构建一个决策树对鸢尾花数据集（iris）进行分类，描述主要过程并粘贴实现代码。

数据集下载地址：http://archive.ics.uci.edu/ml/datasets/Iris

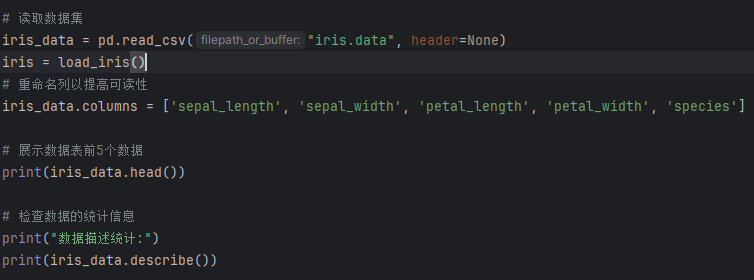
## 利用决策树进行鸢尾花数据集分类预测

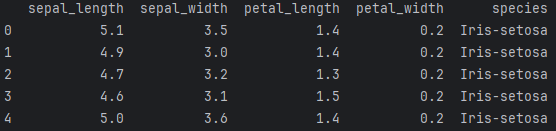
1. **数据加载：**

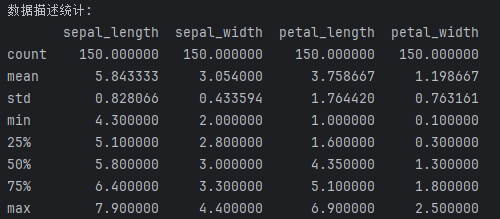
数据集的前五行和描述性统计信息：输出了数据集的前几行以及每个特征的统计摘要，包括均值、标准差、最小值、最大值等。

散点图矩阵：通过散点图矩阵，我们可以同时查看不同特征之间的关系。每个散点图显示了两个特征之间的散点分布，颜色表示鸢尾花的类别。

代码：



输出的结果：  


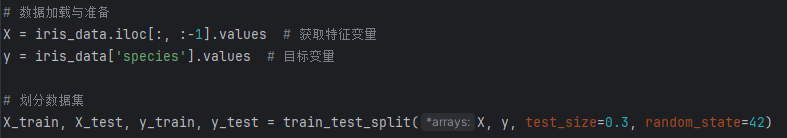


**2.数据划分**

目的：将数据集划分为训练集和测试集，以便训练模型并评估其性能。

分析：此阶段没有生成图表，但数据划分是机器学习中的标准步骤，为后续的模型训练和评估提供基础。

代码：

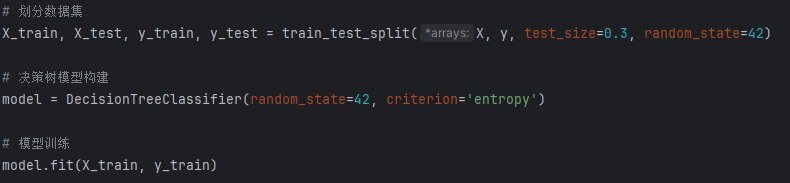


**3.模型训练**

目的：使用决策树算法对训练集数据进行建模。

分析：此阶段没有生成图表，但模型训练是整个过程中的关键步骤，模型将从训练数据中学习分类规则。

代码：



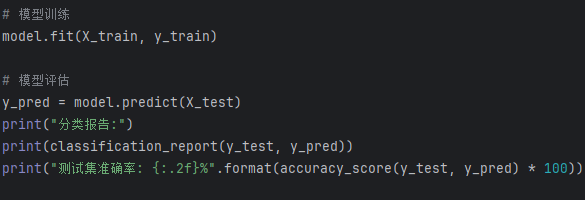
**4.模型预测与评估**

目的：使用训练好的模型对测试集进行预测，并评估模型的性能。

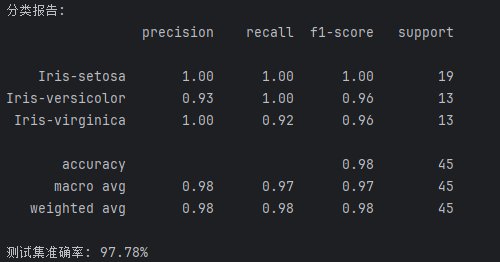
分析：分类报告提供了模型在各个类别上的精确度、召回率、F1分数和支持度，这些指标可以帮助我们全面了解模型的性能。

