实验五：鸢尾花分类（决策树）

一、实验介绍

1.1 实验内容

决策树是机器学习中一种简单而又经典的算法。本次实验将带领了解决策树的基本原理，并学习使用 scikit-learn 来构建一个决策树分类模型，最后使用此模型预测鸢尾花的种类。

1.2 实验知识点

决策树的基本原理。

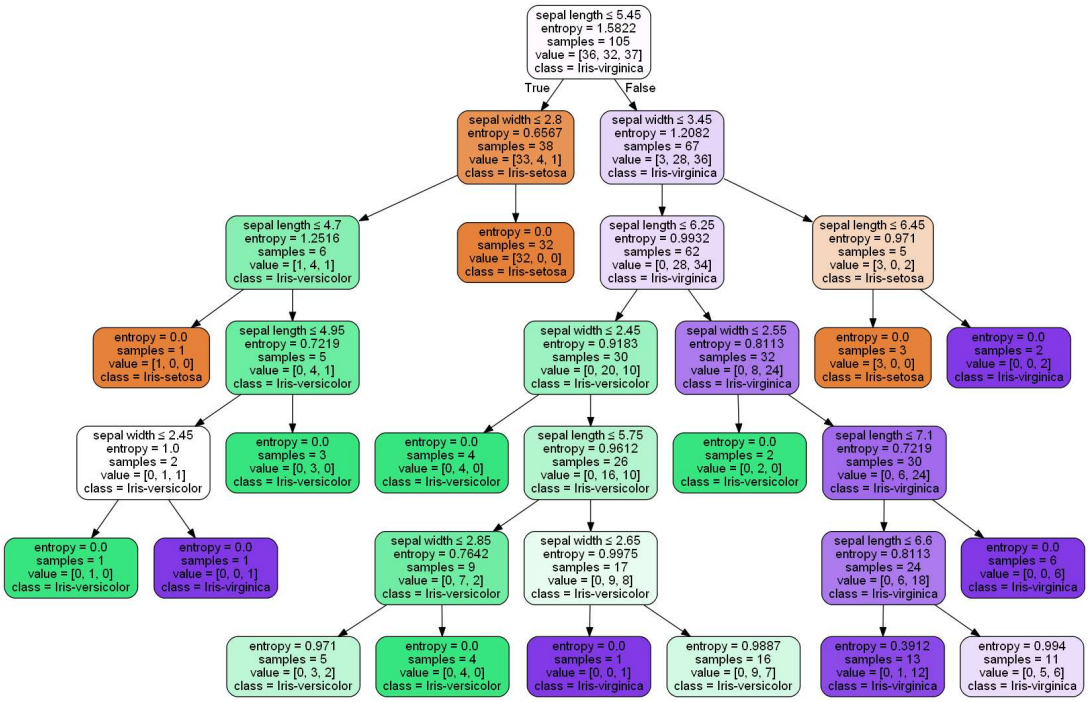
决策树在生成和修剪中使用的 ID3, C4.5 及 CART 算法。

使用 scikit-learn 中提供的决策树分类器进行实例验证。

二、决策树基本原理

2.1 决策树简介

决策树是一种特殊的树形结构，一般由节点和有向边组成。其中，节点表示特征、属性或者一个类。而有向边包含有判断条件。如图所示，决策树从根节点开始延伸，经过不同的判断条件后，到达不同的子节点。而上层子节点又可以作为父节点被进一步划分为下层子节点。一般情况下，从根节点输入数据，经过多次判断后，这些数据就会被分为不同的类别。这就构成了一颗简单的分类决策树。



2.2 决策树学习

将决策数的思想引入到机器学习中，就产生了一种简单而又经典的预测方法——决策树学习（Decision Tree Learning），亦简称为决策树。决策树可以用来解决分类或回归问题，分别称之为分类树或回归树。其中，分类树的输出是一个标量，而回归树的一般输出为一个实数。

