决策树

刘德华

2023 年 6 月 24 日

例子 1

使用决策树分类模型进行分类[a]。

[a]实现的程序见例1

例子 2

使用决策树分类模型对 iris 数据集进行分类，并绘制树状图[a]。

[a]实现的程序见例1

例子 3

使用决策树回归模型进行回归[a]。

[a]实现的程序见例1

例子 4

基于代价复杂度的决策树剪枝[a]。

[a]实现的程序见例1

1 代码

Python 代码 1

```python
# 导入操作系统库
import os
# 更改工作目录
os.chdir(r"D:\softwares\applied statistics\pythoncodelearning\chap5\sourcecode")
# 导入基础计算库
import numpy as np
# 导入绘图库
import matplotlib.pyplot as plt
# 导入数据生成工具
from sklearn.datasets import load_iris
# 导入决策树分类器
from sklearn.tree import DecisionTreeClassifier
# 导入决策边界显示工具
from sklearn.inspection import DecisionBoundaryDisplay
# 导入绘图库中的字体管理包
```

```
from matplotlib import font_manager
# 实现中文字符正常显示
font = font_manager.FontProperties(fname=r"C:\Windows\Fonts\SimKai.ttf")
# 使用 seaborn 风格绘图
plt.style.use("seaborn-v0_8")
# 获取数据
iris = load_iris()
# 类别数
n_classes = 3
# 绘图颜色
plot_colors = "ryb"
# 步长
plot_step = 0.02
fig, axs = plt.subplots(2, 3, figsize=(6,6), tight_layout=True)
for pairidx, pair in enumerate([[0, 1], [0, 2], [0, 3], [1, 2], [1, 3], [2, 3]]):
    # 两个维度
    X = iris.data[:, pair]
    y = iris.target
    # 决策树建模
    clf = DecisionTreeClassifier()
    # 模型拟合
    clf.fit(X, y)
    # 绘制决策边界
    ax = axs.flatten()[pairidx]
    DecisionBoundaryDisplay.from_estimator(
        clf,
        X,
        cmap=plt.cm.RdYlBu,
        response_method="predict",
        ax=ax,
        xlabel=iris.feature_names[pair[0]],
        ylabel=iris.feature_names[pair[1]],
    )
    # 训练样本散点图
    for i, color in zip(range(n_classes), plot_colors):
        idx = np.where(y == i)
        ax.scatter(
            X[idx, 0],
            X[idx, 1],
            c=color,
            label=iris.target_names[i],
            edgecolor="black",
            s=15,
```

```
        )

plt.suptitle("Decision surface of decision trees trained on pairs of features")
ax.legend(loc="lower right", borderpad=0, handletextpad=0)
plt.show()
fig.savefig("../codeimage/code1.pdf")
```

