重庆科技学院学生实验报告

课程名称	机器学习		实验项目名称	决策树	
开课学院及实验室	计算机科学与工程学院 I316			实验日期	2025 年 10 月 23 日
学生姓名	刘志贵	学号	2023440221	专业班级	智科 2023-02
指导教师	易军		实验成绩		

一、实验目的和要求

1. 理解决策树的基本原理。
2. 理解决策树生成和修剪中使用的 ID3, C4.5 及 CART 算法。
3. 用 scikit-learn 库构建决策树分类模型。
4. 用构建好的模型对鸢尾花（Iris）的种类进行预测。

二、实验内容

这次实验用的经典鸢尾花数据集（Iris Data Set），数据集包含了 150 条数据，每条数据包含了 4 个特征（萼片长度、萼片宽度、花瓣长度、花瓣宽度）和 1 个目标值（花的类别：Iris Setosa, Iris Versicolour, Iris Virginica）。

实验将完成以下任务：

1. 数据的读取与查看。
2. 数据的可视化分析。
3. 构建决策树分类器（DecisionTreeClassifier）。
4. 模型的训练与预测。
5. 结果可视化的展示。

三、实验步骤与结果

1. 先导入需要的模块

导入 pandas，matplotlib，sklearn 这些实验需要的模块：

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
from sklearn import tree
import numpy as np

%matplotlib inline
```

[1] ✓ 2.7s

图 1 导入模块

2. 读取数据

然后使用 scikit-learn 的方式读取数据，也就是 load_iris 方法加载数据并且查看结构，鸢尾花有 150 组数据 3 个类别，分别是 Iris Setosa（山鸢花）、Iris Versicolour（杂色鸢尾）、Iris Virginica（维吉尼亚鸢尾），下面用 0，1，2 标签表示：

```
iris = load_iris()
iris_feature = iris.data
iris_target = iris.target

print(iris.data)
print('-----------------------------')
print(iris.target)
print(len(iris.target))
print('-----------------------------')
print(iris.data.shape)
✓ 0.0s
```

图 2 加载数据

```
 [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)
```

图 3 鸢尾花数据与维度

3. 用 Pandas 读取数据并进行可视化

把上面加载的 iris 数据直接转换为了 Pandas DataFrame 进行可视化。

```
df = pd.DataFrame(iris.data, columns=['sepal-length', 'sepal-width', 'petal-length', 'petal-width'])
df['class'] = iris.target

print(df.describe())
print('-----------------------------')