通常情况下，决策树利用损失函数最小的原则建立模型，然后再利用该模型进行预测。决策树学习通常包含三个阶段：特征选择、树的生成，树的修剪。

三、鸢尾花分类实验

构建一个决策树分类模型，实现对鸢尾花分类。

3.1 数据集简介

鸢尾花数据集是机器学习领域一个非常经典的分类数据集。接下来，我们就用这个训练集为基础，一步一步地训练一个机器学习模型。首先，我们来看一下该数据集的基本构成。数据集名称的准确名称为 Iris Data Set，总共包含 150 行数据。每一行数据由 4 个特征值及一个目标值组成。其中 4 个特征值分别为：萼片长度、萼片宽度、花瓣长度、花瓣宽度。而目标值及为三种不同类别的鸢尾花，分别为：Iris Setosa，Iris Versicolour，Iris Virginica。

3.2 数据获取及划分

通过著名的 UCI 机器学习数据集网站下载该数据集。本实验中，为了更加

便捷地实验。我们直接实验 scikit-learn 提供的方法导入该数据集即可。打开实验环境右下角的菜单 > 附件 > ipython，依次键入代码。

```
#基于决策树的鸢尾花分类
'''
- 程序开发环境：win10 64 位
- Python 版本：64 位 3.7
- 依赖库：numpy、pandas、sklearn、matplotlib
- 程序输入：iris.cvs
- 程序输出：iris_classification_result.xlsx
# -*- coding: utf-8 -*-
'''
```

接下来，可以直接通过 print iris_target 查看一下花的分类数据。这里，scikit-learn 已经将花的原名称进行了转换，其中 0，1，2 分别代表 Iris Setosa，Iris Versicolour 和 Iris Virginica。

这些数据是按照鸢尾花类别的顺序排列的。所以，如果将其直接划分为训练集和数据集的话，就会造成数据的分布不均。详细来讲，直接划分容易造成某种类型的花在训练集中一次都未出现，训练的模型就永远不可能预测出这种花来。你可能会想到，将这些数据打乱后再划分训练集和数据集。当然，更方便地，scikit-learn 为我们提供了训练集和数据集的方法。

其中，feature_train，feature_test，target_train，target_test 分别代表训练集特征、测试集特征、训练集目标值、验证集特征。test_size 参数代表划分到测试集数据占全部数据的百分比，你也可以用 train_size 来指定训练集所占全部数据的百分比。一般情况下，我们会将整个训练集划分为 70% 训练集和 30% 测试集。最后的 random_state 参数表示乱序程度。

导入模块

```
import pandas as pdimport matplotlib.pyplot as pltfrom sklearn.datasets import load_irisfrom sklearn import tree
```

读取数据

```
iris = load_iris()

iris_feature = iris.data #特征数据

iris_target = iris.target #分类数据



#鸢尾花数组长度为 150，共 3 种类别。

#种类为：Iris Setosa（山鸢花）、Iris Versicolour（杂色鸢尾）Iris Virginica（维吉尼亚鸢尾），分别用 0、1、2 标签代表

print (iris.data)          #输出数据集

print ('------------------------------')

print (iris.target)        #输出真实标签

print (len(iris.target) )

print ('------------------------------')

print (iris.data.shape )   #150 个样本  每个样本 4 个特征

print ('------------------------------')
```