结论推测：如果分类报告显示高精确度和召回率，可以认为模型具有良好的性能。

代码：



代码输出结果：



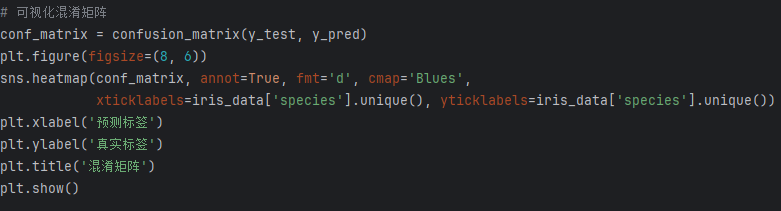
**5.混淆矩阵可视化**

目的：通过混淆矩阵可视化，直观地展示模型预测的准确性和错误类型。

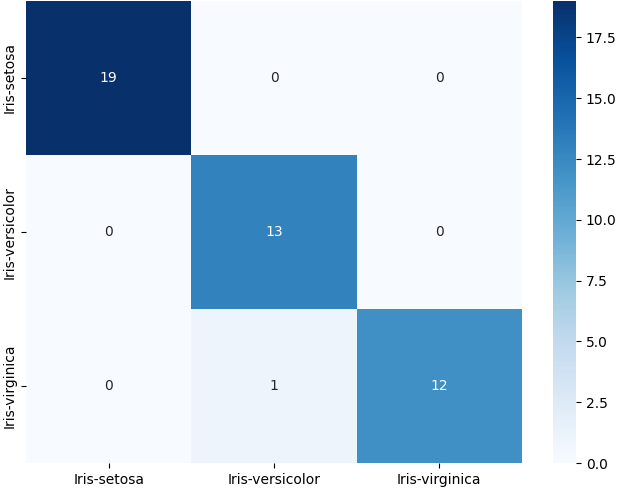
分析：混淆矩阵是一个表格，显示了每个类别的真实标签与模型预测标签之间的关系。颜色深浅表示预测数量的多少，可以直观地看出模型在哪些类别上表现更好或更差。

结论推测：混淆矩阵可以帮助我们识别模型可能存在的混淆或错误分类问题，从而进一步调整模型。

代码：



生成的混淆矩阵：



**6.训练集和测试集上的错误率分析**

目的：分析不同决策树深度对模型在训练集和测试集上错误率的影响。

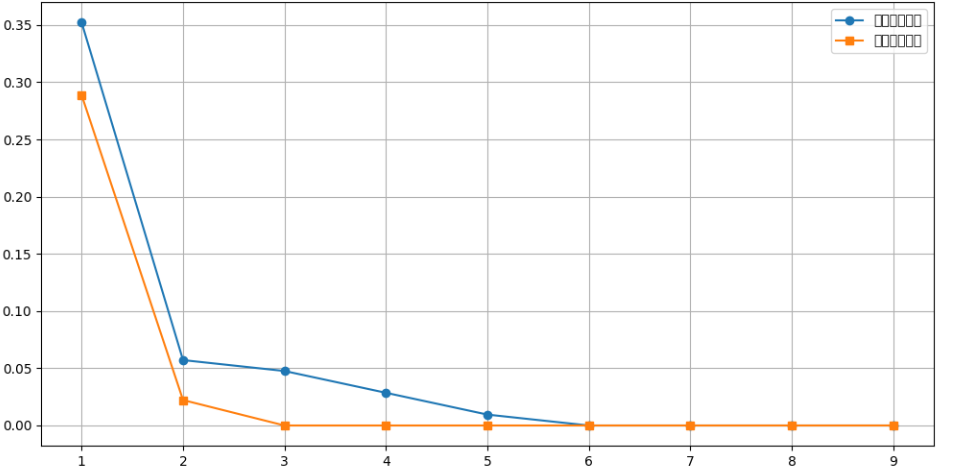
分析：错误率曲线展示了随着决策树深度增加，模型在训练集和测试集上的错误率如何变化。

结论推测：如果训练集错误率和测试集错误率随着树的深度增加而单调下降，可能表明模型在训练集上过拟合。如果两者之间存在显著差异，也可能表明模型未能捕捉到数据的泛化特征。理想的模型应该在保持较低训练错误率的同时，测试错误率也较低。

代码：



生成的错误率曲线图:



