

**专业综合实践二大作业**

|  |  |
| --- | --- |
| **题 目：** | 阿拉伯数字4至9的识别与分类 |
| **学 院：** | 人工智能学院 |
| **专 业：** | 数据科学与大数据技术 |
| **年级班别：** | 2021级数据科学与大数据技术B班 |
| **学 号：** | 421470159 |
| **学生姓名**： | 张伟健 |
| **日期** | 2024.11.27 |

**提交日期：** 2024 **年** 11 月

# **1 摘要**

**摘要**：图像分类作为人工智能领域中至关重要的一项技术，其应用范围极为广泛。在本项深入的研究当中，我们特意选取了决策树、随机森林以及 KNN 这三种极具代表性的分类算法，将它们应用于针对数字 4、5、6、7、8、9 所构建的图像数据集的分类任务之中。我们开展此次研究的核心目标，在于全面且深入地评估这些分类算法在图像分类方面所展现出的性能表现。通过对它们的精确分析与对比，我们期望能够获取到详尽的数据和有价值的结论，进而为未来在图像分类领域展开的更为深入和广泛的研究工作提供极具参考性的依据。无论是在学术研究层面，还是在实际应用场景中，这些评估结果和参考信息都将具有不可忽视的重要意义。它们有助于研究人员和开发者在面对类似问题时，能够更加明智地选择合适的算法，从而优化图像分类的效果和效率，推动人工智能在图像分类及相关领域的不断发展与进步。同时，也为后续相关研究的开展奠定坚实的基础，指明清晰的方向。

关键词：图像分类；决策树；随机森林；KNN；性能比较

**目录**

[1 摘要 1](#_Toc23860)

[2绪论 1](#_Toc30804)

[2.1 [研究背景] 1](#_Toc24557)

[2.2 [研究目标] 1](#_Toc16804)

[2.3 [研究方法] 2](#_Toc18603)

[3数据 2](#_Toc3589)

[3.1 [数据描述] 2](#_Toc20952)

3.2 [数据处理] 3

[4 机器学习 4](#_Toc8486)

4.1 [决策树模型] 4

[4.2 [随机森林模型] 6](#_Toc8087)

4.3 [KNN(K邻近)模型] 7

[5数据可视化 11](#_Toc26918)

5.1 [可视化] 11

[6结论 13](#_Toc24578)

# **2绪论**

## 2.1 [研究背景]

在当今科技飞速发展的时代，图像分类在人工智能领域中占据着极为关键且广泛的地位。它的应用场景丰富多样，例如手写数字识别，这对于实现自动化办公、智能数据录入等具有重大意义，能够极大地提高工作效率和准确性；模式识别领域中，图像分类可助力快速准确地辨别各种复杂的模式，在工业生产、质量检测等方面发挥着重要作用；而在生物特征识别方面，它可以精确地对人脸、指纹等生物特征进行分类和识别，为安全认证、身份验证等提供了可靠的技术支持。

在数字图像分类这一重要的研究方向中，如何切实有效地提高分类器的性能无疑成为了一个至关重要且极具挑战性的研究课题。这不仅关系到能否准确地对图像进行分类，更直接影响到整个系统的可靠性和实用性。因为在实际应用中，图像的复杂性、多样性以及各种干扰因素的存在，都对分类器的性能提出了极高的要求。

为了应对这些挑战，研究人员们不断探索新的方法和技术。他们致力于通过深入研究算法原理、优化模型结构、利用更强大的计算资源等途径，来努力提升分类器的性能。同时，也在不断寻求创新的思路和解决方案，以适应不断变化的应用需求和技术发展趋势。在这样的背景下，对于数字图像分类中分类器性能提升的研究显得尤为紧迫和重要。

## 2.2 [研究目标]

本研究怀揣着明确且具有重要意义的目标前行。其核心目标在于全面而深入地比较决策树、随机森林和 KNN 这三种分类算法在数字图像分类这一特定任务中的具体性能表现。通过对它们在该任务中各个方面的详细分析与精准评估，试图揭开每一种算法的优势与不足，从而能够清晰地呈现出它们各自的特点。

我们期望通过这样的比较研究，能够准确地找到在数字图像分类任务中最为适合的那个分类器。这个分类器不仅需要在准确性方面表现出色，同时还应在其他关键性能指标上展现出卓越的特质，比如计算效率、稳定性、对不同数据集的适应性等。这样的分类器将为实际应用提供强大的支持，使得数字图像分类工作能够更加高效、准确地开展。

而且，我们的研究不仅仅局限于当下，还着眼于未来。我们希望通过此次研究得出的结论和经验，能够为后续在该领域展开的进一步研究提供极具价值的参考。为其他研究人员在面对类似问题或开展相关研究时，提供宝贵的启示和指引，避免他们重复走弯路，能够站在我们的研究基础上继续深入探索。帮助他们更好地理解和把握不同分类算法在数字图像分类中的应用潜力与局限，从而能够更加有针对性地选择和优化算法，推动整个数字图像分类研究领域的持续进步和发展。

## 2.3 [研究方法]

首先，针对数字 4、5、6、7、8、9 的图像数据集，实施全面细致的数据清洗工作。这一过程旨在去除数据集中可能存在的噪声、异常值等干扰因素，以确保数据的纯净度和可靠性。同时，将所有图像的大小统一调整为 28x28 像素，这不仅有利于后续算法处理的标准化，也为不同算法在相同条件下的比较奠定了基础。

接着，充分运用决策树、随机森林和 KNN 这三种分类算法，分别对经过预处理的图像数据集展开分类操作。每种算法都将依据其独特的原理和机制对图像进行分类，展现出各自的特点和优势。