打印返回值：

```
[[5.1 3.5 1.4 0.2]

 [4.9 3.  1.4 0.2]

 [4.7 3.2 1.3 0.2]

 [4.6 3.1 1.5 0.2]

 [5.  3.6 1.4 0.2]

 [5.4 3.9 1.7 0.4]

 [4.6 3.4 1.4 0.3]

 [5.  3.4 1.5 0.2]

 [4.4 2.9 1.4 0.2]

 [4.9 3.1 1.5 0.1]

 [5.4 3.7 1.5 0.2]
```

```
[4.8 3.4 1.6 0.2]

[4.8 3.  1.4 0.1]

[4.3 3.  1.1 0.1]

[5.8 4.  1.2 0.2]

[5.7 4.4 1.5 0.4]

[5.4 3.9 1.3 0.4]

[5.1 3.5 1.4 0.3]

[5.7 3.8 1.7 0.3]

[5.1 3.8 1.5 0.3]

[5.4 3.4 1.7 0.2]

[5.1 3.7 1.5 0.4]

[4.6 3.6 1.  0.2]

[5.1 3.3 1.7 0.5]

[4.8 3.4 1.9 0.2]

[5.  3.  1.6 0.2]

[5.  3.4 1.6 0.4]

[5.2 3.5 1.5 0.2]

[5.2 3.4 1.4 0.2]

[4.7 3.2 1.6 0.2]

[4.8 3.1 1.6 0.2]

[5.4 3.4 1.5 0.4]

[5.2 4.1 1.5 0.1]

[5.5 4.2 1.4 0.2]

[4.9 3.1 1.5 0.2]

[5.  3.2 1.2 0.2]

[5.5 3.5 1.3 0.2]

[4.9 3.6 1.4 0.1]
```

```
[4.4 3.  1.3 0.2]

[5.1 3.4 1.5 0.2]

[5.  3.5 1.3 0.3]

[4.5 2.3 1.3 0.3]

[4.4 3.2 1.3 0.2]

[5.  3.5 1.6 0.6]

[5.1 3.8 1.9 0.4]

[4.8 3.  1.4 0.3]

[5.1 3.8 1.6 0.2]

[4.6 3.2 1.4 0.2]

[5.3 3.7 1.5 0.2]

[5.  3.3 1.4 0.2]

[7.  3.2 4.7 1.4]

[6.4 3.2 4.5 1.5]

[6.9 3.1 4.9 1.5]

[5.5 2.3 4.  1.3]

[6.5 2.8 4.6 1.5]

[5.7 2.8 4.5 1.3]

[6.3 3.3 4.7 1.6]

[4.9 2.4 3.3 1. ]

[6.6 2.9 4.6 1.3]

[5.2 2.7 3.9 1.4]

[5.  2.  3.5 1. ]

[5.9 3.  4.2 1.5]

[6.  2.2 4.  1. ]

[6.1 2.9 4.7 1.4]

[5.6 2.9 3.6 1.3]
```

```
[6.7 3.1 4.4 1.4]

[5.6 3.  4.5 1.5]

[5.8 2.7 4.1 1. ]

[6.2 2.2 4.5 1.5]

[5.6 2.5 3.9 1.1]

[5.9 3.2 4.8 1.8]

[6.1 2.8 4.  1.3]

[6.3 2.5 4.9 1.5]

[6.1 2.8 4.7 1.2]

[6.4 2.9 4.3 1.3]

[6.6 3.  4.4 1.4]

[6.8 2.8 4.8 1.4]

[6.7 3.  5.  1.7]

[6.  2.9 4.5 1.5]

[5.7 2.6 3.5 1. ]

[5.5 2.4 3.8 1.1]

