学习笔记

任务一：

一.python入门

1. 基础语法，操作等

\_\_init\_\_ 是 Python 中的一个特殊方法，用于初始化对象的属性。它是类的构造函数（constructor），在创建类的实例时自动调用

1. Numpy库的使用

np.bincount() 是 NumPy 中的一个函数，用于统计非负整数数组中每个值的出现次数

np.argmax() 是 NumPy 中的一个函数，用于返回数组中最大值所在的索引

np.shape 用于获取数组的形状（即数组的维度信息）

np.unique 用于提取数组中的唯一值（去重），并可以返回唯一值的索引、计数等信息

for feature in x.shape[1]: 和 for feature\_index in range(X.shape[1]):前一种写发错

if (not) isinstance检查对象类型

np[:,n]表示索引数组内所有行内第n列数据

max() 函数：

1找出可迭代对象中的最大值

2找出多个参数中的最大值

3. 使用 key 参数

max() 函数有一个可选的 key 参数，它允许你传入一个函数，该函数会对每个元素进行处理，然后根据处理结果来确定最大值。

np.random.choice 是 NumPy 库中的一个函数，用于从给定的数组或列表中随机抽取样本。它常用于随机采样、打乱顺序或生成随机序列。

1. pandas库

a.数据加载和存储（iris.data.features和iris.data.targets）。

b.缺失值检查（isnull().sum()）。

c.缺失值填充（fillna()）。

d.数据划分（train\_test\_split）。

e.特征名称获取（X.columns）。

1. 4.sklearn库

a.数据划分：使用 train\_test\_split 将数据集划分为训练集和测试集。

b.模型评估：使用 KFold 和 cross\_val\_score 进行 K 折交叉验证。使用 accuracy\_score 和 classification\_report 计算模型的准确率和生成分类报告。

c.模型保存和加载：使用 joblib 保存和加载模型。

决策树

1.理论与理解

a.应该将样本数据的特征按照怎样的顺序添加到一颗决策树的各级结点中？这便是构建决策树所需要关注的问题核心。

2.熵

a. 作用，定义和计算

熵是表示随机变量不确定性的度量。说简单点就是物体内部的混乱程度。故选取能最大程度降低熵的特征在前，熵是衡量分类效果的指标。

1. 条件熵:

在给定 𝑋 的条件下 𝑌 的条件概率分布对 𝑋 的数学期望。即依靠某一特征划分出子集后得到的整个数据集的熵。

3.划分选择

选择最优评估标准来划分特征

1. 信息增益（ID3）
2. 信息增益率（C4.5）
3. 基尼系数（CART）

我们选择基尼指数，原因如下

基尼指数避免了复杂的对数运算，同时具备熵模型的优点。（Gini不纯度：计算简单，适合大数据集。

信息熵：计算稍微复杂，但对数据的不确定性描述更准确。）

具体操作

4.连续值处理

把连续取值的数据值域划分为多个区间，并将每个区间视为该特征的一个取值，如此就完成了从连续性数据到离散性数据的转变。

计算连续型特征中间值寻找阈值时，不能改变原有数组的特征排列顺序

5.预剪枝避免过拟合

a. 限制决策树的深度

b. 限制决策树中叶子结点的个数

c. 限制决策树中叶子结点包含的样本个数

d. 限制决策树的最低信息增益

随机森林

集成学习通过训练学习出多个估计器，当需要预测时通过结合器将多个估计器的结果整合起来当作最后的结果输出。

Bagging算法

  Bagging 算法是一种集成学习算法，其全称为自助聚集算法（Bootstrap aggregating），顾名思义算法由 Bootstrap 与 Aggregating 两部分组成。

算法的具体步骤为：假设有一个大小为 N 的训练数据集，每次从该数据集中有放回的取选出大小为 M 的子数据集，一共选 K 次，根据这 K 个子数据集，训练学习出 K 个模型。当要预测的时候，使用这 K 个模型进行预测，再通过取平均值或者多数分类的方式，得到最后的预测结果。（本次不选取特征子集）

