1. 作业说明

自选数据集（例如 iris 数据集）实现决策树算法，并用 Micro-F1 和 Macro-F1 分数进行验证集评估，语言和工具库不限。提交 pdf 格式报告以及可运行代码压缩包，报告内容包括但不限于：

1. 数据的分析与处理；
2. 决策树的设计原理和核心代码；
3. 验证集评估结果（ Micro-F1 和 Macro-F1 截图）；
4. 使用开源库对决策树的可视化结果。

2. 作业要求

提交pdf格式报告以及可运行代码压缩包，报告内容包括但不限于：

- 数据的分析与处理（1`）；

- 决策树的设计原理和核心代码（2`）；

- 验证集评估结果（ Micro-F1和Macro-F1 截图）（1`）；

- 使用开源库对决策树的可视化结果（1`）。

实现决策树改进方案（加分项）（1`）：

- 修剪枝叶；

- 随机森林；

- 其他。

3. 数据的分析和处理

主要是加载数据集并且打印一些数据集数据的分析

3.1. IrisDataLoad.py

```python
from sklearn.datasets import load_iris


class MyIrisDataLoader:

    def __init__(self):
        self.iris = load_iris()
        # Iris数据集包含数据和标签
        self.X = self.iris.data  # 特征数据
        self.y = self.iris.target  # 目标标签

        self.attributes = self.iris.feature_names  # 属性名称
        self.labels = self.iris.target_names  # 标签名称

    def my_load_iris_data(self):
```

```
        return self.X, self.y, self.attributes, self.labels

    def print_iris_data_info(self):
        # 可以查看数据集的描述信息
        print(self.iris.DESCR)

        print(type(self.X))
        print(type(self.y))
        print(self.X.shape)
        print(self.y.shape)

        print(self.labels)
        print(self.attributes)
```

```
:Summary Statistics:

============== ==== ==== ======= ===== ====================
                Min  Max   Mean    SD   Class Correlation
============== ==== ==== ======= ===== ====================
sepal length:   4.3  7.9   5.84   0.83    0.7826
sepal width:    2.0  4.4   3.05   0.43   -0.4194
petal length:   1.0  6.9   3.76   1.76    0.9490  (high!)
petal width:    0.1  2.5   1.20   0.76    0.9565  (high!)
============== ==== ==== ======= ===== ====================
```

```
<class 'numpy.ndarray'>
<class 'numpy.ndarray'>
(150, 4)
(150,)
['setosa' 'versicolor' 'virginica']
['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)']
```

可见Iris数据集是150个X和y

X有4个属性，y则有三个种类，以及一些统计数据如min和sd

3.2. 数据处理

```
# 中值滤波和高斯滤波
X = medfilt(X, kernel_size=3)
X = gaussian_filter(X, sigma=1)
```

数据集划分是7:3

4. 决策树的设计原理和核心代码

4.1. 设计原理

决策树算法是一个递归的过程，它通过将数据集按特征分裂成更小的子集来构建一个树状的结构。

4.1.1. 特征选择

决策树的特征选择是构建树的关键步骤之一。常用方法包括信息增益和基尼不纯度。信息增益衡量使用特征划分数据后的信息不确定性减少程度，基尼不纯度衡量随机选择一个样本被错误分类的概率。选择具有最大信息增益或最小基尼不纯度的特征进行分裂。

4.1.2. 结点分裂

一旦选定特征，决策树根据该特征将数据集分割成子集。对于离散特征，可能根据每个取值创建不同的分支；对于连续特征，可能通过阈值划分数据。这个过程是递归的，不断重复选择最佳特征、分割数据集，直至满足停止条件。

4.1.3. 剪枝

决策树容易过拟合训练数据，因此需要剪枝来避免过度拟合。预剪枝在树的构建过程中设置停止条件，如限制树的深度或节点中最少样本数；后剪枝则在构建完整树后通过剪枝策略修剪不必要的节点。

4.1.4. 树的评估

评估决策树性能的指标包括准确性、信息增益、基尼不纯度等。通过测试数据集验证模型的准确性，评估分裂效果和节点纯度提升情况。交叉验证等方法可评估模型的稳定性和泛化能力。