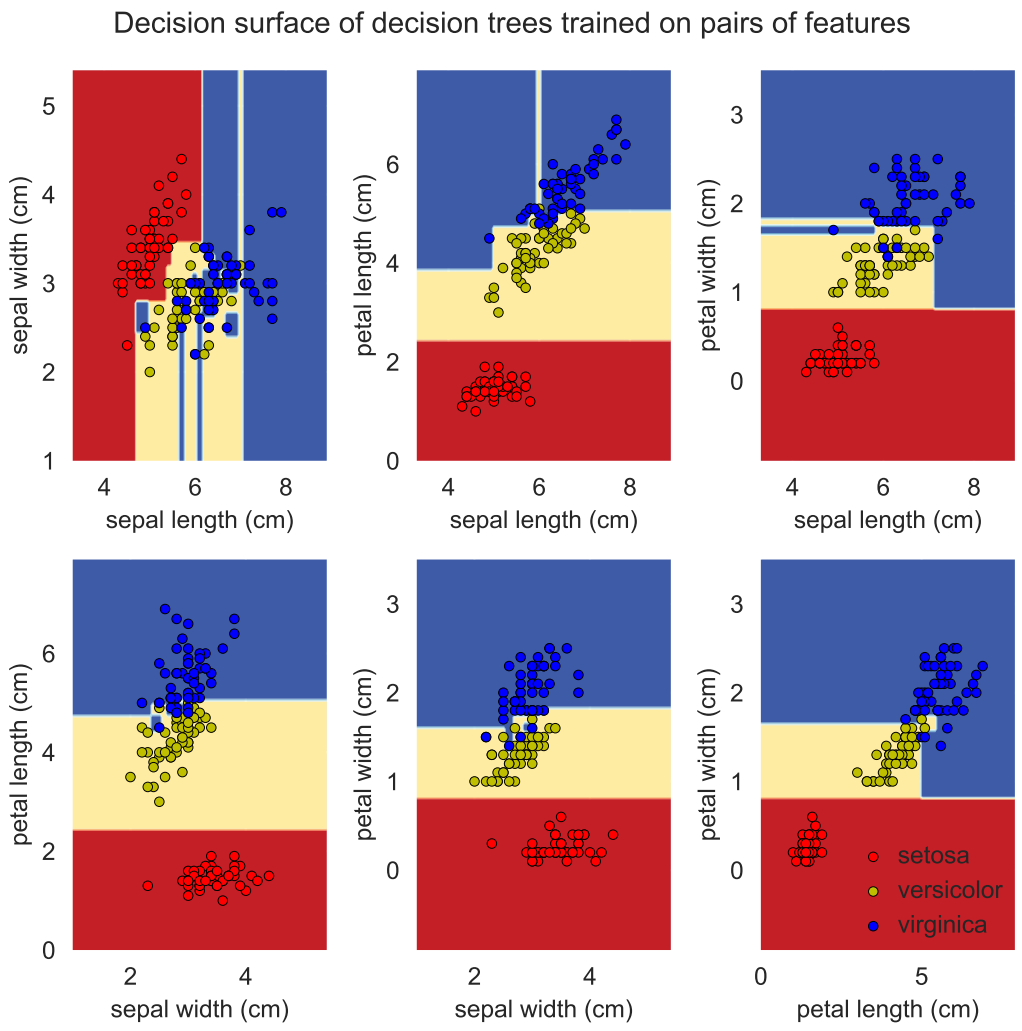

图 1: code1

Python 代码 2

```
# 导入操作系统库
import os
# 更改工作目录
os.chdir(r"D:\softwares\applied statistics\pythoncodelearning\chap5\sourcecode")
# 导入绘图库
import matplotlib.pyplot as plt
```

```
# 导入数据生成工具
from sklearn.datasets import load_iris
# 导入决策树分类器
from sklearn.tree import DecisionTreeClassifier
# 导入绘制树状图的工具
from sklearn.tree import plot_tree
# 导入绘图库中的字体管理包
from matplotlib import font_manager
# 实现中文字符正常显示
font = font_manager.FontProperties(fname=r"C:\Windows\Fonts\SimKai.ttf")
# 使用 seaborn 风格绘图
plt.style.use("seaborn-v0_8")
# 获取数据
iris = load_iris()
# 构建模型
clf = DecisionTreeClassifier()
# 模型拟合
clf.fit(iris.data, iris.target)
fig, ax = plt.subplots(figsize=(6,6), tight_layout=True)
plot_tree(clf, filled=True)
ax.set_title("Decision tree trained on all the iris features")
plt.show()
fig.savefig("../codeimage/code2.pdf")
```

Python 代码 3

```

# 导入操作系统库
import os
# 更改工作目录
os.chdir(r"D:\softwares\applied statistics\pythoncodelearning\chap5\sourcecode")
# 导入基础计算库
import numpy as np
# 导入绘图库
import matplotlib.pyplot as plt
# 导入决策树分类器
from sklearn.tree import DecisionTreeRegressor
# 导入绘图库中的字体管理包
from matplotlib import font_manager
# 实现中文字符正常显示
font = font_manager.FontProperties(fname=r"C:\Windows\Fonts\SimKai.ttf")
# 使用 seaborn 风格绘图
plt.style.use("seaborn-v0_8")
# 生成数据
np.random.seed(1)
```