# 直方图 histograms
df.iloc[:, 0:4].hist()
plt.show()
✓ 0.3s
```

图 4 iris 转换以及可视化代码

```
       sepal-length  sepal-width  petal-length  petal-width       class
count    150.000000   150.000000    150.000000   150.000000  150.000000
mean       5.843333     3.057333      3.758000     1.199333    1.000000
std        0.828066     0.435866      1.765298     0.762238    0.819232
min        4.300000     2.000000      1.000000     0.100000    0.000000
25%        5.100000     2.800000      1.600000     0.300000    0.000000
50%        5.800000     3.000000      4.350000     1.300000    1.000000
75%        6.400000     3.300000      5.100000     1.800000    2.000000
max        7.900000     4.400000      6.900000     2.500000    2.000000
----------------------------
```

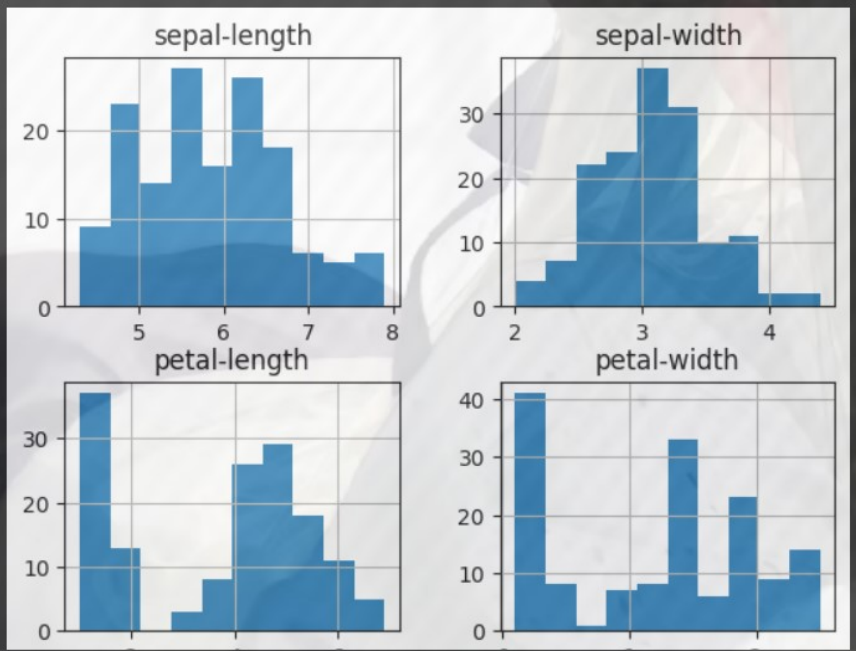

图 5 打印数据描述与可视化直方图

4. 模型训练及预测

导入决策树分类器，先初始化决策树分类器 ，使用的默认参数，然后训练模型，最后预测模型，用的原数据进行预测演示：

```python
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score

clf = DecisionTreeClassifier()

clf.fit(iris.data, iris.target)

print("模型信息:")
print(clf)
print('------------------------------')

predicted = clf.predict(iris.data)
print("预测结果:")
print(predicted)
print('------------------------------')

acc = accuracy_score(iris.target, predicted)
print(f"默认参数模型在训练集上的准确率: {acc:.4f}")
```

0.0s

图 6 模型训练与预测代码

```
模型信息:
DecisionTreeClassifier()
----------------------------
预测结果:
[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 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
 2 2]
----------------------------
默认参数模型在训练集上的准确率: 1.0000
```

图 7 模型信息与预测结果

5. 修改模型参数带来的影响

这里先修改一下参数看一下影响如何，修改的 max_depth（最大深度）和 criterion（划分标准，如 'entropy' 信息熵）来观察模型变化。

```python
print("修改模型参数")
clf_tuned = DecisionTreeClassifier(max_depth=2, criterion='entropy')
clf_tuned.fit(iris.data, iris.target)

accuracy = clf_tuned.score(iris.data, iris.target)
print(f"训练集准确率: {accuracy:.2f}")
print(clf_tuned)
```

✓ 0.0s

```
修改模型参数
训练集准确率: 0.96
DecisionTreeClassifier(criterion='entropy', max_depth=2)
```

图 8 修改模型参数并打印结果

6. 手写字体识别 (Digits 数据集)

然后是第二个课后小题，加载 scikit-learn 提供的 digits 数据集并且用决策树分类：

```python
print("手写数字识别 (Digits)")
from sklearn.datasets import load_digits
from sklearn.model_selection import train_test_split

digits = load_digits()
print(f"Digits 数据集: {digits.data.shape}")

X_train, X_test, y_train, y_test = train_test_split(digits.data, digits.target, test_size=0.3, random_state=42)

digits_clf = DecisionTreeClassifier()
digits_clf.fit(X_train, y_train)

digits_score = digits_clf.score(X_test, y_test)
print(f"手写数字识别测试集准确率: {digits_score:.2f}")

print(f"前5个样本预测: {digits_clf.predict(X_test[:5])}")
print(f"前5个样本真实: {y_test[:5]}")
```

✓ 0.0s