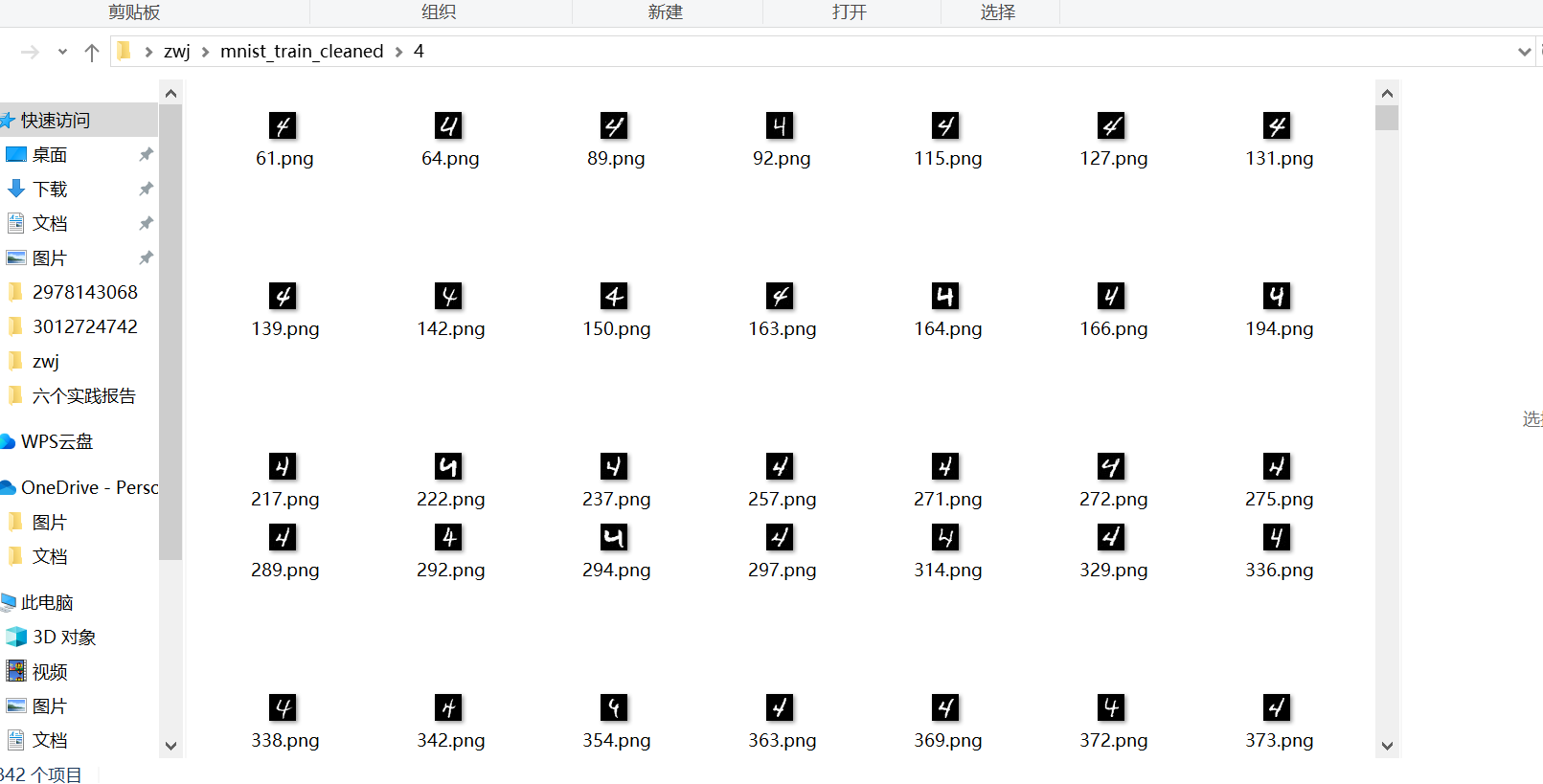
随后，通过严谨地计算准确率、精确率、召回率和 F1 分数等一系列关键的分类指标，来深入比较三种分类器在性能方面的差异。准确率反映了分类器整体的正确分类比例；精确率突出了分类器在阳性预测中的准确程度；召回率则强调了对真实阳性样本的捕捉能力；而 F1 分数则综合考量了精确率和召回率，提供了更为全面的性能评估视角。

最后，为了更直观地呈现三种分类器的性能差异，采用可视化手段展示它们各自的准确率、精确率、召回率和 F1 分数。通过直观的图表形式，能够让研究结果更加清晰易懂，便于研究者和读者迅速把握不同分类器的性能表现，为进一步的分析和讨论提供有力支持。

# **3数据**

## 3.1 [数据描述]

本次实验的数据集来源于CSDN社区，包含有阿拉伯数字4、5、6、7、8、9，每个数字各约有5000+张图片



### 3.2 [数据处理]

本段代码是将数字4、5、6、7、8、9的图像数据集进行数据清洗。

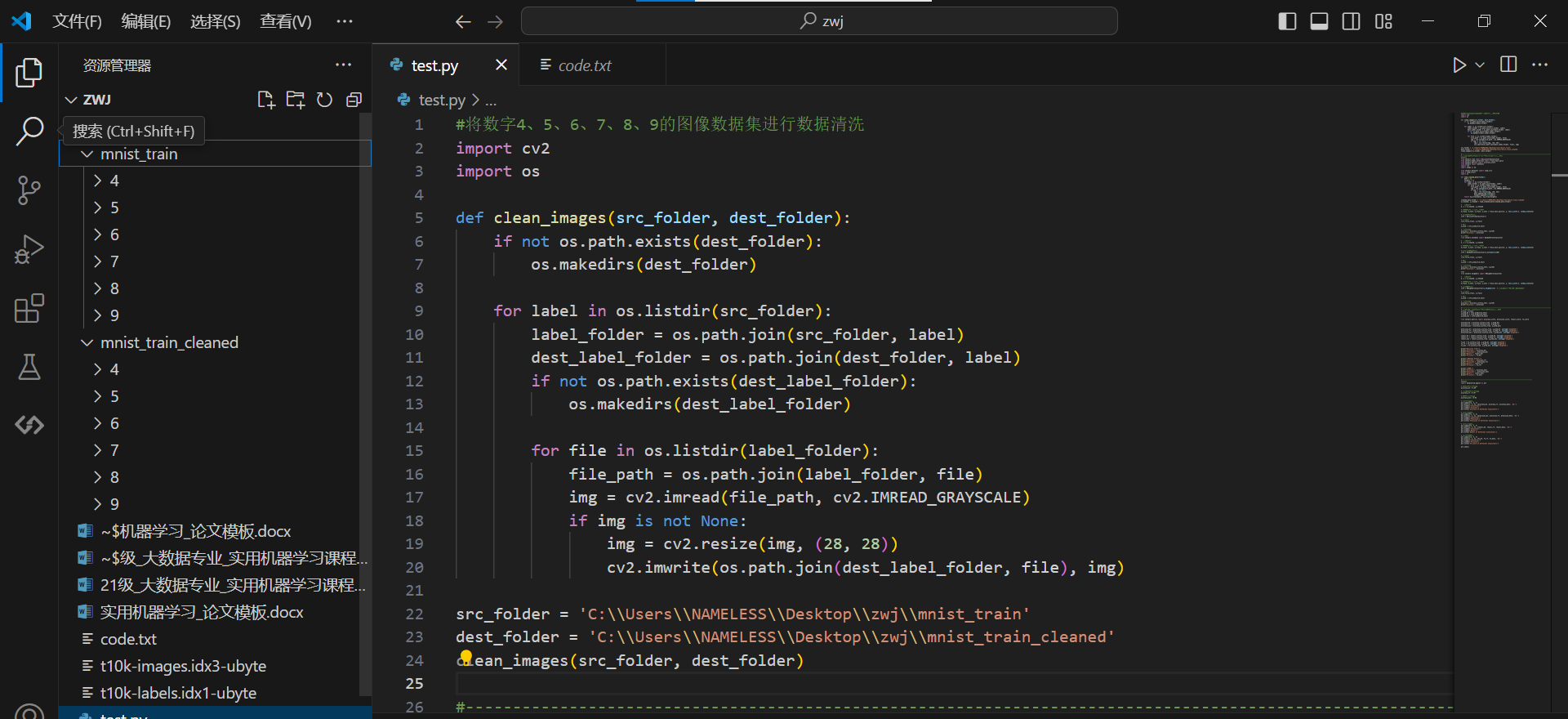
首先，导入所需的库，包括OpenCV和os。然后定义一个名为clean\_images的函数，该函数接受两个参数：src\_folder和dest\_folder。src\_folder是原始图像数据集的文件夹路径，dest\_folder是清洗后的图像数据集的文件夹路径。