[5.5 2.4 3.7 1. ]

[5.8 2.7 3.9 1.2]

[6.  2.7 5.1 1.6]

[5.4 3.  4.5 1.5]

[6.  3.4 4.5 1.6]

[6.7 3.1 4.7 1.5]

[6.3 2.3 4.4 1.3]

[5.6 3.  4.1 1.3]

[5.5 2.5 4.  1.3]

[5.5 2.6 4.4 1.2]

[6.1 3.  4.6 1.4]
```

```
[5.8 2.6 4.  1.2]

[5.  2.3 3.3 1. ]

[5.6 2.7 4.2 1.3]

[5.7 3.  4.2 1.2]

[5.7 2.9 4.2 1.3]

[6.2 2.9 4.3 1.3]

[5.1 2.5 3.  1.1]

[5.7 2.8 4.1 1.3]

[6.3 3.3 6.  2.5]

[5.8 2.7 5.1 1.9]

[7.1 3.  5.9 2.1]

[6.3 2.9 5.6 1.8]

[6.5 3.  5.8 2.2]

[7.6 3.  6.6 2.1]

[4.9 2.5 4.5 1.7]

[7.3 2.9 6.3 1.8]

[6.7 2.5 5.8 1.8]

[7.2 3.6 6.1 2.5]

[6.5 3.2 5.1 2. ]

[6.4 2.7 5.3 1.9]

[6.8 3.  5.5 2.1]

[5.7 2.5 5.  2. ]

[5.8 2.8 5.1 2.4]

[6.4 3.2 5.3 2.3]

[6.5 3.  5.5 1.8]

[7.7 3.8 6.7 2.2]

[7.7 2.6 6.9 2.3]
```

```
[6.  2.2 5.  1.5]

[6.9 3.2 5.7 2.3]

[5.6 2.8 4.9 2. ]

[7.7 2.8 6.7 2. ]

[6.3 2.7 4.9 1.8]

[6.7 3.3 5.7 2.1]

[7.2 3.2 6.  1.8]

[6.2 2.8 4.8 1.8]

[6.1 3.  4.9 1.8]

[6.4 2.8 5.6 2.1]

[7.2 3.  5.8 1.6]

[7.4 2.8 6.1 1.9]

[7.9 3.8 6.4 2. ]

[6.4 2.8 5.6 2.2]

[6.3 2.8 5.1 1.5]

[6.1 2.6 5.6 1.4]

[7.7 3.  6.1 2.3]

[6.3 3.4 5.6 2.4]

[6.4 3.1 5.5 1.8]

[6.  3.  4.8 1.8]

[6.9 3.1 5.4 2.1]

[6.7 3.1 5.6 2.4]

[6.9 3.1 5.1 2.3]

[5.8 2.7 5.1 1.9]

[6.8 3.2 5.9 2.3]

[6.7 3.3 5.7 2.5]

[6.7 3.  5.2 2.3]
```

```
 [6.3 2.5 5.  1.9]

 [6.5 3.  5.2 2. ]

 [6.2 3.4 5.4 2.3]

 [5.9 3.  5.1 1.8]]

-----------------------------

[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2

 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2

 2 2]

150

-----------------------------

(150, 4)

-----------------------------
```