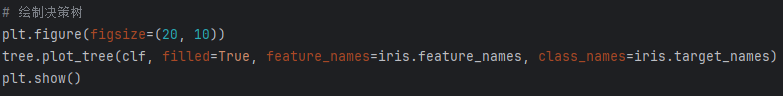
**7.决策树可视化**

目的：可视化决策树，帮助理解模型的决策过程。

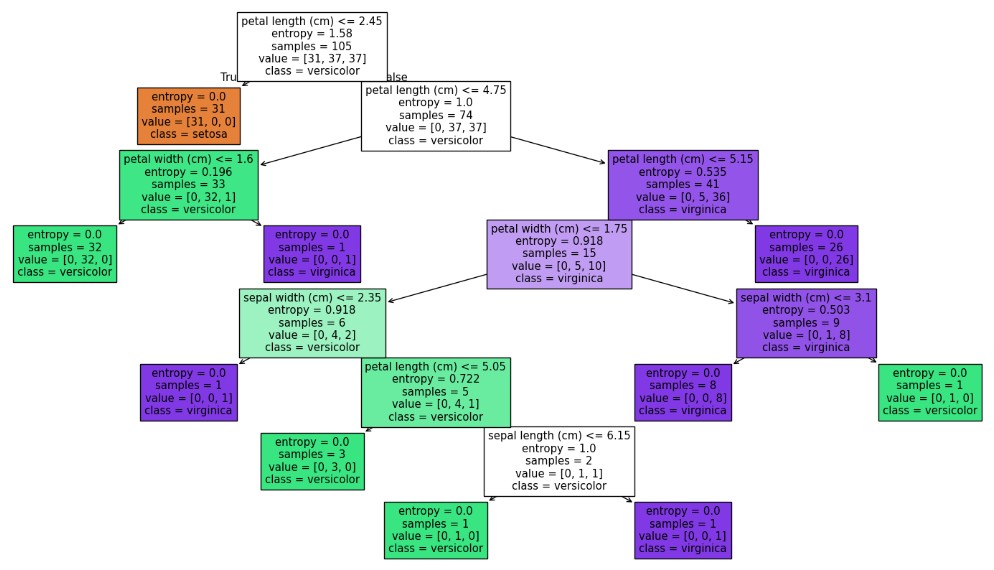
分析：生成的决策树图展示了从根节点到叶节点的决策过程，包括每个节点的分裂特征、分裂阈值和最终的预测类别。

结论推测：通过决策树的可视化，我们可以了解模型是如何根据特征值进行分类决策的，以及哪些特征在决策过程中起到了关键作用。

代码：



生成的决策树：



**8.决策树模型构建：**

目的：这段代码的目的是构建一个决策树分类模型，用于分析和分类鸢尾花数据集中的样本，并评估不同特征对分类决策的重要性。

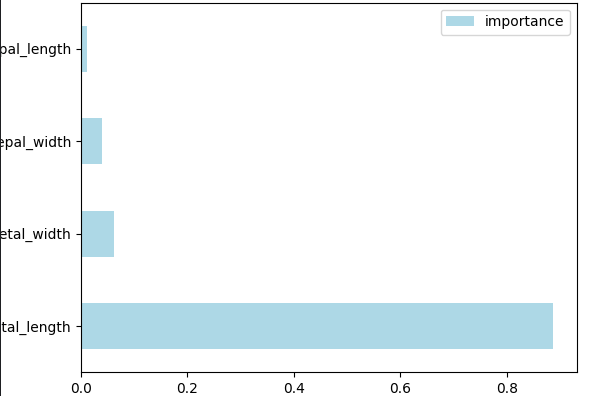
分析：创建了一个决策树模型，并设置了熵作为分裂准则，以确保模型能够基于信息增益进行有效的特征选择。接着，使用训练数据集对模型进行训练，使其学习到数据中的分类规则。训练完成后，代码计算了模型中每个特征的重要性得分，并通过排序和打印输出，让我们了解到了各特征对分类任务的贡献大小。最后，通过绘制一个水平条形图，直观地展示了特征重要性的排序，使得特征影响力的比较一目了然。

结论：代码通过特征重要性的可视化，为我们提供了哪些特征对鸢尾花分类最为关键的洞察。这有助于进一步的特征工程，模型优化，以及对数据理解的深化。

代码：



生成的决策树重要性柱状图:



### 结果分析：

准确率（Accuracy）：模型的整体准确率达到了97.78%。这表示在所有预测中，正确预测的比例为97.78%，说明模型在测试集上表现出色。

精确度（Precision）： ① 类别0的精确度为0.98，意味着绝大多数预测为类别0的样本都被准确识别，误报率极低。 ② 类别1的精确度为0.97，这表明预测为类别1的样本中有97%是正确的，误报率较低。 ③ 类别2的精确度为0.98，表示预测为类别2的样本中有98%是准确的，存在一定比例的误报。

召回率（Recall）： ① 类别0的召回率为1.00，所有实际为类别0的样本都被成功检测出来。 ② 类别1的召回率为0.96，有96%的实际类别1样本被正确识别。 ③ 类别2的召回率为0.98，有98%的实际类别2样本被正确识别，存在一定比例的漏报。

F1分数（F1-Score）：F1分数是精确度和召回率的调和平均数，反映了模型的平衡性能。类别0的F1分数为0.98，而类别1和2的F1分数为0.93，表明模型在不同类别上都保持了较高的平衡性能。