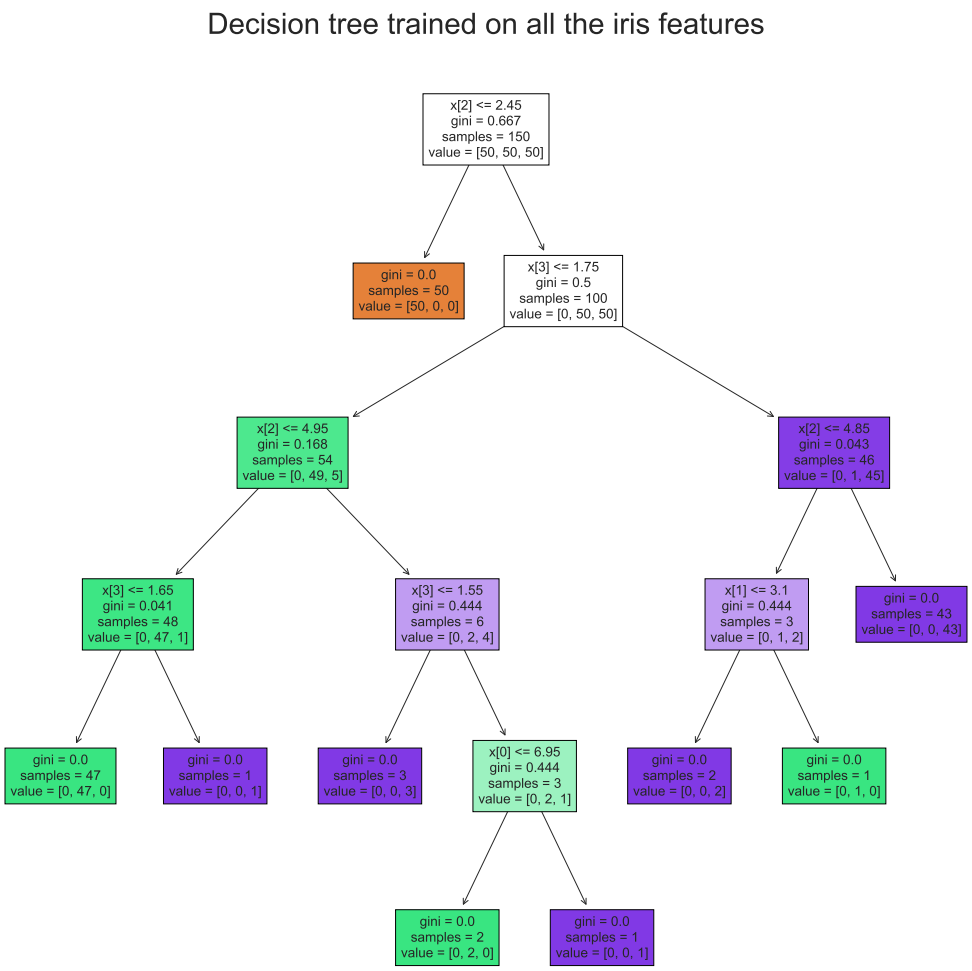
Decision tree trained on all the iris features
x[2] <= 2.45
gini = 0.667
samples = 150
value = [50, 50, 50]
gini = 0.0
samples = 50
value = [50, 0, 0]
x[3] <= 1.75
gini = 0.5
samples = 100
value = [0, 50, 50]
x[2] <= 4.95
gini = 0.168
samples = 54
value = [0, 49, 5]
x[2] <= 4.85
gini = 0.043
samples = 46
value = [0, 1, 45]
x[3] <= 1.65
gini = 0.041
samples = 48
value = [0, 47, 1]
x[3] <= 1.55
gini = 0.444
samples = 6
value = [0, 2, 4]
x[1] <= 3.1
gini = 0.444
samples = 3
value = [0, 1, 2]
gini = 0.0
samples = 43
value = [0, 0, 43]
gini = 0.0
samples = 47
value = [0, 47, 0]
gini = 0.0
samples = 1
value = [0, 0, 1]
gini = 0.0
samples = 3
value = [0, 0, 3]
x[0] <= 6.95
gini = 0.444
samples = 3
value = [0, 2, 1]
gini = 0.0
samples = 2
value = [0, 0, 2]
gini = 0.0
samples = 1
value = [0, 1, 0]
gini = 0.0
samples = 2
value = [0, 2, 0]
gini = 0.0
samples = 1
value = [0, 0, 1]