导入数据集

```
#导入数据集 irisurl = "https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"names = ['sepal-length', 'sepal-width', 'petal-length', 'petal-width', 'class']dataset = pd.read_csv(url, names=names) #读取 csv 数据 print(dataset.describe())print('------------------------------')
```

打印返回值:

```
        sepal-length  sepal-width  petal-length  petal-width

count    150.000000   150.000000    150.000000   150.000000

mean       5.843333     3.054000      3.758667     1.198667

std        0.828066     0.433594      1.764420     0.763161
```

```
min         4.300000      2.000000      1.000000      0.100000
25%         5.100000      2.800000      1.600000      0.300000
50%         5.800000      3.000000      4.350000      1.300000
75%         6.400000      3.300000      5.100000      1.800000
max         7.900000      4.400000      6.900000      2.500000
```

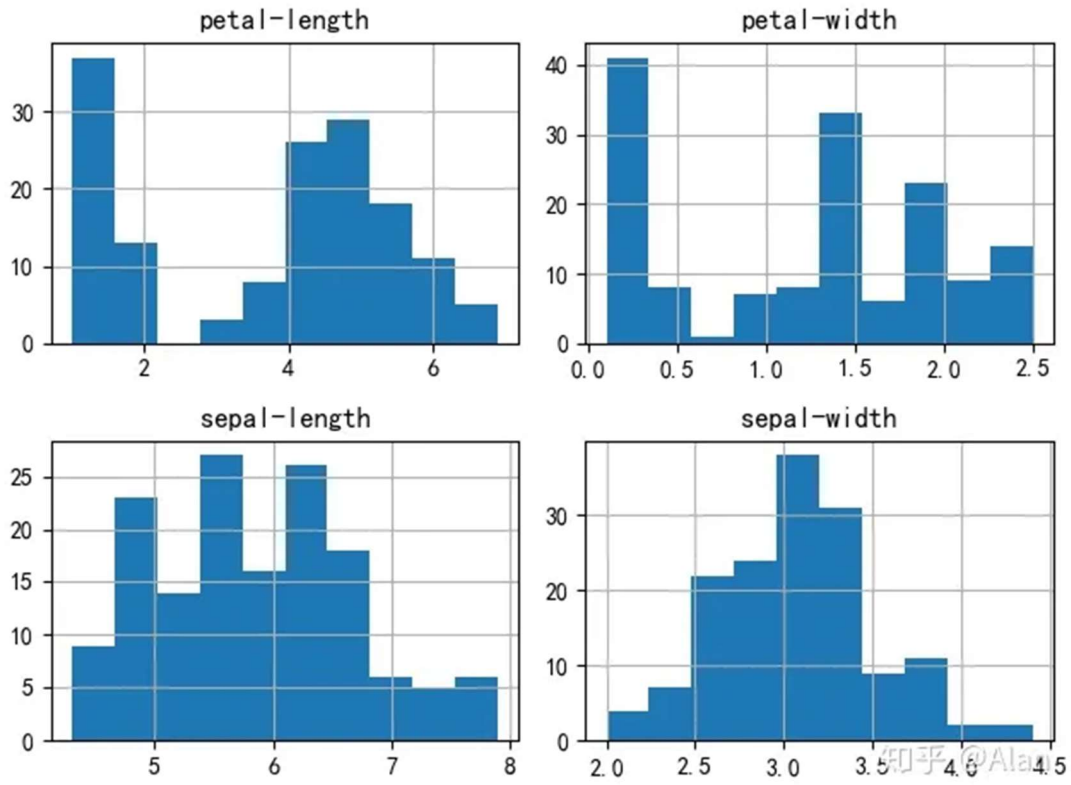
数据可视化

#4 种特征维度分布情况

```
#直方图 histograms

dataset.hist()

plt.show()
```



3.2 模型训练及预测

首先是从 scikit-learn 中导入决策树分类器。然后实验 fit 方法和 predict 方法对模型进行训练和预测。

DecisionTreeClassifier() 模型方法中也包含非常多的参数值。例如：

criterion = gini/entropy 可以用来选择用基尼指数或者熵来做损失函数。

splitter＝best/random 用来确定每个节点的分裂策略。支持“最佳”或者“随机”。

max_depth = int 用来控制决策树的最大深度，防止模型出现过拟合。

min_samples_leaf = int 用来设置叶节点上的最少样本数量，用于对树进行修剪。

可以将预测结果和测试集的真实值分别输出，对照比较。

训练分类

```
from sklearn.tree import DecisionTreeClassifier

clf = DecisionTreeClassifier()      # 所以参数均置为默认状态

clf.fit(iris.data, iris.target)     # 使用训练集训练模型

print(clf)

print ('------------------------------')

predicted = clf.predict(iris.data)

print(predicted)

print ('------------------------------')
```

打印返回值

```
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,

                       max_features=None, max_leaf_nodes=None,

                       min_impurity_decrease=0.0, min_impurity_split=None,

                       min_samples_leaf=1, min_samples_split=2,

                       min_weight_fraction_leaf=0.0, presort=False,

                       random_state=None, splitter='best')
```

```
-----------------------------
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2]
```

获取花卉两列数据集

```
X = iris.data
L1 = [x[0] for x in X]
print(L1)
print ('-----------------------------')
L2 = [x[1] for x in X]
print (L2)
print ('-----------------------------')
```