```
手写数字识别 (Digits)
Digits 数据集: (1797, 64)
手写数字识别测试集准确率: 0.84
前5个样本预测: [6 9 3 7 2]
前5个样本真实: [6 9 3 7 2]
```

图 9 Digits 数据集训练并预测

7. 分类可视化结果

这里主要展示数据的提取和基于两个特征（萼片长度、萼片宽度）的分类可视化结果。

```
X = iris.data

L1 = X[:, 0].tolist()
L2 = X[:, 1].tolist()

print("第一列数据:", L1[:])
print("第二列数据:", L2[:])

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

plt.rcParams['font.sans-serif']=['SimHei']
plt.rcParams['axes.unicode_minus']=False

plt.title("DTC 基于决策树的鸢尾花分类")
plt.xlabel('Sepal length')
plt.ylabel('Sepal width')
plt.legend(loc='upper right')
plt.show()
✓ 0.1s
```

图 10 分类可视化代码

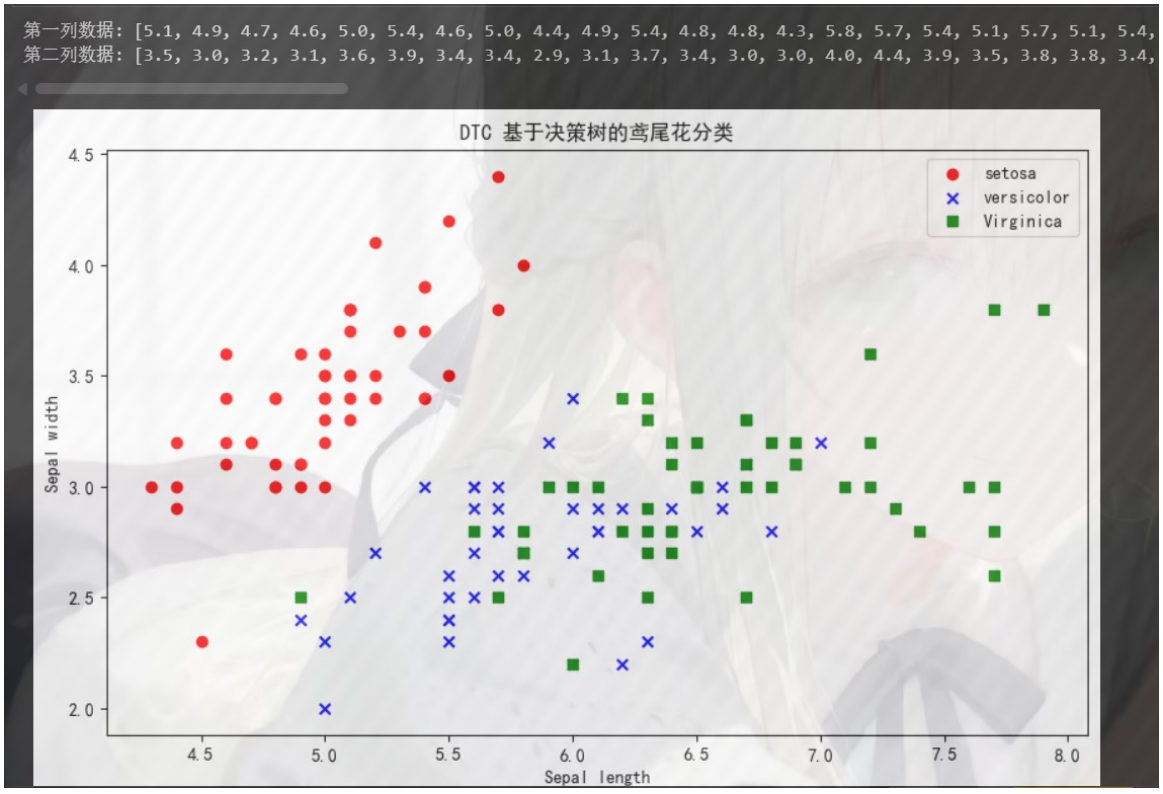

图 11 分类可视化图像

第一列数据:

[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]

第二列数据:

[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]

四、实验结果

默认模型的准确率为 1.0，树深度较深，明显过拟合了。修改参数后（max_depth=2）准确率下降到 0.96，树深度强制为 2。所以设置 max_depth 就限制了树的复杂度（剪枝）。虽然降低了训练集上的准确率，但有更好的泛化能力。

在 digits 数据集上，决策树模型的测试集准确率为 84% 。逻辑回归等线性模型在这个数据集上可以达到 95%以上的准确率，之前的实验数据是这样。所以决策树能够处理高维数据（64 维像素特征）并且完成多分类任务（0-9）。但对于像图像像素这种特征之间存在较强线性关系或平滑过渡的数据，单一的决策树效果可能不是很好。

五、模型评价

图 11 看到红色标记的 Setosa 类别和其他两类在“萼片长度”和“萼片宽度”这两个维度上完全分离，线性可分性很好。蓝色标记的 Versicolor 和绿色标记的 Virginica 在图中存有部分重叠，这说明就靠这两个特征简单的线性划分可能产生误判。决策树通过引入更多特征（花瓣长度、宽度）和更深的分裂层级就可以解决问题，实现精分数据集。

在鸢尾花数据集上，默认的决策树有 100% 的训练准确率，证明了它的拟合能力。但需要注意控制树的深度来平衡偏差与方差，获得更好的泛化性能。

附件

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
from sklearn import tree
import numpy as np
```

```
%matplotlib inline

iris = load_iris()
iris_feature = iris.data
iris_target = iris.target

print(iris.data)
print('------------------------------')
print(iris.target)
print(len(iris.target))
print('------------------------------')
print(iris.data.shape)

df = pd.DataFrame(iris.data, columns=['sepal-length', 'sepal-width', 'petal-length', 'petal-width'])
df['class'] = iris.target

print(df.describe())
print('------------------------------')

# 直方图 histograms
df.iloc[:, 0:4].hist()
plt.show()

from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score

clf = DecisionTreeClassifier()

clf.fit(iris.data, iris.target)

print("模型信息:")
print(clf)
print('------------------------------')

predicted = clf.predict(iris.data)
print("预测结果:")
```

