实验名称：线性回归

实验人：刘逸杰，学号：202210310115

一、实验目的

学会并掌握使用 python 实现决策树算法

二、实验内容

1. 实现决策树算法
2. 决策树的数据分割
3. 决策树的特征分析

三、实验步骤

代码结果与过程分析：

Import

```
from sklearn import tree
from sklearn.datasets import load_wine
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
import pprint
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

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

导入库与工具

```
wine = load_wine()
wine.data.shape
```

```
(178, 13)
```

```
wine.target
```

```
array([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, 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, 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])
```

```
wine.target_names
```

```
array(['class_0', 'class_1', 'class_2'], dtype='<U7')
```

```
wine.feature_names
```

```
['alcohol',
 'malic_acid',
 'ash',
 'alcalinity_of_ash',
 'magnesium',
 'total_phenols',
 'flavanoids',
 'nonflavanoid_phenols',
 'proanthocyanins',
 'color_intensity',
 'hue',
 'od280/od315_of_diluted_wines',
 'proline']
```

```python
X = pd.DataFrame(data=wine["data"],columns=wine.feature_names)
Y = pd.DataFrame(data=wine["target"],columns=['target'])
df = pd.concat([X,Y],axis=1)
df.head()
```

Python

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonflavanoid_phenols	proanthocyanins	color_intensity	hue	od280/od315_of_dilute
0	14.23	1.71	2.43	15.6	127.0	2.80	3.06	0.28	2.29	5.64	1.04	
1	13.20	1.78	2.14	11.2	100.0	2.65	2.76	0.26	1.28	4.38	1.05	
2	13.16	2.36	2.67	18.6	101.0	2.80	3.24	0.30	2.81	5.68	1.03	
3	14.37	1.95	2.50	16.8	113.0	3.85	3.49	0.24	2.18	7.80	0.86	
4	13.24	2.59	2.87	21.0	118.0	2.80	2.69	0.39	1.82	4.32	1.04	

传入数据

```python
x_train,x_test,y_train,y_test = train_test_split(X,Y,test_size=0.3,shuffle=True,random_state=13)
print(f"x_train: {x_train.shape}\ty_train: {y_train.shape}\nx_test: {x_test.shape}\ty_test: {y_test.shape}")
```

```
x_train: (124, 13)      y_train: (124, 1)
x_test: (54, 13)        y_test: (54, 1)
```

数据集划分

```python
clf = tree.DecisionTreeClassifier(criterion='entropy',random_state=30)
clf = clf.fit(x_train, y_train)
score = clf.score(x_test,y_test)
```

使用 entropy 作为分裂标准，创建并训练一个决策树分类器，并评估测试集得分