4.2. 核心代码：DTree.py

```
from matplotlib import pyplot as plt
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report
import IrisDataLoad
import numpy as np
import pandas as pd
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import f1_score
from sklearn.tree import export_text
from sklearn.tree import export_graphviz
import graphviz
from sklearn.metrics import precision_score
from sklearn import tree

if __name__ == '__main__':
    choice = "DecisionTree"
    # choice = "RandomForest"

    # 1.加载数据集
```

```
    myIrisDataLoader = IrisDataLoad.MyIrisDataLoader()  # 需要加上()，否则会报错
    X, y, attributes, labels = myIrisDataLoader.my_load_iris_data()
    print(X)  # 150行4列
    print(y)  # 150行1列
    print(attributes)  # 四种，分别是sepal length (cm), sepal width (cm), petal length (cm), petal width (cm)
    print(labels)  # 三种，分别是setosa, versicolor, virginica

    # 打印数据集信息
    # myIrisDataLoader.print_iris_data_info()

    # 2.划分数据集
    # 分割数据集为训练集和验证集
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=35)

    # 打印X_train和y_train的形状
    print("X_train and y_train shape:")
    print(X_train.shape)
    print(y_train.shape)

    # 3.训练模型
    # 创建决策树分类器
    clf = None
    if choice == "DecisionTree":
        clf = DecisionTreeClassifier()
    elif choice == "RandomForest":
        clf = RandomForestClassifier(n_estimators=10)
    else:
        print("Wrong choice or no choice set!")
        exit(0)

    # 训练模型
    clf.fit(X_train, y_train)

    # 在验证集上进行预测
    y_pred = clf.predict(X_test)

    # 4.评估模型
    # 计算F1-score
    # macro和micro的含义是
    # macro：对每个类别的f1-score进行算术平均
    # micro：对每个类别的f1-score进行算术平均
    f1_macro = f1_score(y_test, y_pred, average='macro')
    f1_micro = f1_score(y_test, y_pred, average='micro')
    print("F1-score(macro):", f1_macro)
    print("F1-score(micro):", f1_micro)

    classfication_report = classification_report(y_test, y_pred, target_names=labels)
    print("Classification report:")
    print(classfication_report)

    # 准确度计算precision
    # 计算决策树模型的精确度
```

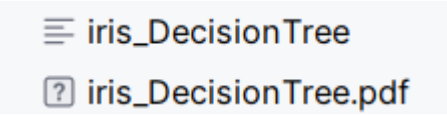
```
    precision = precision_score(y_test, y_pred, average='micro')

    # 5.可视化
    if choice == "DecisionTree":
        # 一、使用tree.plot_tree()方法可视化决策树
        # plt.figure(dpi=200)
        # tree.plot_tree(clf, feature_names=attributes, class_names=labels)
        # plt.show()

        # 二、使用export_graphviz()方法可视化决策树
        dot_data = export_graphviz(
            clf,
            out_file=None,
            feature_names=attributes,
            class_names=labels,
            filled=True,
            rounded=True,
            special_characters=True
        )
        graph = graphviz.Source(dot_data)
        graph.render("iris" + "_" + choice)
    elif choice == "RandomForest":
        dot_data_rf = export_graphviz(
            clf.estimators_[0],
            out_file=None,
            feature_names=attributes,
            class_names=labels,
            filled=True,
            rounded=True,
            special_characters=True
        )
        graph_rf = graphviz.Source(dot_data_rf)
        graph_rf.render("iris" + "_" + choice)
```