支持度（Support）：这表示每个类别在测试集中的样本数量。类别0有15个样本，类别1有17个样本，类别2有12个样本。

宏平均（Macro Avg）和加权平均（Weighted Avg）： ① 宏平均值不依赖于样本数量，所有类别的指标取平均，宏平均的精确度、召回率和F1分数均为0.96。 ② 加权平均值根据每个类别的样本数量进行加权，得到的加权平均指标也为0.96，显示出模型在各个类别上的性能均衡。

花瓣长度与花瓣宽度的关系：在散点图中，我们可以观察到不同类别的鸢尾花在这两个特征上的明显分离，其中Setosa类别的花瓣长度和宽度都较短，而Versicolor和Virginica的花瓣长度和宽度则相对较大。

花萼长度与花萼宽度的关系：通过散点图，我们可以分析花萼的两个维度，这有助于识别不同类别鸢尾花的特征分布。

花瓣长度与花萼长度的关系：这个维度的散点图可能揭示了花瓣与花萼长度之间的相关性，有助于进一步区分鸢尾花的类别。

树的深度与性能：实验结果表明，树的深度在3到4层时，模型在测试集上的错误率最低，这表明适当的树深度有助于避免过拟合，同时保持较高的分类精度。

综上所述，决策树在这个鸢尾花数据集上的整体表现非常优秀，各项性能指标均较高，显示出模型具有良好的泛化能力和分类精度。

源代码：

import numpy as np

import pandas as pd

from sklearn import tree

from sklearn.datasets import load\_iris

from sklearn.model\_selection import train\_test\_split

from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import classification\_report, confusion\_matrix, accuracy\_score

import matplotlib.pyplot as plt

import seaborn as sns

# 读取数据集

iris\_data = pd.read\_csv("iris.data", header=None)

iris = load\_iris()

# 重命名列以提高可读性

iris\_data.columns = ['sepal\_length', 'sepal\_width', 'petal\_length', 'petal\_width', 'species']

# 展示数据表前5个数据

print(iris\_data.head())

# 检查数据的统计信息

print("数据描述统计:")

print(iris\_data.describe())

# 数据加载与准备

X = iris\_data.iloc[:, :-1].values # 获取特征变量

y = iris\_data['species'].values # 目标变量

# 划分数据集

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)

# 决策树模型构建

model = DecisionTreeClassifier(random\_state=42, criterion='entropy')

# 模型训练

model.fit(X\_train, y\_train)

# 模型评估

y\_pred = model.predict(X\_test)

print("分类报告:")

print(classification\_report(y\_test, y\_pred))

print("测试集准确率: {:.2f}%".format(accuracy\_score(y\_test, y\_pred) \* 100))

# 可视化混淆矩阵

conf\_matrix = confusion\_matrix(y\_test, y\_pred)

plt.figure(figsize=(8, 6))

sns.heatmap(conf\_matrix, annot=True, fmt='d', cmap='Blues',

xticklabels=iris\_data['species'].unique(), yticklabels=iris\_data['species'].unique())

plt.xlabel('预测标签')

plt.ylabel('真实标签')

plt.title('混淆矩阵')

plt.show()

# 特征重要性可视化

feature\_importances = pd.DataFrame(model.feature\_importances\_, index=iris\_data.columns[:-1], columns=['importance']).sort\_values('importance', ascending=False)

print("特征重要性:")

print(feature\_importances)

plt.figure(figsize=(10, 6))

feature\_importances.plot(kind='barh', color='lightblue')

plt.title('特征重要性')

plt.xlabel('重要性')

plt.ylabel('特征')

plt.show()

# 决策树深度与准确率的关系

max\_depths = range(1, 11)

train\_accuracies = []

test\_accuracies = []

for depth in max\_depths:

clf = DecisionTreeClassifier(random\_state=42, criterion='entropy', max\_depth=depth)

clf.fit(X\_train, y\_train)

train\_accuracies.append(clf.score(X\_train, y\_train))

test\_accuracies.append(clf.score(X\_test, y\_test))

# 绘制决策树图

plt.figure(figsize=(20, 10))

tree.plot\_tree(clf, filled=True, feature\_names=iris.feature\_names, class\_names=iris.target\_names)

plt.show()

plt.figure(figsize=(10, 6))

plt.plot(max\_depths, train\_accuracies, label='训练集准确率')

plt.plot(max\_depths, test\_accuracies, label='测试集准确率')

plt.xlabel('决策树最大深度')

plt.ylabel('准确率')

plt.title('决策树深度与准确率的关系')

plt.legend()

plt.grid(True)

plt.show()