. 结果拼接
```

```
y_pred_idx_per_sample = np.argmax(y_pred_proba, axis=1).tolist()
y_pred_proba_per_sample = y_pred_proba[range(len(y_pred_idx_per_sample)), y_pred_
idx_per_sample].round(3).tolist()
result = list(map(lambda t: {'id': t[0], 'proba': t[1]}, zip(y_pred_idx_per_sampl
e, y_pred_proba_per_sample)))
print(result)
# [{'id': 1, 'proba': 0.969}, {'id': 1, 'proba': 0.939}, {'id': 1, 'proba': 0.88
8}, {'id': 0, 'proba': 1.0}, {'id': 0, 'proba': 0.627}, {'id': 0, 'proba': 0.99
9}]
```