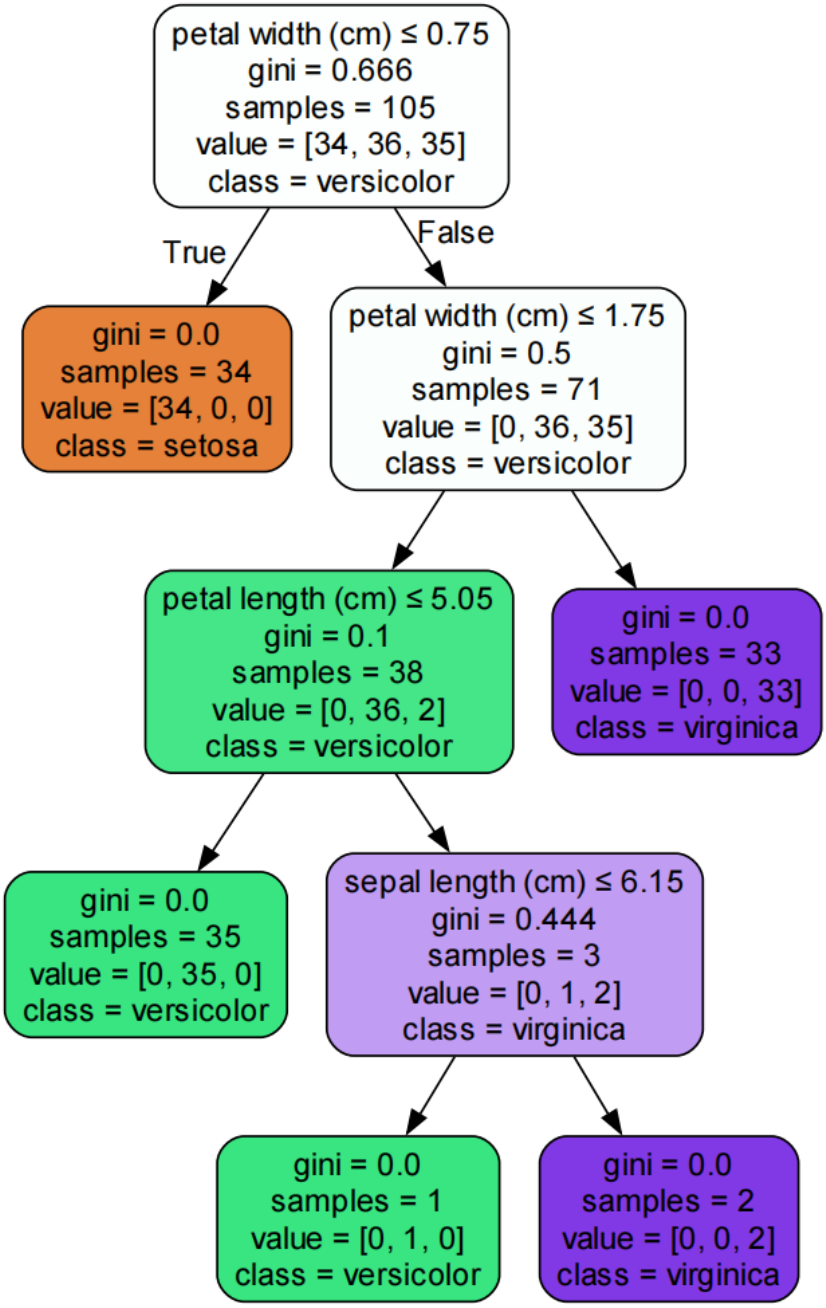
5. 结果展示

```
F1-score(macro): 0.9079365079365079
F1-score(micro): 0.9111111111111111
Classification report:
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        16
  versicolor       0.81      0.93      0.87        14
   virginica       0.92      0.80      0.86        15

    accuracy                           0.91        45
   macro avg       0.91      0.91      0.91        45
weighted avg       0.92      0.91      0.91        45
```