图 2: code2

```
X = np.sort(5 * np.random.rand(80, 1), axis=0)
y = np.sin(X).ravel()
y[::5] += 3 * (0.5 - np.random.rand(16))
# 构造回归树模型
regr_1 = DecisionTreeRegressor(max_depth=2)
regr_2 = DecisionTreeRegressor(max_depth=5)
# 模型拟合
regr_1.fit(X, y)
regr_2.fit(X, y)
# 预测
X_test = np.arange(0.0, 5.0, 0.01)[:, np.newaxis]
y_1 = regr_1.predict(X_test)
y_2 = regr_2.predict(X_test)
# 图示结果
fig, ax = plt.subplots(figsize=(6,6), tight_layout=True)
ax.scatter(X, y, s=20, edgecolor="black", c="darkorange", label="data")
ax.plot(X_test, y_1, color="cornflowerblue", label="max_depth=2", linewidth=2)
ax.plot(X_test, y_2, color="yellowgreen", label="max_depth=5", linewidth=2)
ax.set_xlabel("data")
ax.set_ylabel("target")
ax.set_title("Decision Tree Regression")
ax.legend()
plt.show()
fig.savefig("../codeimage/code3.pdf")
```

Python 代码 4

```
# 导入操作系统库
import os
# 更改工作目录
os.chdir(r"D:\softwares\applied statistics\pythoncodelearning\chap5\sourcecode")
# 导入绘图库
import matplotlib.pyplot as plt
# 导入数据集划分工具
from sklearn.model_selection import train_test_split
# 导入数据集工具
from sklearn.datasets import load_breast_cancer
# 导入分类树
from sklearn.tree import DecisionTreeClassifier
# 导入绘图库中的字体管理包
from matplotlib import font_manager
# 实现中文字符正常显示
font = font_manager.FontProperties(fname=r"C:\Windows\Fonts\SimKai.ttf")
# 使用 seaborn 风格绘图
```