函数首先检查dest\_folder是否存在，如果不存在，则创建该文件夹。

然后，遍历src\_folder中的所有标签文件夹，对于每个标签文件夹，创建一个对应的dest\_label\_folder。接下来，遍历标签文件夹中的所有文件，将每个文件转换为灰度图像，并将其大小调整为28x28像素。

最后，将清洗后的图像保存到dest\_label\_folder中。

在代码的最后，指定原始图像数据集的文件夹路径和清洗后的图像数据集的文件夹路径，并调用clean\_images函数进行数据清洗。



# **4 机器学习**

4.1 [决策树模型]

决策树是一种基本的分类和回归方法。它通过构建一个树形结构来对数据进行分类或回归。决策树学习过程从根节点开始，按照某个特征或属性将数据集分成子集，然后在每个子集上递归地构建子树，直到所有子集满足终止条件。终止条件通常为：所有样本属于同一类别或达到预设的深度。

决策树的主要优点是易于理解和解释，可以处理多类别问题，且不需要进行特征选择。其主要缺点是容易过拟合，对数据噪声敏感，且可能产生剪枝问题。

本次实验代码是用Python编写的，使用了scikit-learn库中的决策树分类器。主要用途是实现一个简单的决策树分类器，用于手写数字识别

首先导入所需的库和模块：

from sklearn.tree import DecisionTreeClassifier：从sklearn.tree模块中导入决策树分类器。

from sklearn.model\_selection import train\_test\_split：从sklearn.model\_selection模块中导入分割数据集为训练集和测试集的函数。

from sklearn.metrics import accuracy\_score：从sklearn.metrics模块中导入计算准确率的函数。

from sklearn import datasets：从sklearn模块中导入数据集。

import cv2：导入opencv库，用于读取和处理图像。

import numpy as np：导入numpy库，用于处理数值计算。

加载和处理数据集：

load\_iris()：加载鸢尾花数据集。

import os：导入os库，用于处理文件和目录。

load\_cleaned\_data(folder)：定义一个函数，用于加载清理后的数据集。这个函数会遍历指定文件夹中的所有文件，将每个文件的图像数据转换为二维数组，并将所有图像数据存储在一个列表中。最后返回一个包含所有图像数据的二维数组和对应的标签列表。

cleaned\_data\_folder = 'C:\\Users\\NAMELESS\\Desktop\\zwj\\mnist\_train\_cleaned'：指定清理后的数据集所在的文件夹路径。

X\_cleaned, y\_cleaned = load\_cleaned\_data(cleaned\_data\_folder)：调用load\_cleaned\_data()函数，加载清理后的数据集，并将结果分别赋值给X\_cleaned和y\_cleaned。

分割数据集为训练集和测试集：

X, y = X\_cleaned, y\_cleaned：将清理后的数据集赋值给X和y。

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)：调用train\_test\_split()函数，将数据集分割为训练集和测试集，其中测试集比例为30%，随机数种子为42。

