In [ ]:
```
# 安装nibabel库,只需要运行一次
!pip install nibabel -i https://pypi.douban.com/simple/
```

In [ ]:
```
# 解压数据集, 只需要运行一次
!unzip /home/aistudio/data/data229672/脑PET图像分析和疾病预测挑战赛数据集.zip -d
```

In [ ]:
```
# 重命名
!mv ─╘PET═╝╧ё╗╓╬ў║═╝▓6╘д▓т╠Ї╒╜╚№╝л┐к╩¤╛▌ 脑PET图像分析和疾病预测挑战赛数据集
```

基于脑PET图像的疾病预测挑战赛

导言

近年来，深度学习在医学图像处理和疾病预测方面取得了显著的进展。脑PET（Positron Emission Tomography）图像是一种用于研究脑部功能和代谢活动的重要影像技术。在本篇博客中，我们将使用Python编程语言和一些常用的深度学习库，展示如何处理脑PET图像数据，并构建逻辑回归模型来进行脑PET图像的疾病预测。

数据集介绍

为研究基于脑PET图像的疾病预测，本次大赛提供了海量脑PET数据集作为脑PET图像检测数据库的训练样本，参赛者需根据提供的样本构建模型，对轻度认知障碍进行分析和预测。脑PET图像检测数据库，记录了老年人受试志愿者的脑PET影像资料，其中包括确诊为轻度认知障碍（MCI）患者的脑部影像数据和健康人（NC）的脑部影像数据。 被试者按医学诊断分为两类：

- NC：健康
- MCI：轻度认知障碍

本次大赛所用脑PET图像检测数据库，图像格式为nii。

数据类别	文件夹名称	数值格式	解释
基本数据	NC	nii	正常老年人数据
	MCI	nii	轻度认知障碍患者数据

本次竞赛的评价标准采用F1_score，分数越高，效果越好。

步骤一：数据准备

首先，我们需要导入一些必要的Python库来处理图像数据和构建模型。以下是导入的库：

我们使用 `glob` 库来获取文件路径， `numpy` 用于数值计算， `pandas` 用于数据处理， `nibabel` 用于加载和处理医学图像数据， `OrthoSlicer3D` 用于图像可视化， `Counter` 用于计数统计。

In [ ]:
```
import glob                 # 用于获取文件路径
import numpy as np
import pandas as pd
import nibabel as nib       # 处理医学图像数据
from nibabel.viewers import OrthoSlicer3D    # 图像可视化
from collections import Counter              # 计数统计
```

步骤二：数据预处理

接下来，我们将读取训练集和测试集的文件路径，并对它们进行随机打乱，以保证数据的随机性。

In [ ]:
```
# 读取训练集文件路径,得到包含训练集与测试集路径的列表
train_path = glob.glob('./脑PET图像分析和疾病预测挑战赛数据集/Train/*/*')
test_path = glob.glob('./脑PET图像分析和疾病预测挑战赛数据集/Test/*')

# 打乱训练集和测试集的顺序(train_path,test_path是两个list)
np.random.shuffle(train_path)
np.random.shuffle(test_path)
```

步骤三：特征提取

对于深度学习任务，特征提取是非常重要的一步。在本例中，我们定义了一个函数 `extract_feature` ，用于从脑PET图像中提取特征。

`extract_feature` 函数从文件路径加载PET图像数据，并从中随机选择10个通道。然后，它计算了一系列统计特征，如非零像素数量、零像素数量、平均值、标准差等。最后，函数根据文件路径判断样本类别，并将提取到的特征和类别作为返回值。

In [ ]:
```
def extract_feature(path):
    # 加载PET图像数据
    img = nib.load(path)
    # 获取第一个通道的数据
    img = img.dataobj[:, :, :, 0]
    # 随机筛选其中的10个通道提取特征
    # np.random.choice表示从一个范围内抽取多少个
    random_img = img[:, :, np.random.choice(range(img.shape[2]), 10)]

    # 对图片计算统计值,这个列表作为函数的返回值
    feat = [
        (random_img != 0).sum(),                # 非零像素的数量
        (random_img == 0).sum(),                # 零像素的数量
        random_img.mean(),                      # 平均值
        random_img.std(),                       # 标准差
        len(np.where(random_img.mean(0))[0]),   # 在列方向上平均值不为零的数量
        len(np.where(random_img.mean(1))[0]),   # 在行方向上平均值不为零的数量
        random_img.mean(0).max(),               # 列方向上的最大平均值
        random_img.mean(1).max()                # 行方向上的最大平均值
    ]

    # 根据路径判断样本类别('NC'表示正常， 'MCI'表示异常),并且将判断的类别增加到fea
    if 'NC' in path:
        return feat + ['NC']
```

```
    else:
        return feat + ['MCI']
```

步骤四：模型训练

在这一步骤中，我们将利用 `extract_feature` 函数提取训练集和测试集的特征，并使用逻辑回归模型对训练集进行训练。

在这里，我们通过循环将特征提取过程重复进行30次，这是为了增加训练样本的多样性。然后，我们使用逻辑回归模型 `LogisticRegression` 来训练数据。在训练完成后，模型已经学习到了从特征到类别的映射关系。