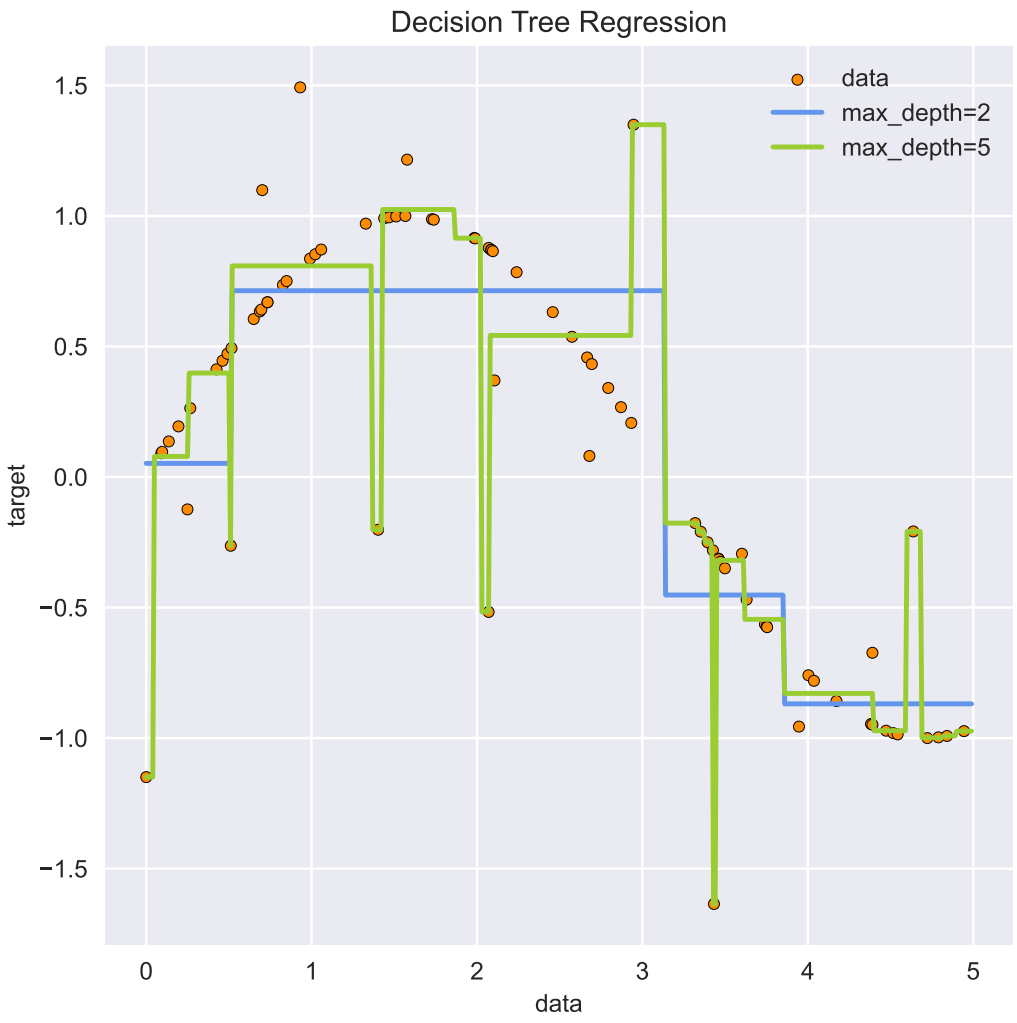
Decision Tree Regression
data
max_depth=2
max_depth=5
target
data
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
0
1
2
3
4
5