鸢尾花数据集

1. 特征

数据集大小：鸢尾花数据集非常小，只有 150 个样本。

数据分布：数据集是均衡的，每个类别有 50 个样本。

特征数量：只有 4 个特征，特征数量较少。

故针对鸢尾花数据集特征小的特点，我们不划分特征子集

集成学习随机性

1. 数据采样随机性

Bootstrap采样：Bagging方法（如随机森林）通过有放回抽样生成多个训练子集，每个子集用于训练不同的基学习器。

随机子空间：从特征空间中随机选取子集进行训练，增加基学习器的多样性。

2. 模型初始化随机性

神经网络：初始权重随机设置，导致不同的训练结果。

K均值聚类：初始聚类中心随机选择，影响最终聚类效果。

3. 参数随机性

随机森林：分裂节点时随机选择特征子集，增加树之间的差异性。

梯度提升树：引入随机梯度提升，随机选择样本或特征进行训练。

4. 集成方法随机性

投票法：基学习器的预测结果通过投票或加权平均结合，随机性影响最终结果。

堆叠法：使用元学习器结合基学习器的输出，随机性影响元学习器的训练。

5. 随机性对集成学习的影响

多样性：随机性增加基学习器的多样性，有助于提升集成模型的泛化能力。

稳定性：随机性可能导致结果波动，但通过多次实验或交叉验证可以缓解。

6. 控制随机性

固定随机种子：确保结果可复现。

交叉验证：减少随机性对模型评估的影响。

本次采用了以下方法：

Bootstrap 采样：每棵决策树使用不同的训练子集

5 折交叉验证：每棵决策树使用不同的特征子集。

多棵决策树的集成：通过投票或平均的方式得到最终预测结果。

特征重要性

特征重要性计算：

在 RandomForest 类中实现了 tree\_feature\_importance 和 forest\_feature\_importance 方法，用于计算特征重要性。

特征重要性基于每棵决策树中特征被用于分裂的次数，最终对所有树的结果取平均。

特征重要性可视化：

在 work.py 中，使用 matplotlib 绘制了特征重要性的柱状图。

图中展示了每个特征的重要性，便于直观理解特征对模型的贡献。

问题：

！cross\_val\_score 函数要求传入的模型（estimator 参数）必须实现 fit 方法。

！Pycharm虚拟环境搭建

！if X.isnull().any().any() or y.isnull().any():运行错误: y.isnull().any() 返回的是一个 Pandas Series，而不是一个布尔值：改为y.isnull().any().any()

！cross\_val\_score 无法克隆RandomForest 类，因为没有实现 get\_params 和 set\_params 方法。

！best\_split 方法未能找到有效的分裂点,node.threshold的值为None，导致分类比较是出错：

找不到有效分裂点原因：1. 数据中存在唯一值或常量特征，个特征的所有值都是相同的（即没有变化）

解决方案：if len(np.unique(X[:, feature])) == 1:

Continue

数据中存在缺失值或异常值

解决：X = X.fillna(X.mean()) # 用均值填充缺失值

y = y.fillna(y.mode().iloc[0]) # 用众数填充缺失值

3. 特征数量较少或特征选择不当

故本次不划分特征子集

4.数据集过于纯净

解决：# 检查数据集是否已经非常纯净

if len(np.unique(y)) == 1:

return best\_feature, best\_threshold

5.特征列的值范围过窄

解决：unique\_values = np.unique(x)

if len(unique\_values) < 2:

return []# 如果值范围过窄，返回空列表

6. 在bulid\_tree时 如果没有找到有效的分裂点，标记为叶子节点

if node.feature is None or node.threshold is None:

node.threshold = max(y\_classes, key=y.tolist().count)