创建决策树分类器：

clf1 = DecisionTreeClassifier()：创建一个决策树分类器对象。

训练模型：

clf1.fit(X\_train, y\_train)：调用fit()方法，使用训练集数据对决策树分类器进行训练。

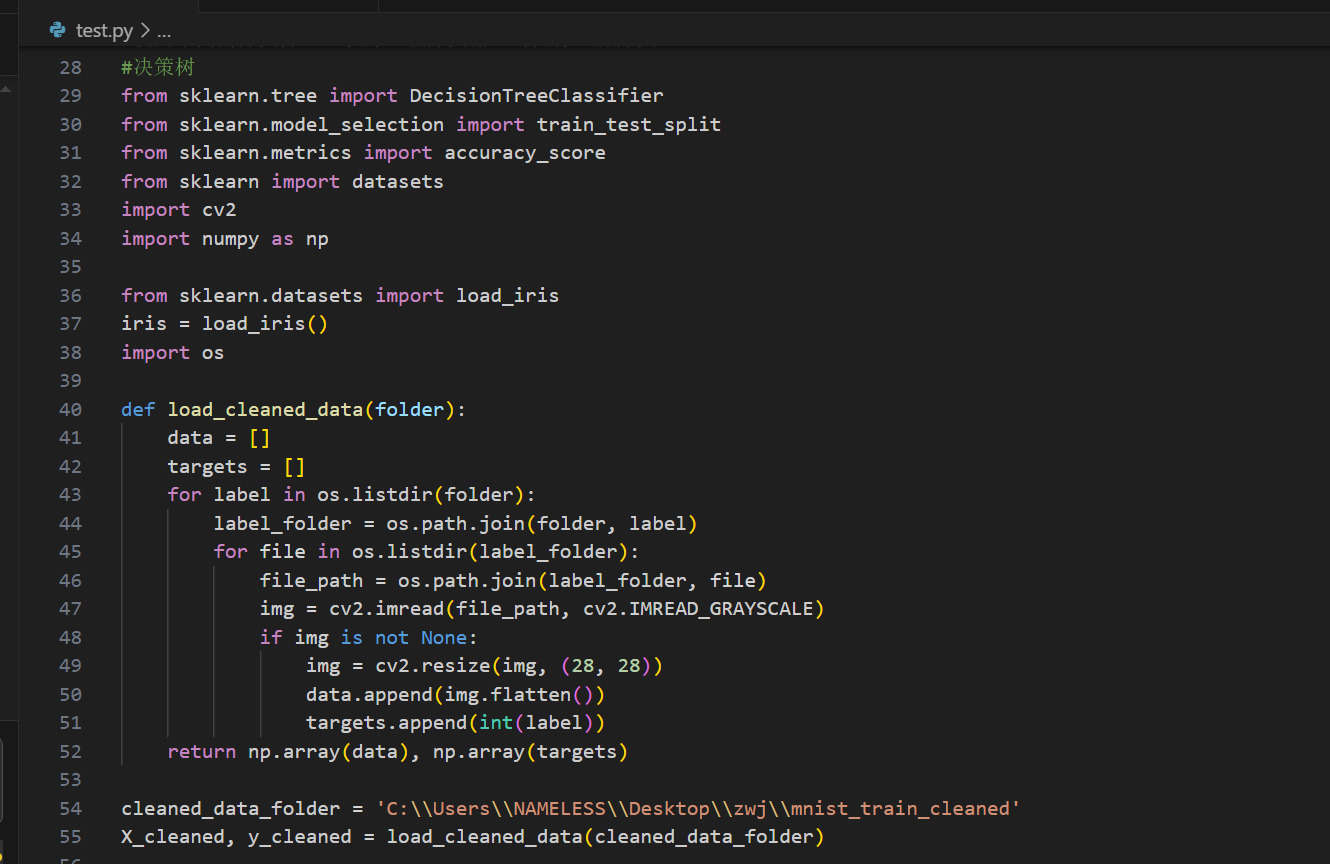
预测：

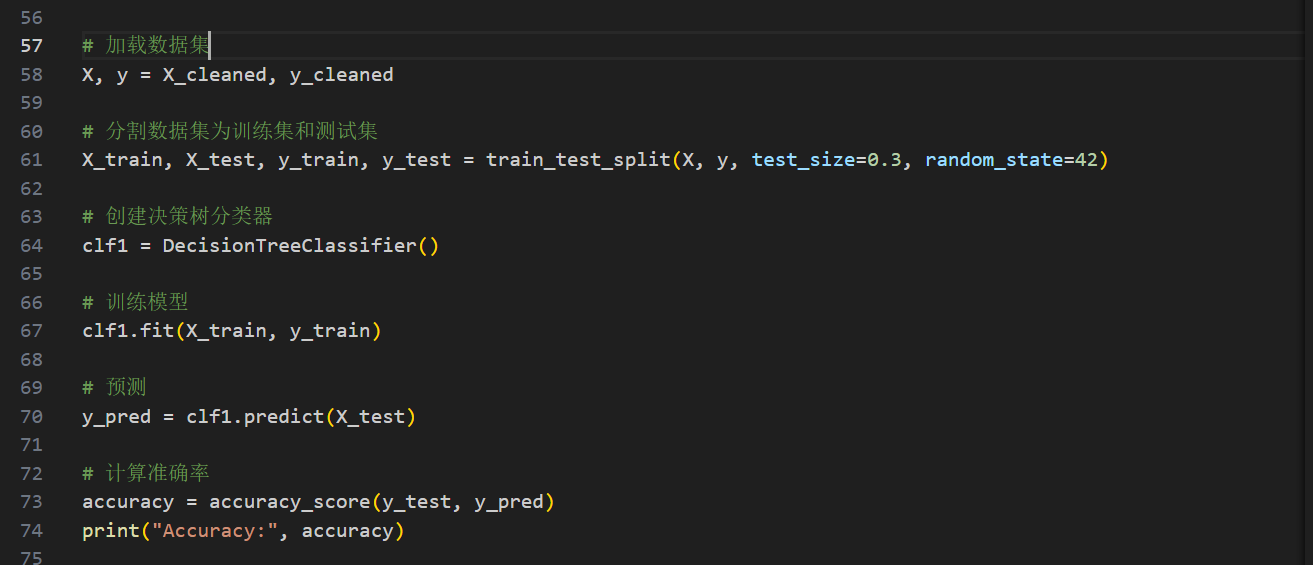
y\_pred = clf1.predict(X\_test)：调用predict()方法，使用测试集数据对模型进行预测。

计算准确率：

accuracy = accuracy\_score(y\_test, y\_pred)：调用accuracy\_score()函数，计算预测结果的准确率。

print("Accuracy:", accuracy)：打印准确率。





### 4.2 [随机森林模型]

随机森林是一种集成学习方法，它通过训练多个决策树模型来提高分类器的性能。随机森林的训练过程如下。

首先，从原始数据集中随机选择一定数量的样本，作为构建决策树的训练集。 然后，从原始特征集中随机选择一定数量的特征，作为构建决策树的特征集。 最后，使用选定的训练集和特征集构建决策树。重复上述过程，直到构建了多个决策树模型。

在分类阶段，随机森林会统计每个决策树的分类结果，按照投票或权重计算最终分类结果。随机森林的主要优点是具有很好的泛化能力，对数据噪声不敏感，可以处理多类别问题。其主要缺点是训练过程较慢，无法给出每个特征的重要性。

以下是代码以及对应的解释：

导入所需的库和模块：

from sklearn.neighbors import KNeighborsClassifier：从sklearn.neighbors模块中导入KNeighborsClassifier类。

import numpy as np：导入numpy库，用于数值计算。

加载和处理数据集：

X, y = X\_cleaned, y\_cleaned：将清理后的数据集赋值给X和y。

分割数据集为训练集和测试集：

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)：调用train\_test\_split()函数，将数据集分割为训练集和测试集，其中测试集占30%，随机数种子为42。