```
score

0.9814814814814815
```

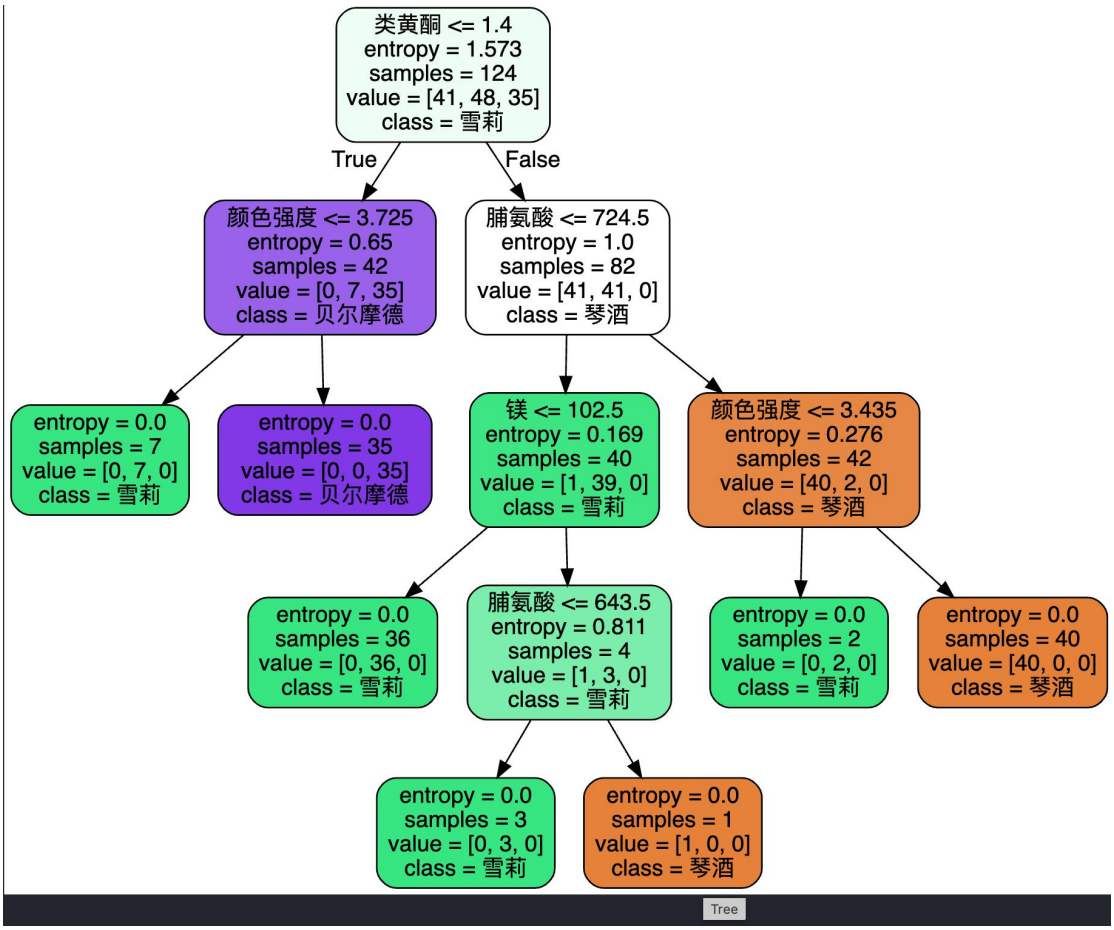
显示训练模型在测试集上的得分。

```python
feature_name = ['酒精','苹果酸','灰','灰的碱性','镁','总酚','类黄酮','非黄烷类酚类','花青素','颜色强度','色调','od280/od315稀释葡萄酒','脯氨酸']
pd.DataFrame([*zip(feature_name,clf.feature_importances_)],columns=['feature','importance'])
```

	feature	importance
0	酒精	0.000000
1	苹果酸	0.000000
2	灰	0.000000
3	灰的碱性	0.000000
4	镁	0.017950
5	总酚	0.000000
6	类黄酮	0.439648
7	非黄烷类酚类	0.000000
8	花青素	0.000000
9	颜色强度	0.199434
10	色调	0.000000
11	od280/od315稀释葡萄酒	0.000000
12	脯氨酸	0.342968

显示各个特征的重要性分数。

```
import graphviz
dot_data = tree.export_graphviz(clf
                                ,feature_names= feature_name
                                ,class_names=["琴酒","雪莉","贝尔摩德"]
                                ,filled=True
                                ,rounded=True
                                )
graph = graphviz.Source(dot_data)
graph
```



使用 Graphviz 库可视化决策树结构。

附加实验

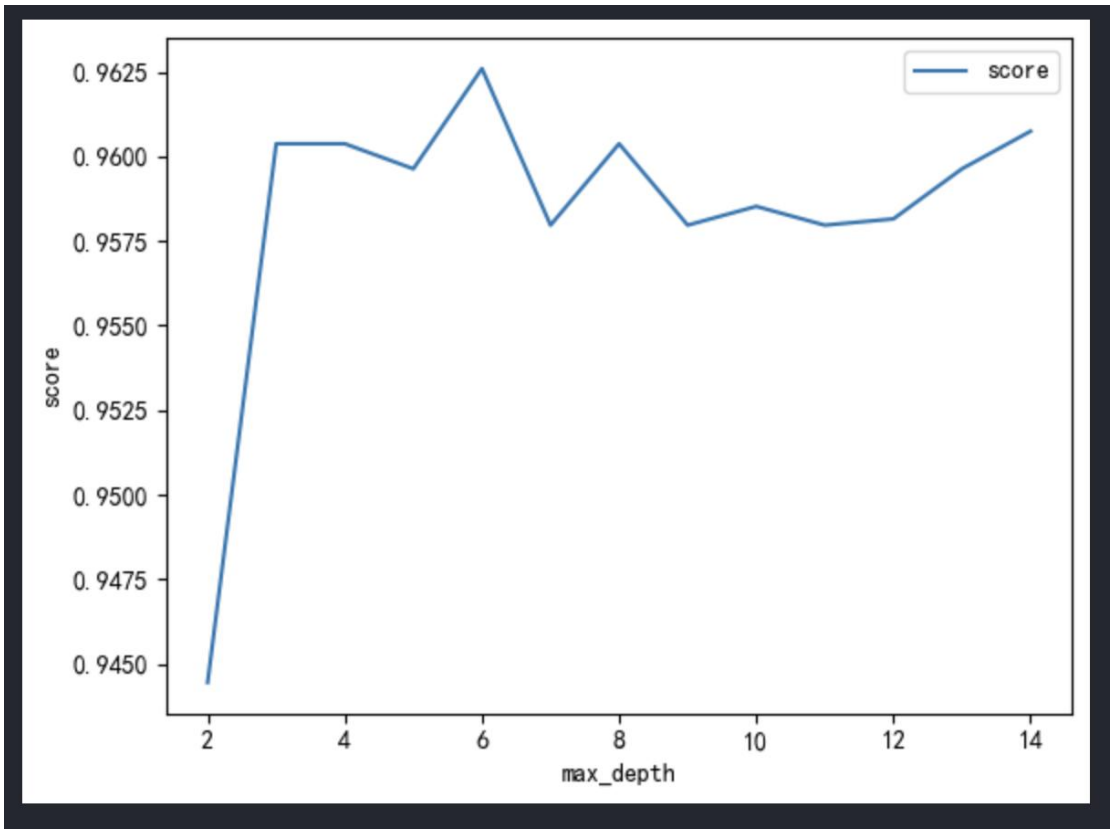
```
from sklearn.model_selection import RandomizedSearchCV
from sklearn.model_selection import GridSearchCV
```

14]