打印返回值:

```
[5.1, 4.9, 4.7, 4.6, 5.0, 5.4, 4.6, 5.0, 4.4, 4.9, 5.4, 4.8, 4.8, 4.3, 5.8, 5.7, 5.4,
 5.1, 5.7, 5.1, 5.4, 5.1, 4.6, 5.1, 4.8, 5.0, 5.0, 5.2, 5.2, 4.7, 4.8, 5.4, 5.2, 5.5,
 4.9, 5.0, 5.5, 4.9, 4.4, 5.1, 5.0, 4.5, 4.4, 5.0, 5.1, 4.8, 5.1, 4.6, 5.3, 5.0, 7.0,
 6.4, 6.9, 5.5, 6.5, 5.7, 6.3, 4.9, 6.6, 5.2, 5.0, 5.9, 6.0, 6.1, 5.6, 6.7, 5.6, 5.8,
6.2, 5.6, 5.9, 6.1, 6.3, 6.1, 6.4, 6.6, 6.8, 6.7, 6.0, 5.7, 5.5, 5.5, 5.8, 6.0, 5.4,
6.0, 6.7, 6.3, 5.6, 5.5, 5.5, 6.1, 5.8, 5.0, 5.6, 5.7, 5.7, 6.2, 5.1, 5.7, 6.3, 5.8,
 7.1, 6.3, 6.5, 7.6, 4.9, 7.3, 6.7, 7.2, 6.5, 6.4, 6.8, 5.7, 5.8, 6.4, 6.5, 7.7, 7.7,
6.0, 6.9, 5.6, 7.7, 6.3, 6.7, 7.2, 6.2, 6.1, 6.4, 7.2, 7.4, 7.9, 6.4, 6.3, 6.1, 7.7,
6.3, 6.4, 6.0, 6.9, 6.7, 6.9, 5.8, 6.8, 6.7, 6.7, 6.3, 6.5, 6.2, 5.9]
-----------------------------
```