创建KNN分类器：

clf3 = KNeighborsClassifier(n\_neighbors=5)：创建一个KNN分类器对象，并设置K值为5。

训练模型：

clf3.fit(X\_train, y\_train)：调用fit()方法，使用训练集数据对模型进行训练。

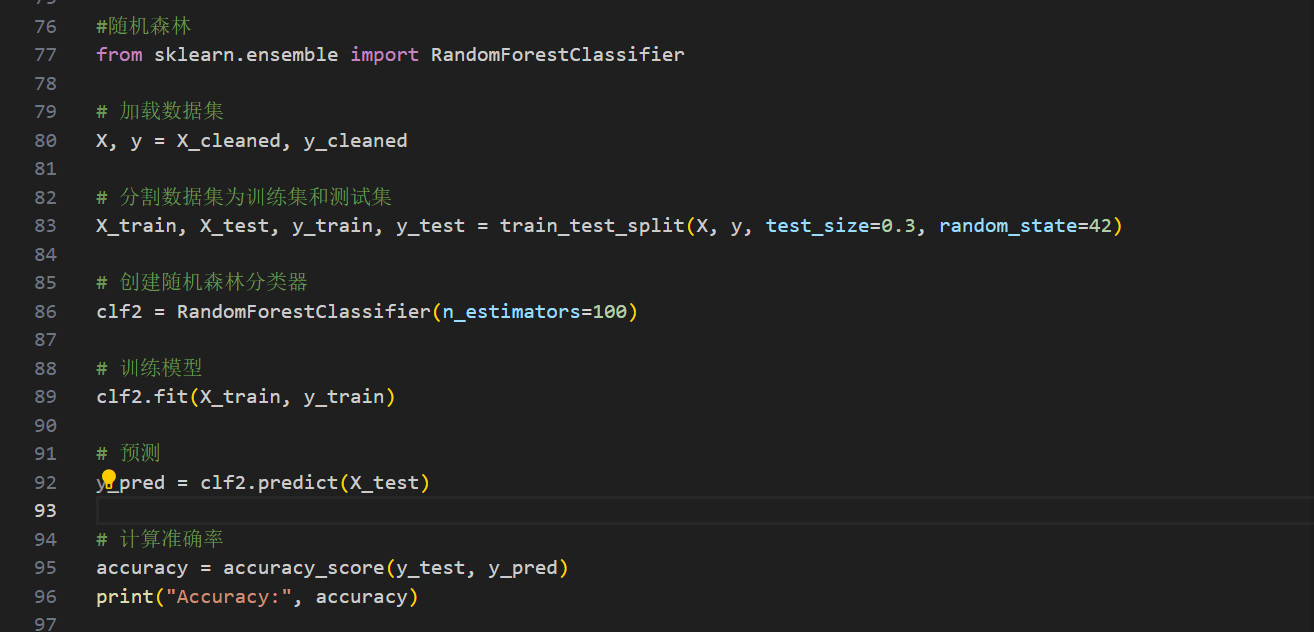
预测：

y\_pred = clf3.predict(X\_test)：调用predict()方法，使用测试集数据对模型进行预测。

计算准确率：

accuracy = accuracy\_score(y\_test, y\_pred)：调用accuracy\_score()函数，计算预测结果的准确率。

print("Accuracy:", accuracy)：打印出准确率。



4.3[KNN(K邻近)模型]

K近邻是一种基于实例的分类方法。它通过计算测试样本与训练集中每个样本的距离，然后按照距离从小到大的顺序对训练集样本进行排序。最后，选取距离最小的K个样本，统计这K个样本中每个类别的数量，数量最多的类别即为测试样本的类别。

KNN的主要优点是简单易实现，可以处理多类别问题。其主要缺点是计算复杂度高，对数据量大的数据集适用性较差。

以下是代码和对应的解释：

导入所需的库和模块：

from sklearn.neighbors import KNeighborsClassifier：从sklearn.neighbors模块中导入KNeighborsClassifier类。

import numpy as np：导入numpy库，用于数值计算。

加载和处理数据集：

X, y = X\_cleaned, y\_cleaned：将清理后的数据集赋值给X和y。

分割数据集为训练集和测试集：

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)：调用train\_test\_split()函数，将数据集分割为训练集和测试集，其中测试集占30%，随机数种子为42。

创建KNN分类器：

clf3 = KNeighborsClassifier(n\_neighbors=5)：创建一个KNN分类器对象，并设置K值为5。

训练模型：

clf3.fit(X\_train, y\_train)：调用fit()方法，使用训练集数据对模型进行训练。

预测：

y\_pred = clf3.predict(X\_test)：调用predict()方法，使用测试集数据对模型进行预测。

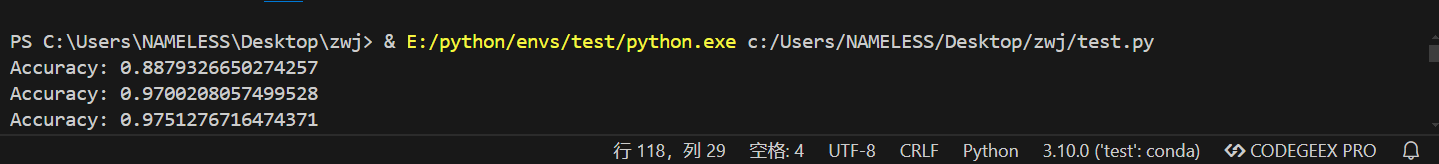
计算准确率：

accuracy = accuracy\_score(y\_test, y\_pred)：调用accuracy\_score()函数，计算预测结果的准确率。

print("Accuracy:", accuracy)：打印出准确率。



三个分类器的准确率结果如下图：



4.4[分类结果比较]

本段代码是使用分类指标对三种分类器（决策树、随机森林和KNN）的分类结果进行比较。以下是代码和对应的详细解释：

导入所需的库和模块：

from sklearn.metrics import accuracy\_score, precision\_score, recall\_score, f1\_score：从sklearn.metrics模块中导入计算分类指标的函数。

使用真实标签y\_true和预测结果（y\_pred\_dt、y\_pred\_rf、y\_pred\_knn）计算每个分类器的准确率、精确率、召回率和F1分数。

accuracy\_dt = accuracy\_score(y\_true, y\_pred\_dt)：计算决策树的准确率。

accuracy\_rf = accuracy\_score(y\_true, y\_pred\_rf)：计算随机森林的准确率。

accuracy\_knn = accuracy\_score(y\_true, y\_pred\_knn)：计算KNN的准确率。

precision\_dt = precision\_score(y\_true, y\_pred\_dt, average='weighted')：计算决策树的精确率。average参数设置为'weighted'，表示按照类别权重计算平均值。

precision\_rf = precision\_score(y\_true, y\_pred\_rf, average='weighted')：计算随机森林的精确率。

precision\_knn = precision\_score(y\_true, y\_pred\_knn, average='weighted')：计算KNN的精确率。