6. 可视化

调用sklearn.tree中的export_graphviz函数



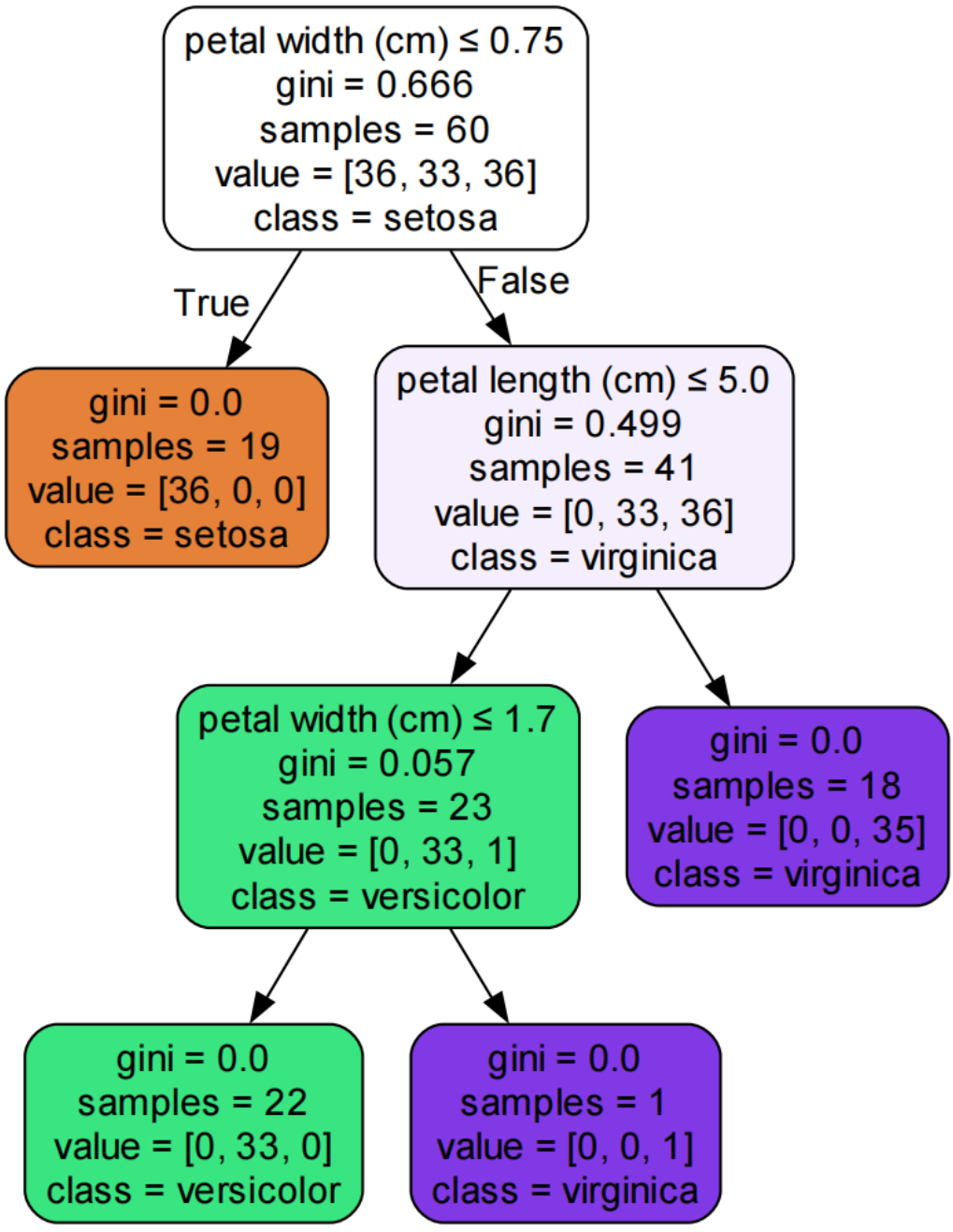


7. 实现决策树改进方案（加分项）

对比了随机森林的结果

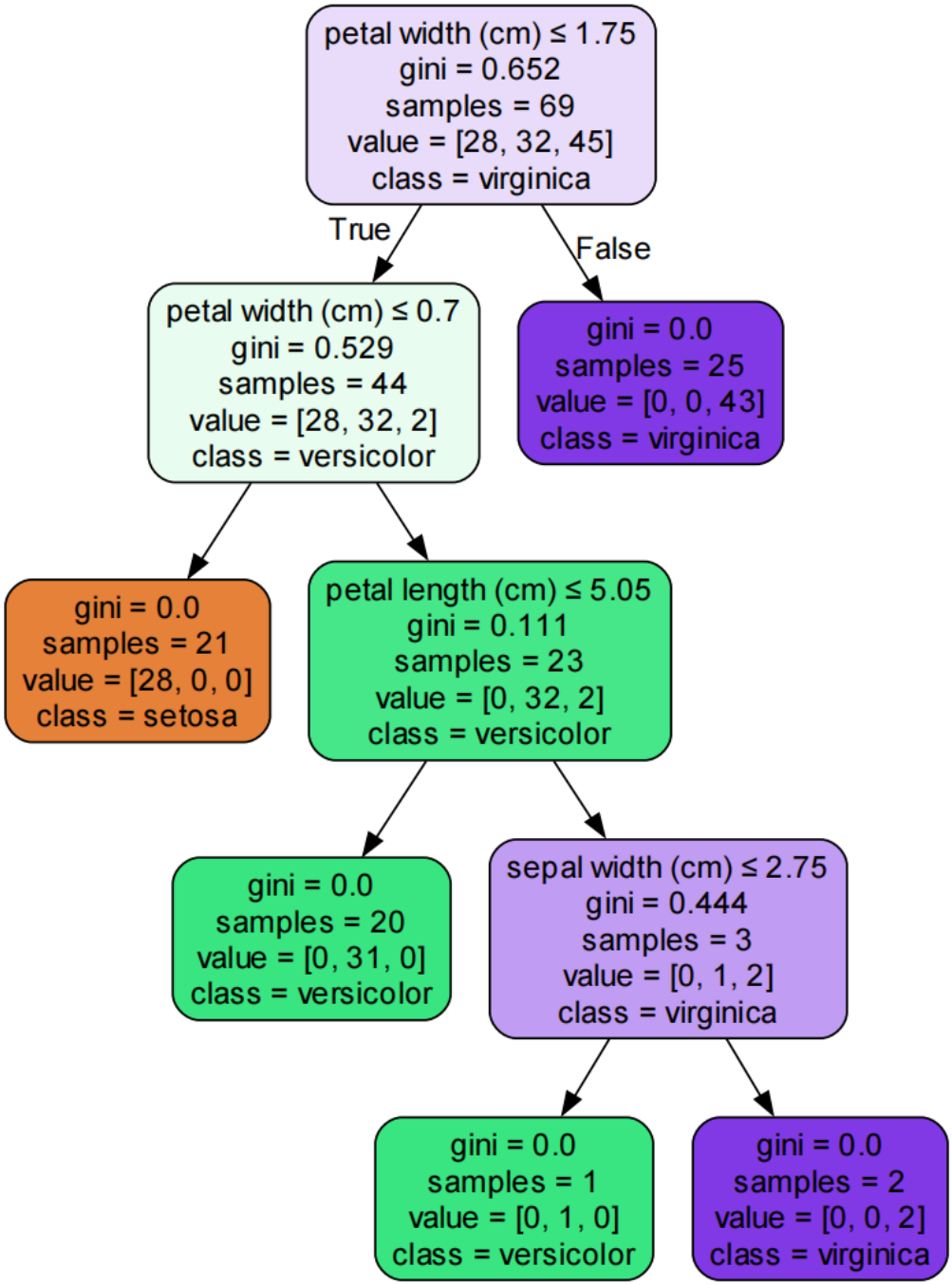
```
n_estimators=2下的结果
```

```
F1-score(macro): 0.860576923076923
F1-score(micro): 0.8666666666666667
Classification report:
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        16
  versicolor       0.72      0.93      0.81        14
   virginica       0.91      0.67      0.77        15

    accuracy                           0.87        45
   macro avg       0.88      0.87      0.86        45
weighted avg       0.88      0.87      0.86        45
```



```
n_estimators=4下的结果
```

```
F1-score(macro): 0.9079365079365079
F1-score(micro): 0.9111111111111111
Classification report:
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        16
  versicolor       0.81      0.93      0.87        14
   virginica       0.92      0.80      0.86        15

    accuracy                           0.91        45
   macro avg       0.91      0.91      0.91        45
weighted avg       0.92      0.91      0.91        45
```


petal width (cm) ≤ 1.75
gini = 0.652