In [ ]:
```
# 对训练集进行30次特征提取,每次提取的特征随机,每次提取后的特征以及类别（'NC'表示正
train_feat = []
for _ in range(30):
    for path in train_path:
        train_feat.append(extract_feature(path))

# 对测试集进行30次特征提取,每次提取的特征随机
test_feat = []
for _ in range(30):
    for path in test_path:
        # 这里所有测试集都被标记为"MCI",其实不准确,因为标记为"MCI"只是因为路径中没
        test_feat.append(extract_feature(path))

# 使用训练集的特征作为输入，训练集的类别作为输出，对逻辑回归模型进行训练。
from sklearn.linear_model import LogisticRegression
# 设置了最大迭代次数
m = LogisticRegression(max_iter=2000)
# 模型拟合
m.fit(
    # 取了除了最后一列切片
    np.array(train_feat)[:, :-1].astype(np.float32),  # 特征
    # 取了最后一列切片为标签
    np.array(train_feat)[:, -1]                       # 类别
)
```

在 `scikit-learn` （sklearn）中，除了逻辑回归（Logistic Regression）之外，还有许多其他的机器学习模型可以用于分类任务中，以下是一些常用于分类任务的机器学习模型：

1. 支持向量机（Support Vector Machines，SVM）：用于二分类和多分类问题，通过构建一个超平面来区分不同类别的样本。
2. 决策树（Decision Trees）：适用于二分类和多分类问题，通过对特征空间进行划分来分类样本。
3. 随机森林（Random Forests）：基于多个决策树的集成算法，用于二分类和多分类问题，提高了模型的泛化能力。
4. K最近邻算法（K-Nearest Neighbors，KNN）：根据最近邻样本的类别来分类新样本，适用于二分类和多分类问题。
5. 朴素贝叶斯（Naive Bayes）：基于贝叶斯定理的分类方法，适用于文本分类等问题。
6. 多层感知器（Multi-layer Perceptrons，MLP）：一种人工神经网络，用于解决复杂的分类问题。

7. 卷积神经网络（Convolutional Neural Networks，CNN）：专用于处理图像和视觉数据的神经网络，在图像分类任务中表现出色。

这些模型在分类任务中有不同的应用场景和性能表现，取决于数据集的特征、样本数量和问题的复杂性。在实际应用中，通常需要根据数据集的特点和具体任务来选择合适的分类模型，并进行模型调参和性能评估，以达到最佳的分类效果。

步骤五：预测与结果提交

在这一步骤中，我们使用训练好的逻辑回归模型对测试集进行预测，并将预测结果进行投票，选出最多的类别作为该样本的最终预测类别。最后，我们将预测结果存储在CSV文件中并提交结果。

具体来说，我们使用了 `Counter` 来统计每个样本的30次预测结果中最多的类别，并将结果存储在 `test_pred_label` 列表中。然后，我们将样本ID和对应的预测类别存储在一个DataFrame中，并将其按照ID排序后保存为CSV文件，这样我们就得到了最终的结果提交文件。

In [ ]:
```
# 对测试集进行预测并进行转置操作,使得每个样本有30次预测结果,此时的预测结果是一个一
# 于是以30为单位(因为每个预测样本进行了30次特征提取,使得一个预测样本对应了30个不同
test_pred = m.predict(np.array(test_feat)[:, :-1].astype(np.float32))
# 转置是为了使得行为预测样本数,列为每个预测样本的n次特征提取后预测的标签
test_pred = test_pred.reshape(30, -1).T

# 对每个样本的30次预测结果进行投票，选出最多的类别作为该样本的最终预测类别，存储在
# Counter(x)返回一个列表,列表中位序第(1)的是最大出现次数与对应的标签,组成的元组组
test_pred_label = [Counter(x).most_common(1)[0][0] for x in test_pred]
# 生成提交结果的DataFrame，其中包括样本ID和预测类别。
submit = pd.DataFrame(
    {
        'uuid': [int(x.split('/')[-1][:-4]) for x in test_path],  # 提取测试集文
        'label': test_pred_label                                  # 预测的类别
    }
)
# 按照ID对结果排序并保存为CSV文件
submit = submit.sort_values(by='uuid')
# 转化为CSV文件
submit.to_csv('submit.csv', index=None)
```

总结

本篇baseline介绍了如何使用Python编程语言和机器学习库处理脑PET图像数据，并构建逻辑回归模型来进行脑PET图像的疾病预测。特征提取是一个关键步骤，通过合适的特征提取方法，可以更好地表征图像数据。逻辑回归模型在本例中是一个简单而有效的分类器，但在实际应用中，可能需要更复杂的深度学习模型来提高预测性能。希望本篇baseline对你学习深度学习在医学图像处理中的应用有所帮助！

如果你对这个挑战赛感兴趣，不妨动手尝试一下。你可以在竞赛中改进模型和特征提取方法，或者尝试使用其他深度学习模型来进行预测。祝你好运！