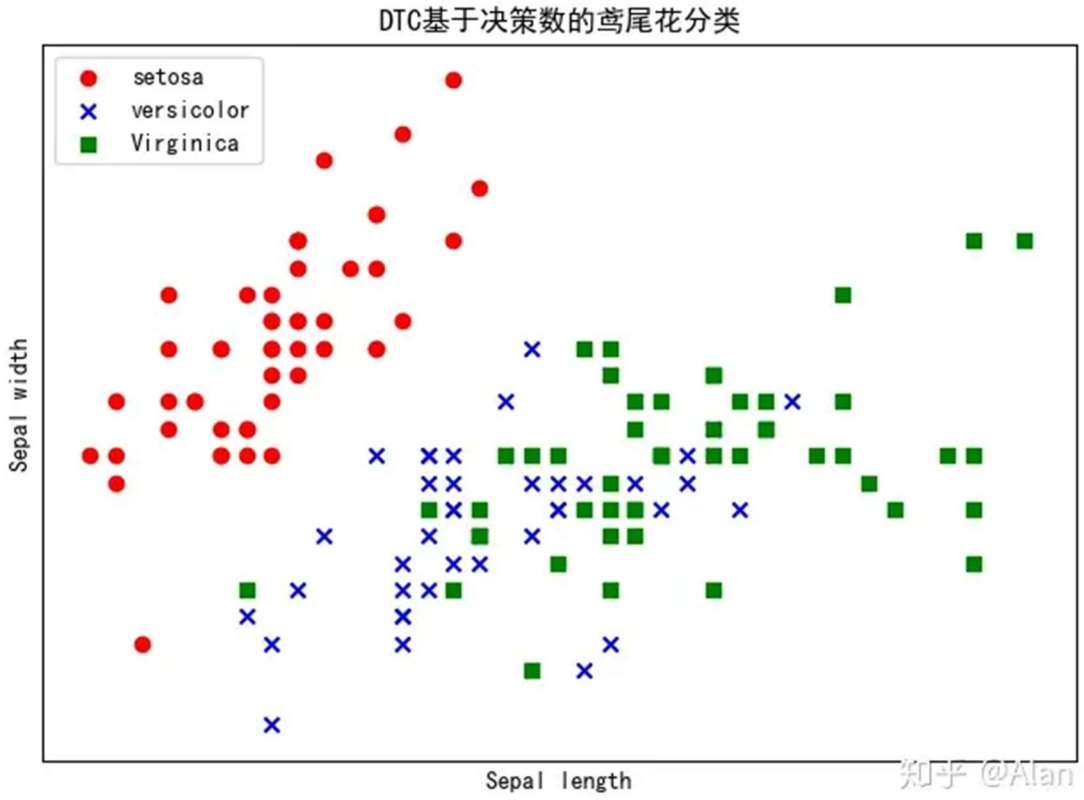
```
[3.5, 3.0, 3.2, 3.1, 3.6, 3.9, 3.4, 3.4, 2.9, 3.1, 3.7, 3.4, 3.0, 3.0, 4.0, 4.4, 3.9,
3.5, 3.8, 3.8, 3.4, 3.7, 3.6, 3.3, 3.4, 3.0, 3.4, 3.5, 3.4, 3.2, 3.1, 3.4, 4.1, 4.2,
3.1, 3.2, 3.5, 3.6, 3.0, 3.4, 3.5, 2.3, 3.2, 3.5, 3.8, 3.0, 3.8, 3.2, 3.7, 3.3, 3.2,
 3.2, 3.1, 2.3, 2.8, 2.8, 3.3, 2.4, 2.9, 2.7, 2.0, 3.0, 2.2, 2.9, 2.9, 3.1, 3.0, 2.7,
2.2, 2.5, 3.2, 2.8, 2.5, 2.8, 2.9, 3.0, 2.8, 3.0, 2.9, 2.6, 2.4, 2.4, 2.7, 2.7, 3.0,
 3.4, 3.1, 2.3, 3.0, 2.5, 2.6, 3.0, 2.6, 2.3, 2.7, 3.0, 2.9, 2.9, 2.5, 2.8, 3.3, 2.7,
3.0, 2.9, 3.0, 3.0, 2.5, 2.9, 2.5, 3.6, 3.2, 2.7, 3.0, 2.5, 2.8, 3.2, 3.0, 3.8, 2.6,
 2.2, 3.2, 2.8, 2.8, 2.7, 3.3, 3.2, 2.8, 3.0, 2.8, 3.0, 2.8, 3.8, 2.8, 2.8, 2.6, 3.0,
 3.4, 3.1, 3.0, 3.1, 3.1, 3.1, 2.7, 3.2, 3.3, 3.0, 2.5, 3.0, 3.4, 3.0]
----------------------------
```

绘图

```
plt.scatter(X[:50, 0], X[:50, 1], color='red', marker='o', label='setosa')
plt.scatter(X[50:100, 0], X[50:100, 1], color='blue', marker='x', label='versicolor')
plt.scatter(X[100:, 0], X[100:, 1], color='green', marker='s', label='Virginica')


#中文乱码解决
plt.rcParams['font.sans-serif']=['SimHei']
plt.rcParams['axes.unicode_minus']=False


plt.title("DTC 基于决策数的鸢尾花分类")#标题
plt.xlabel('Sepal length')
plt.ylabel('Sepal width')
plt.xticks(())
plt.yticks(())
plt.legend(loc=2)
plt.show()
```



四、课后习题

尝试通过修改 DecisionTreeClassifier() 方法里面的值，查看模型参数对实验结果带来的影响。

尝试载入 scikit-learn 中提供的另一个著名的 digits 数据集，同样实验决策树分类器实现手写字体识别实验。