samples = 69
value = [28, 32, 45]
class = virginica
True
False
petal width (cm) ≤ 0.7
gini = 0.529
samples = 44
value = [28, 32, 2]
class = versicolor
gini = 0.0
samples = 25
value = [0, 0, 43]
class = virginica
gini = 0.0
samples = 21
value = [28, 0, 0]
class = setosa
petal length (cm) ≤ 5.05
gini = 0.111
samples = 23
value = [0, 32, 2]
class = versicolor
gini = 0.0
samples = 20
value = [0, 31, 0]
class = versicolor
sepal width (cm) ≤ 2.75
gini = 0.444
samples = 3
value = [0, 1, 2]
class = virginica
gini = 0.0
samples = 1
value = [0, 1, 0]
class = versicolor
gini = 0.0
samples = 2
value = [0, 0, 2]
class = virginica

增加预处理后

```
F1-score(macro): 0.9777530589543938
F1-score(micro): 0.9777777777777777
Classification report:
              precision    recall  f1-score   support

      setosa       1.00      0.94      0.97        16
  versicolor       0.93      1.00      0.97        14
   virginica       1.00      1.00      1.00        15

    accuracy                           0.98        45
   macro avg       0.98      0.98      0.98        45
weighted avg       0.98      0.98      0.98        45
```

可见有所提升