return node

！y的数据类型从字符串转为整数标签哦

！K折交叉验证与测试集验证的区别

1. 目的不同

K折交叉验证：

目的：用于模型的超参数调优和模型性能评估。通过将训练数据分成多个子集，多次训练和验证模型，以获得更稳定的性能估计。

测试集验证：

目的：用于评估模型在未见过的数据上的性能，以模拟模型在实际应用中的表现。

2. 应用场景不同

K折交叉验证：

适用于超参数调优，例如选择最佳的树的数量、最大深度等。

适用于模型选择，例如比较不同模型的性能。

适用于数据量有限的情况，通过多次划分数据来充分利用有限的训练样本。

测试集验证：

适用于最终模型性能评估，在模型训练完成后，使用测试集来评估模型在未知数据上的表现。

适用于模型部署前的最终验证，确保模型在实际应用中的性能。

二者不一定需要同时存在，但二者结合使用可以提供更全面的评估。

任务二：

一.使用爬虫脚本爬取数据与清洗

1. 调整爬虫脚本

设置请求频率

为了避免对目标网站造成过大压力或被封禁IP，需要合理设置请求频率。通常可以通过在请求之间添加延迟（time.sleep）来实现。

1. 在 get\_bgm\_url 函数中设置请求频率
2. 在 get\_points 函数中设置请求频率

2.清洗爬取数据

数据质量问题通常可以分为以下几类：

缺失值：数据中存在缺失值，导致分析结果不准确或不完整。

异常值：数据中存在异常值，可能是由于测量误差或录入错误导致的。

一致性问题：数据中存在不一致的情况，比如同一属性的取值不一致。

不准确性：数据本身可能存在错误或不准确的情况。

而针对评论数据，可以有针对性的处理：

1. 过滤掉无意义的评论
2. 去除停用词
3. 对于中文文本，需要进行分词处理。
4. 去除极端评分（本次采用四分位/IQR法处理，因为大部分人对一部作品的评分应集中在一个区间，而IQR法适合处理偏态分布的数据）
5. 去除重复评论

二.模型微调（微调的目的是让模型适应特定任务（如 Bangumi 评论分数预测），因此需要在预训练模型的基础上进行调整。）

1.BERT模型特性

BERT 是通过掩码语言模型（Masked Language Model, MLM）和下一句预测（Next Sentence Prediction, NSP）任务进行预训练的。

预训练模型已经学习到了丰富的语言表示能力，因此微调时不需要从头训练，而是基于预训练模型的权重进行微调。

2. 学习率的选择

较小的学习率：

预训练模型的权重已经非常接近最优值，因此微调时使用较小的学习率（如 2e-5），以避免破坏预训练模型的权重。

较大的学习率可能导致模型权重更新过快，破坏预训练模型学到的语言表示。

学习率调整策略：

可以使用学习率调度器（如 linear scheduler）在训练过程中动态调整学习率。

3. 训练轮数的选择

避免过拟合：

训练轮数过多可能导致模型过拟合，即在训练集上表现很好，但在验证集上表现较差。

通常选择 3-5 轮训练，以平衡模型的泛化能力和性能。

早停法：

可以监控验证集的性能，如果性能不再提升，则提前停止训练。

4. 分类层的调整

任务特定的输出层：

预训练 BERT 模型的输出是一个高维向量，需要通过分类层将其映射到任务特定的类别（如 5 个评分类别）。

分类层的参数是随机初始化的，需要在训练过程中学习。

损失函数：

使用交叉熵损失函数（Cross-Entropy Loss）来计算模型输出与真实标签之间的差异。

5.数据预处理

文本长度：

将评论内容截断或填充到固定长度（如 128 个 token），以适应模型的输入要求。

较长的文本可能包含冗余信息，较短的文本可能导致信息丢失。

分词器：

使用与预训练模型相同的分词器（如 BertTokenizer），以确保输入格式一致。

6.评估指标

准确率：

使用准确率（Accuracy）作为评估指标，衡量模型在验证集上的性能。

对于分类任务，准确率是一个直观且常用的指标。

其他指标：

可以根据任务需求选择其他指标，如 F1 分数、精确率、召回率等

在本次任务中本次任务中，微调的主要内容包括：

模型结构：添加分类层，将 BERT 输出映射到 5 个评分类别。

数据预处理：对评论内容进行分词和填充，划分训练集和验证集。

训练参数：设置学习率、批量大小、训练轮数和权重衰减。

训练过程：使用 Trainer API 进行训练和评估，监控验证集性能。

出现问题：数据标签超出给定范围，重新映射到范围内

训练模型时未保存分词器相应文件，上传不到huggingface