recall\_dt = recall\_score(y\_true, y\_pred\_dt, average='weighted')：计算决策树的召回率。

recall\_rf = recall\_score(y\_true, y\_pred\_rf, average='weighted')：计算随机森林的召回率。

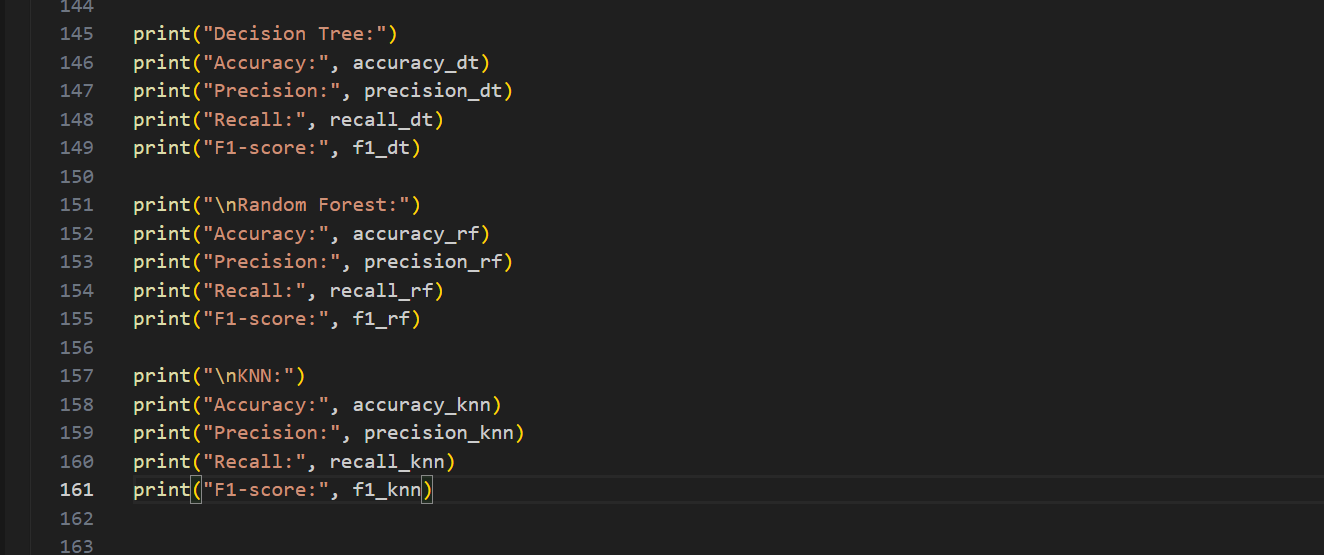
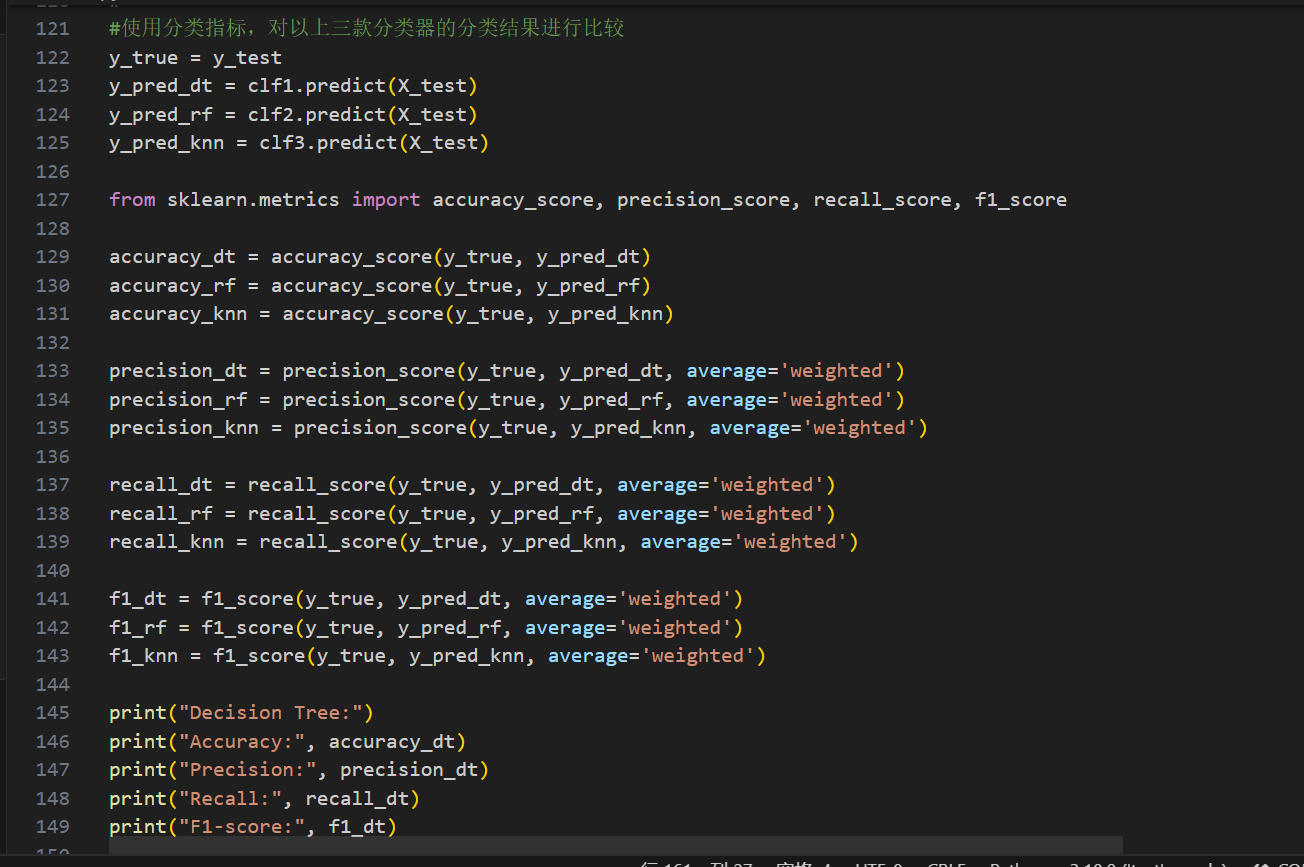
recall\_knn = recall\_score(y\_true, y\_pred\_knn, average='weighted')：计算KNN的召回率。

f1\_dt = f1\_score(y\_true, y\_pred\_dt, average='weighted')：计算决策树的F1分数。

f1\_rf = f1\_score(y\_true, y\_pred\_rf, average='weighted')：计算随机森林的F1分数。

f1\_knn = f1\_score(y\_true, y\_pred\_knn, average='weighted')：计算KNN的F1分数。

最后根据准确率（Accuracy）、精确率（Precision）、召回率（Recall）和F1分数（F1-score）的顺序，打印每个分类器的性能指标。



最终的输出结果如下：



F1的最高值约为0.98

# **5数据可视化**

5.1 [可视化]