图 3: code3

```
plt.style.use("seaborn-v0_8")
# 获取数据
X, y = load_breast_cancer(return_X_y=True)
# 数据集划分
X_train, X_test, y_train, y_test = train_test_split(
    X, y, random_state=0
)
# 构建决策树
clf = DecisionTreeClassifier(random_state=0)
# 代价复杂度路径
path = clf.cost_complexity_pruning_path(X_train, y_train)
# cp
ccp_alphas, impurities = path.ccp_alphas, path.impurities
# 绘图
fig, ax = plt.subplots(figsize=(6,6))
ax.plot(ccp_alphas[:-1], impurities[:-1], marker="o", drawstyle="steps-post")
ax.set_xlabel("effective alpha")
ax.set_ylabel("total impurity of leaves")
ax.set_title("Total Impurity vs effective alpha for training set")
plt.show()
fig.savefig("../codeimage/code4.pdf")
# 不同的 alpha 下构建决策树
clfs = []
for ccp_alpha in ccp_alphas:
    # 构建模型
    clf = DecisionTreeClassifier(
        random_state=0, ccp_alpha=ccp_alpha
    )
    # 模拟拟合
    clf.fit(X_train, y_train)
    # 加入到列表中
    clfs.append(clf)
print(
    "Number of nodes in the last tree is: {} with ccp_alpha: {}".format(
        clfs[-1].tree_.node_count, ccp_alphas[-1]
    )
)
# 去掉最后一个
clfs = clfs[:-1]
ccp_alphas = ccp_alphas[:-1]
# 节点数量
node_counts = [clf.tree_.node_count for clf in clfs]
# 深度
```

```
depth = [clf.tree_.max_depth for clf in clfs]
fig, ax = plt.subplots(2, 1, figsize=(6,6), tight_layout=True)
ax[0].plot(ccp_alphas, node_counts, marker="o", drawstyle="steps-post")
ax[0].set_xlabel("alpha")
ax[0].set_ylabel("number of nodes")
ax[0].set_title("Number of nodes vs alpha")
ax[1].plot(ccp_alphas, depth, marker="o", drawstyle="steps-post")
ax[1].set_xlabel("alpha")
ax[1].set_ylabel("depth of tree")
ax[1].set_title("Depth vs alpha")
plt.show()
fig.savefig("../codeimage/code5.pdf")
# 预测精度和 alpha 的关系
# 训练集的准确率
train_scores = [clf.score(X_train, y_train) for clf in clfs]
# 测试集的准确率
test_scores = [clf.score(X_test, y_test) for clf in clfs]
fig, ax = plt.subplots(figsize=(6,6))
ax.set_xlabel("alpha")
ax.set_ylabel("accuracy")
ax.set_title("Accuracy vs alpha for training and testing sets")
ax.plot(ccp_alphas, train_scores, marker="o", label="train",␣
 ↪drawstyle="steps-post")
ax.plot(ccp_alphas, test_scores, marker="o", label="test", drawstyle="steps-post")
ax.legend()
plt.show()
fig.savefig("../codeimage/code6.pdf")

Number of nodes in the last tree is: 1 with ccp_alpha: 0.3272984419327777
```

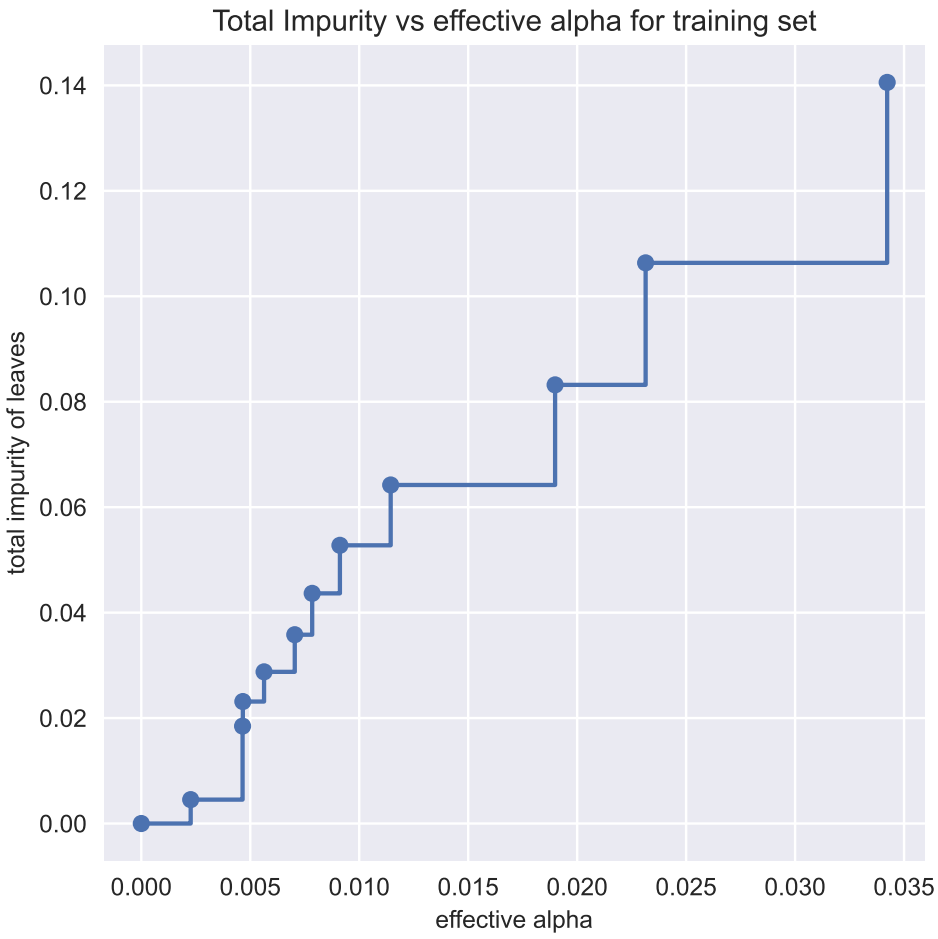
Total Impurity vs effective alpha for training set
total impurity of leaves
effective alpha
0.00
0.02
0.04
0.06
0.08
0.10
0.12
0.14
0.000
0.005
0.010
0.015
0.020
0.025
0.030
0.035

