**1.概述决策树、随机森林**的**算法**思路（信息熵）、关键操作（不超过600字，逻辑要通，**要有小标题**，要排版清晰，请不要网络大幅摘抄）。

决策树和随机森林是两种常用的机器学习算法，它们都可以用于分类和回归问题。

**一、决策树**

1. 算法思路

决策树是一种树形结构，每个内部节点表示一个特征的划分，每个叶节点表示一个类别或一个值。决策树的构建是一个自顶向下的递归过程，从根节点开始，根据数据集的特征和标签，选择一个最优的特征作为划分依据，将数据集分成若干子集，然后对每个子集重复这个过程，直到所有的子集都是纯的（即只包含一个类别或一个值）或者达到预设的停止条件（如最大深度、最小样本数等）。

1. 信息熵

信息熵是一种衡量数据集不确定性的指标，它反映了数据集中的信息量，信息熵越大，数据集越不纯，信息熵越小，数据集越纯。

1. 关键操作

3.1特征选择

特征选择是决策树构建的核心，它决定了如何划分数据集，从而影响了决策树的结构和性能。特征选择的目标是选择一个能够使数据集划分后的信息熵最小的特征，也就是说，选择一个能够使数据集纯度提升最大的特征。

3.2 信息增益

信息增益表示得知一个特征的信息后，数据集的信息熵减少的程度，也就是数据集纯度提升的程度。

3.3剪枝

剪枝是决策树的后处理过程，它的目的是为了防止决策树过拟合，提高决策树的泛化能力。剪枝的方法有预剪枝和后剪枝两种。

1. **随机森林**
2. 算法思路

随机森林是一种基于Bagging思想的集成学习算法，它由多棵决策树组成，每棵决策树都是在随机抽样的数据集和特征上训练得到的。随机森林的预测结果是由所有决策树的预测结果进行投票或平均得到的。随机森林的目的是通过多样化的决策树来降低单棵决策树的方差，提高预测的稳定性和准确性。

1. 信息熵

随机森林的信息熵是指随机森林中每棵决策树的信息熵的平均值，它反映了随机森林对数据集的不确定性。[随机森林的信息熵越小，表示随机森林对数据集的划分效果越好。](https://zhuanlan.zhihu.com/p/30357764)

3. 关键操作

3.1自助采样

自助采样是一种有放回的随机抽样方法，它可以从原始数据集中生成多个不同的子数据集，每个子数据集的样本数和原始数据集相同，但是由于有放回的抽样，每个子数据集中可能有重复的样本，也可能有缺失的样本。

3.2 随机子空间

随机子空间是一种在特征空间上进行随机抽样的方法，它可以从原始特征中抽取一部分特征作为每棵决策树的输入，而不是使用所有的特征。

3.2 集成方法

集成方法是一种将多个模型的预测结果进行整合的方法，它可以提高模型的准确性和稳定性。集成方法有两种常见的形式：投票法和平均法。

**2.利用决策树或随机森林解决问题（回归或分类）。**

附：Sklearn常用数据集

|  |  |  |
| --- | --- | --- |
| 序号 | 数据集 | 学号尾号 |
| 1 | 鸢尾花数据集：load\_iris（） | 0 1 |
| 2 | 手写数字数据集：load\_digits（） | 2 3 |
| 3 | 乳腺癌数据集load-barest-cancer（） | 4 5 8 |
| 4 | 糖尿病数据集：load-diabetes（） | 6 7 9 |

利用决策树或随机森林对以上数据集（学号尾号不同采用不同数据集）进行分析，找出最佳参数下的决策树或随机森林更适合本数据集（准确率更高，不同回归、不同参数下的不同准确率要列出）。