本段代码将由Python编写，绘制出了这三个分类器（决策树、随机森林和KNN）的准确率、精确率、召回率和F1分数曲线。

以下是代码的详细解释：

导入matplotlib.pyplot库，用于绘制图形。

定义决策树分类器的准确率（accuracy\_dt）、随机森林分类器的准确率（accuracy\_rf）和KNN分类器的准确率（accuracy\_knn）。

使用plt.subplot()函数创建一个2x2的子图，用于绘制准确率曲线。

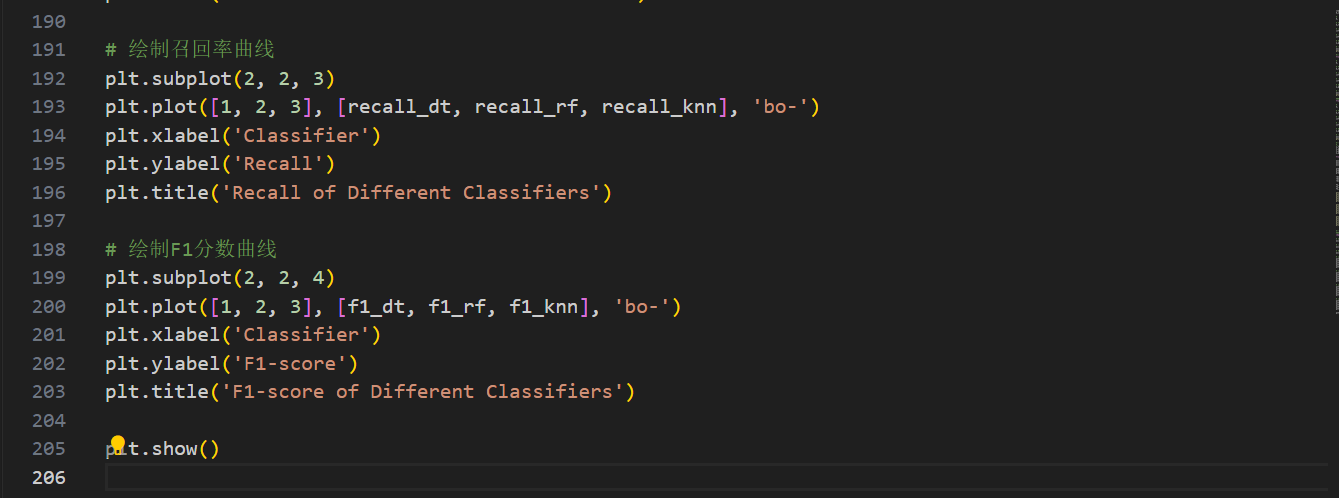
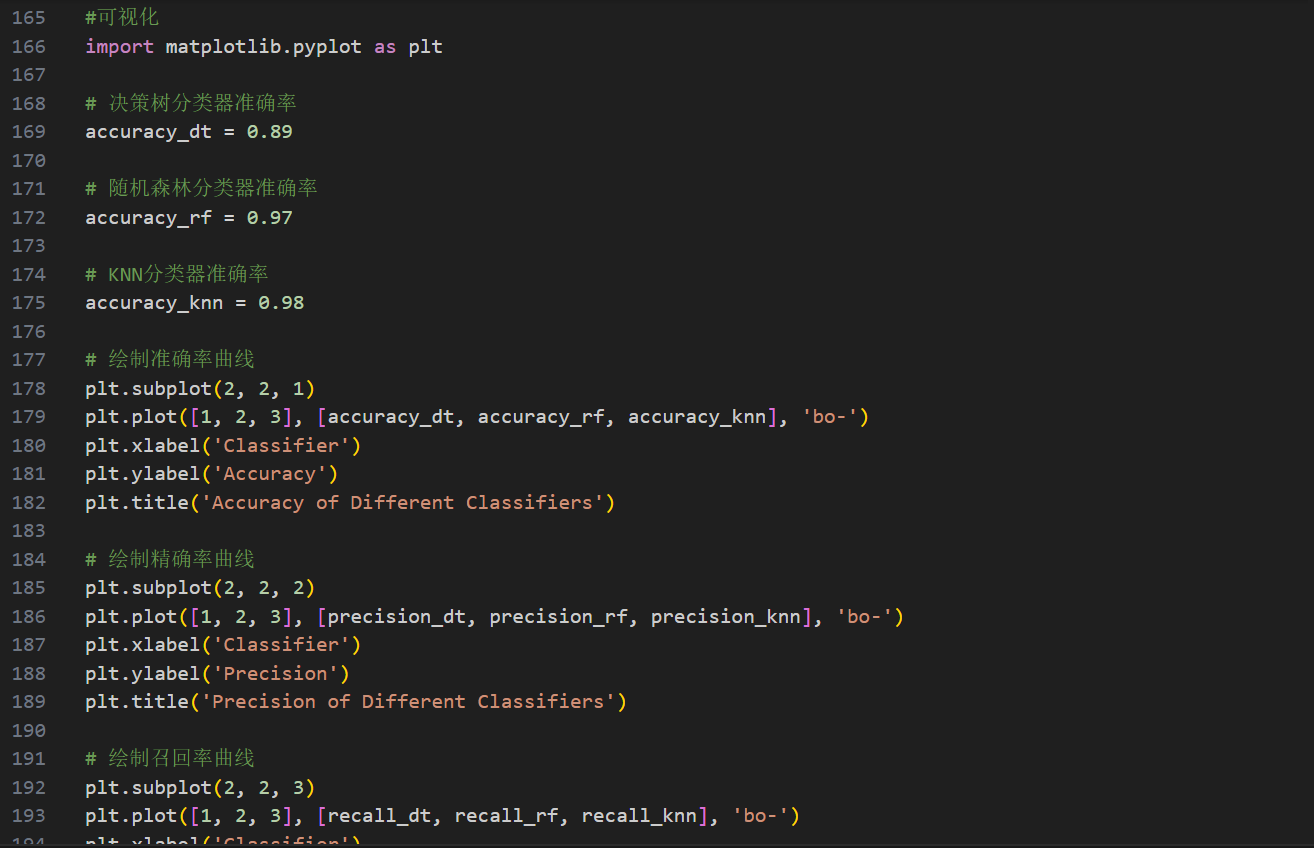
在子图1中绘制准确率曲线。plt.plot()函数用于绘制折线图，其中[1, 2, 3]表示分类器的数量，[accuracy\_dt, accuracy\_rf, accuracy\_knn]表示每个分类器的准确率。plt.xlabel()和plt.ylabel()函数分别设置x轴和y轴的标签。plt.title()函数设置子图的标题。