图 4: code4

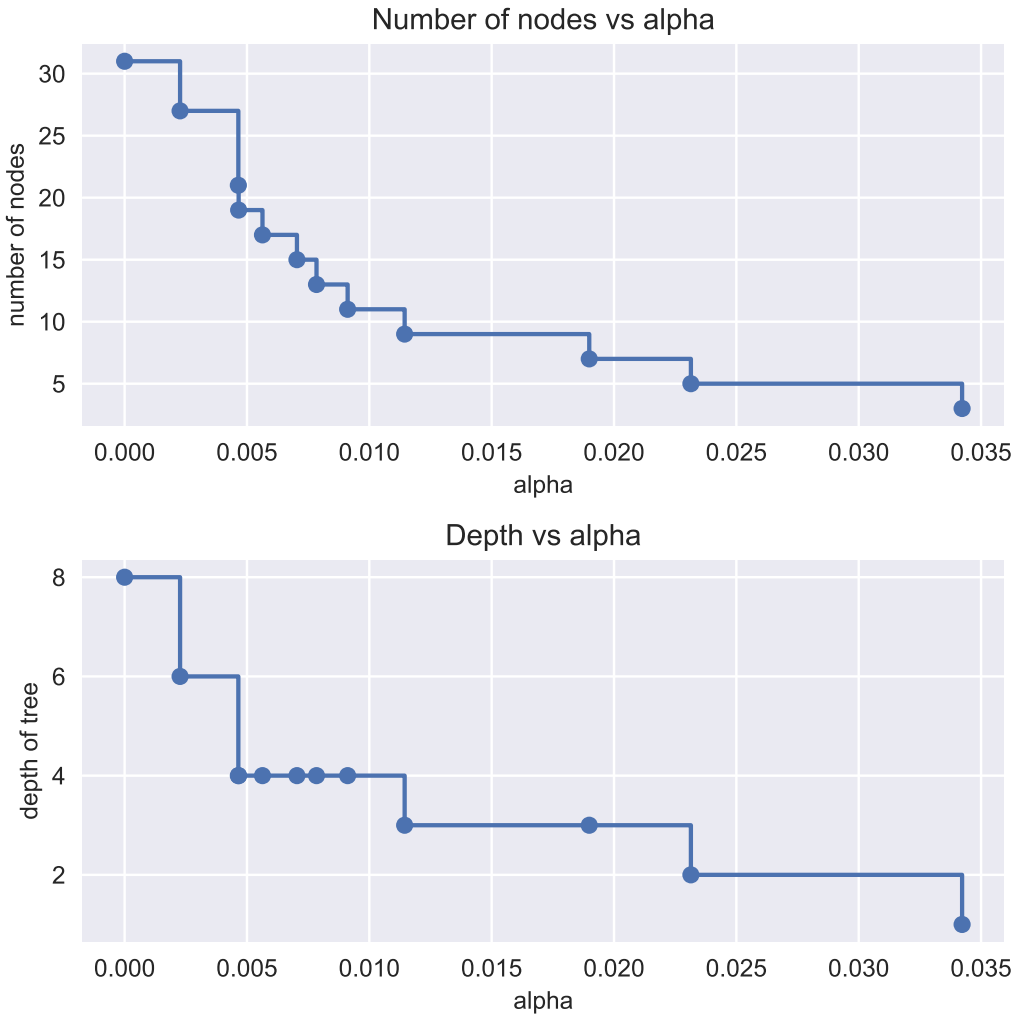
Number of nodes vs alpha
number of nodes
30
25
20
15
10
5
0.000
0.005
0.010
0.015
0.020
0.025
0.030
0.035
alpha
Depth vs alpha
depth of tree
8
6
4
2
0.000
0.005
0.010
0.015
0.020
0.025
0.030
0.035
alpha