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

acc = accuracy_score(iris.target, predicted)
print(f"默认参数模型在训练集上的准确率: {acc:.4f}")

print("修改模型参数")
clf_tuned = DecisionTreeClassifier(max_depth=2, criterion='entropy')
clf_tuned.fit(iris.data, iris.target)

accuracy = clf_tuned.score(iris.data, iris.target)
print(f"训练集准确率: {accuracy:.2f}")
print(clf_tuned)

print("手写数字识别  (Digits)")
from sklearn.datasets import load_digits
from sklearn.model_selection import train_test_split

digits = load_digits()
print(f"Digits  数据集: {digits.data.shape}")

X_train, X_test, y_train, y_test = train_test_split(digits.data, digits.target, test_size=0.3,
random_state=42)

digits_clf = DecisionTreeClassifier()
digits_clf.fit(X_train, y_train)

digits_score = digits_clf.score(X_test, y_test)
print(f"手写数字识别测试集准确率: {digits_score:.2f}")

print(f"前 5 个样本预测: {digits_clf.predict(X_test[:5])}")
print(f"前 5 个样本真实: {y_test[:5]}")

X = iris.data

L1 = X[:, 0].tolist()
L2 = X[:, 1].tolist()
```

```
print("第一列数据:", L1[:])
print("第二列数据:", L2[:])

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

plt.rcParams['font.sans-serif']=['SimHei']
plt.rcParams['axes.unicode_minus']=False

plt.title("DTC 基于决策树的鸢尾花分类")
plt.xlabel('Sepal length')
plt.ylabel('Sepal width')
plt.legend(loc='upper right')
plt.show()
```