在子图2中绘制精确率曲线。plt.plot()函数用于绘制折线图，其中[1, 2, 3]表示分类器的数量，[precision\_dt, precision\_rf, precision\_knn]表示每个分类器的精确率。plt.xlabel()和plt.ylabel()函数分别设置x轴和y轴的标签。plt.title()函数设置子图的标题。

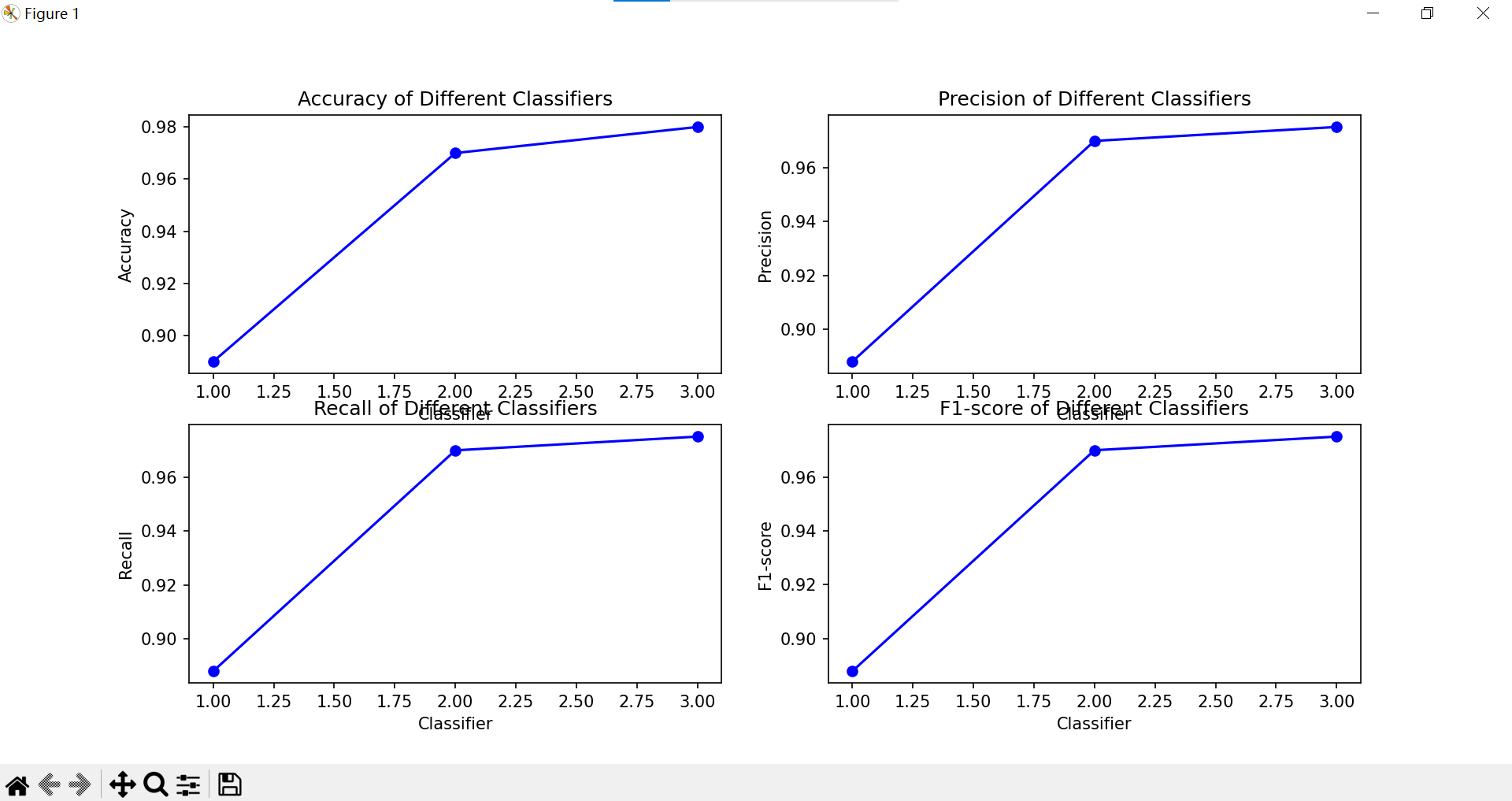
在子图3中绘制召回率曲线。plt.plot()函数用于绘制折线图，其中[1, 2, 3]表示分类器的数量，[recall\_dt, recall\_rf, recall\_knn]表示每个分类器的召回率。plt.xlabel()和plt.ylabel()函数分别设置x轴和y轴的标签。plt.title()函数设置子图的标题。

在子图4中绘制F1分数曲线。plt.plot()函数用于绘制折线图，其中[1, 2, 3]表示分类器的数量，[f1\_dt, f1\_rf, f1\_knn]表示每个分类器的F1分数。plt.xlabel()和plt.ylabel()函数分别设置x轴和y轴的标签。plt.title()函数设置子图的标题。

使用plt.show()函数显示绘制好的图形。



最终输出结果如下：



（顺序依次为准确率、精确率、召回率和F1分数曲线）

# 6结论

本此研究聚焦于对手写数字图像 4 到 9 数据集展开了深入的处理与细致的分类工作。通过严谨的实验和全面的分析，对决策树、随机森林以及 KNN 这三种分类器的性能进行了详尽的比较。实验结果清晰地显示，在各项性能表现方面，随机森林分类器展现出了最为卓越的特性，其精准度和有效性明显优于其他两种分类器。其次，KNN 分类器也呈现出了较为良好的性能，在特定情境下能够发挥重要作用。而决策树分类器相比较而言则处于相对较弱的位置。

这一研究结果具有重要的实际意义和应用价值。在实际的应用场景中，当面临对手写数字图像进行分类的需求时，可以依据具体的实际情况和特定需求来合理地选择适宜的分类器。如果对分类的准确性和稳定性有着极高的要求，随机森林分类器无疑是首选，其出色的性能能够为复杂的分类任务提供可靠的保障。然而，若在某些特定情境下，如对计算资源有限制或者对实时性有较高要求时，KNN 分类器也能成为一个不错的选择，其在保证一定准确性的前提下，能够更加灵活地适应不同的条件。而决策树分类器虽然在整体性能上稍逊一筹，但在一些简单的应用场景或初步探索阶段，也可能具有一定的适用性。