图 5: code5

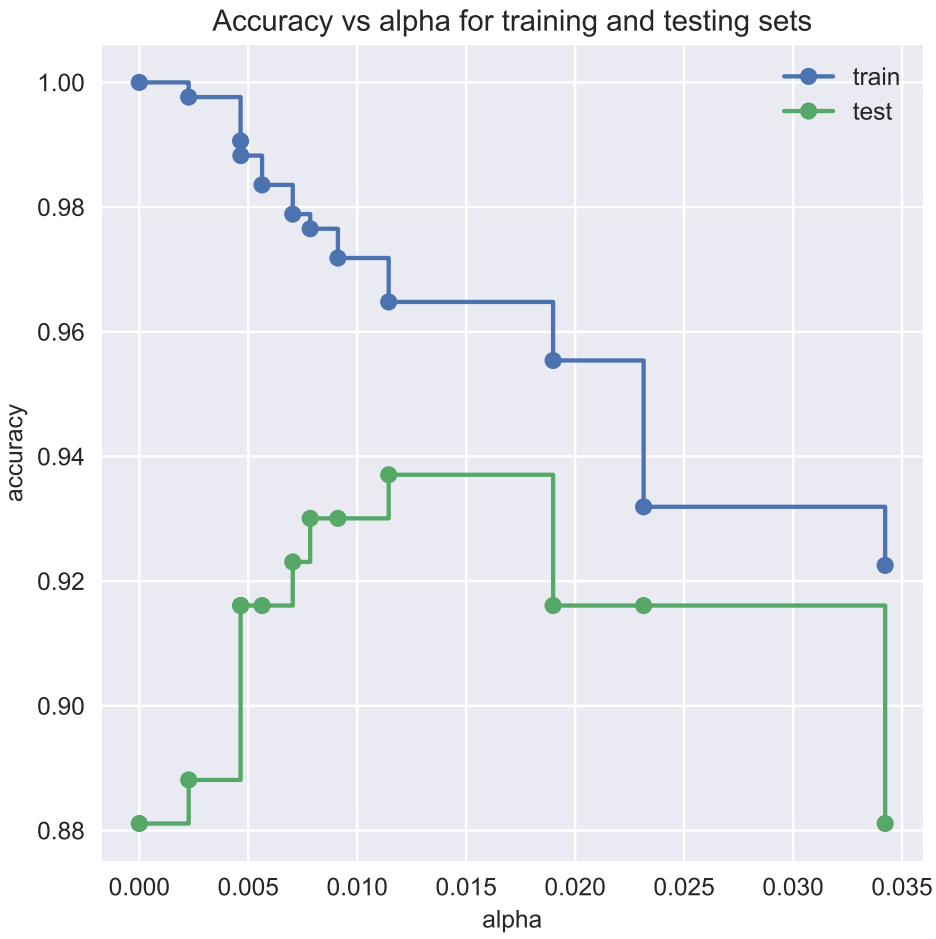
Accuracy vs alpha for training and testing sets
train
test
accuracy
alpha
1.00
0.98
0.96
0.94
0.92
0.90
0.88
0.000
0.005
0.010
0.015
0.020
0.025
0.030
0.035

图 6: code6