要求：

1）代码要列出，重点代码加注释说明，特别是自己调试过程中的自我理解；

2）运行结果要截图，结果要文字解释；

3）注意排版

import numpy as np  
import pandas as pd  
import sklearn.tree as tree  
import matplotlib.pyplot as plt  
from sklearn.tree import DecisionTreeClassifier  
from sklearn.datasets import load\_iris  
# 创建决策树分类器  
clf\_tree = tree.DecisionTreeClassifier()  
# 导入数据拆分工具  
from sklearn.model\_selection import train\_test\_split  
# 赋值样本特征  
data = load\_iris()  
x = data['data']  
y = data['target']  
# 将x,y拆分成训练集和数据集  
x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y, random\_state = 1)  
# 使用决策树分类器对数据进行拟合  
clf\_tree.fit(x\_train, y\_train)  
# 使用验证集评估模型准确率  
print('决策树测试集：',clf\_tree.score(x\_test,y\_test))  
from sklearn.ensemble import RandomForestClassifier  
# 创建空列表用于存储模型准确率  
score\_list = []  
# 创建从10到100的循环，步长为10  
for i in range(10,100,10):  
 # 让随机森林的n\_estimators参数以10为单位从10到100遍历  
 clf\_forest = RandomForestClassifier(n\_estimators = i, random\_state = 1)  
 # 不同参数的随机森林拟合训练集  
 clf\_forest.fit(x\_train, y\_train)  
 # 将模型在验证集中的准确率逐一添加到先前创建的空列表  
 score\_list.append(clf\_forest.score(x\_test, y\_test))  
plt.plot(range(10,100,10), score\_list)  
plt.show()  
print('随机森林测试集：',np.max(score\_list))

**结果：**

文本

描述已自动生成

结果显示，在random\_state = 0 or 1 or 5 or 100时，决策树测试集数据与随机森林相等，都适合鸢尾花数据集。

图表

描述已自动生成

在random\_state =10时的决策树准确率小于随机森林准确率，随机森林更适合本数据集。

文本

描述已自动生成

图片包含 图表

描述已自动生成

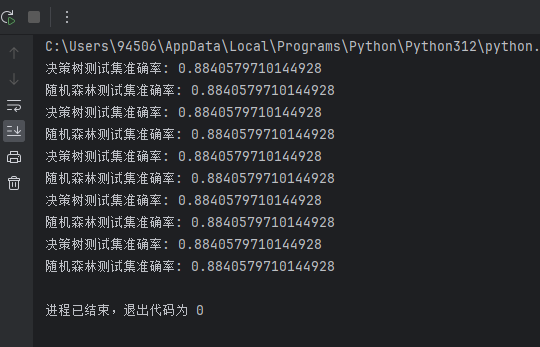
**3.[附加题].**利用**决策树或随机森林**解决**自己遇到的问题**。

[解题要求同题2]

**对帕尔默企鹅size数据集采用决策树和随机森林分析**

import numpy as np  
import pandas as pd  
import sklearn.tree as tree  
import matplotlib.pyplot as plt  
from sklearn.tree import DecisionTreeClassifier  
from sklearn.datasets import make\_blobs  
from sklearn.model\_selection import train\_test\_split  
  
data = pd.read\_csv('penguins\_size.csv', usecols=['culmen\_length\_mm', 'body\_mass\_g'])  
# 删除包含缺失值的行  
data.dropna(inplace=True)  
X = data.drop('culmen\_length\_mm', axis=1) # 特征数据  
y = data['body\_mass\_g'] # 目标数据  
# 将数据拆分为训练集和测试集  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=10)  
# 存储每次训练的准确率  
decision\_tree = []  
random\_forest = []  
for \_ in range(5):  
# 创建决策树分类器模型  
 clf\_tree = DecisionTreeClassifier()  
# 在训练集上训练模型  
 clf\_tree.fit(X\_train, y\_train)  
 test\_tree = clf\_tree.score(X\_test, y\_test)  
 decision\_tree.append(clf\_tree.score(X\_test, y\_test))  
 print("决策树测试集准确率:", test\_tree)  
  
 from sklearn.ensemble import RandomForestClassifier  
# 创建随机森林分类器模型  
 clf\_forest = RandomForestClassifier(n\_estimators=50, random\_state=10)  
# 在训练集上训练随机森林模型  
 clf\_forest.fit(X\_train, y\_train)  
 test\_forest = clf\_forest.score(X\_test, y\_test)  
 random\_forest.append(clf\_forest.score(X\_test, y\_test))  
 print("随机森林测试集准确率:", test\_forest)  
  
# 创建测试集准确率折线图  
plt.plot(range(1, 6), decision\_tree, marker='o', label='Decision Tree')  
plt.plot(range(1, 6), random\_forest, marker='x', label='Random Forest')  
plt.legend()  
plt.show()

**进行五次迭代训练的结果：**



在改变随机种子random\_state 的值时，决策树测试集准确率仍然和随机森林相等，因此可得出帕尔默企鹅数据集中的企鹅居住涵洞长度与企鹅身体大小之间的关系较简单，决策树可以很好地捕捉到这种关系，并且在测试集上表现良好。随机森林在这种情况下可能无法提供额外的性能优势。

图表, 箱线图

描述已自动生成