```
x_train,x_test,y_train,y_test = train_test_split(X,Y,test_size=0.3,shuffle=True,random_state=13)

max_depths = range(2,15)
min_samples_leafs = range(2,15)
min_samples_splits = range(2,15)

# max_depth
score = []
for max_depth in max_depths:
    clf = tree.DecisionTreeClassifier(criterion='entropy',max_depth=max_depth)
    _sc = []
    for i in range(100):
        clf.fit(x_train,y_train)
        _sc.append(clf.score(x_test,y_test))
    score.append([np.array(_sc).mean(),max_depth])
score = np.array(score)

plt.plot(score[:,1],score[:,0],label='score')
plt.xlabel("max_depth")
plt.ylabel('score')
plt.legend()
plt.show()
```

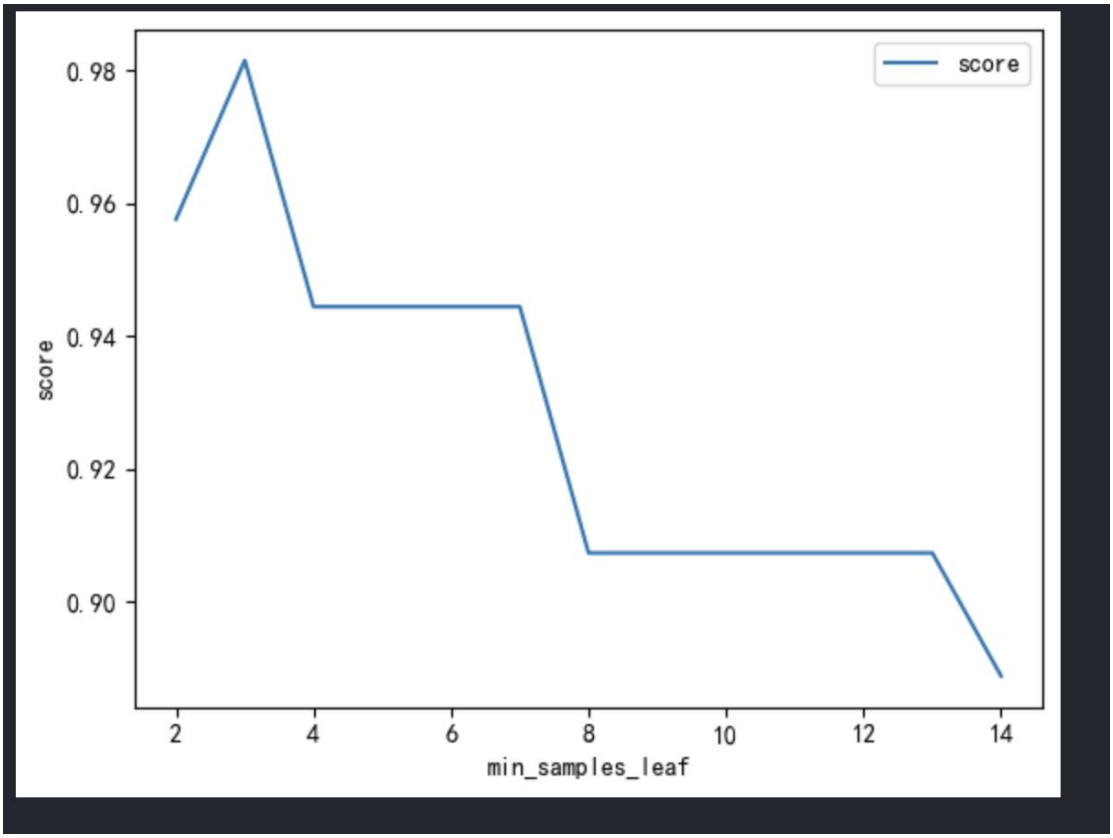
46]

导入库以便后续优化决策树参数。



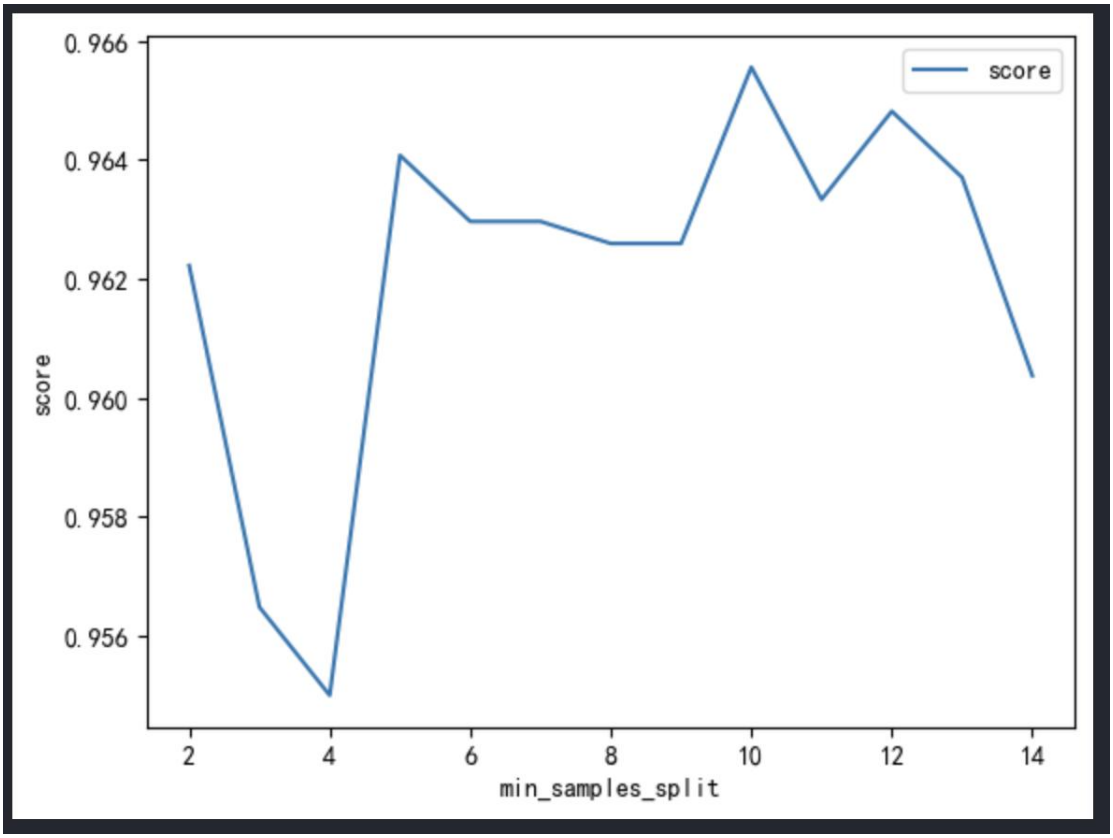
调整 max_depth 参数并记录评分

```
# min_samples_leaf
score = []
for min_samples_leaf in min_samples_leafs:
    clf = tree.DecisionTreeClassifier(criterion='entropy',min_samples_leaf=min_samples_leaf)
    _sc = []
    for i in range(100):
        clf.fit(x_train,y_train)
        _sc.append(clf.score(x_test,y_test))
    score.append([np.array(_sc).mean(),min_samples_leaf])
score = np.array(score)

plt.plot(score[:,1],score[:,0],label='score')
plt.xlabel("min_samples_leaf")
plt.ylabel('score')
plt.legend()
plt.show()
```



调整 min_samples_leaf 参数并记录评分

```
# min_samples_split
score = []
for min_samples_split in min_samples_splits:
    clf = tree.DecisionTreeClassifier(criterion='entropy',min_samples_split=min_samples_split)
    _sc = []
    for i in range(100):
        clf.fit(x_train,y_train)
        _sc.append(clf.score(x_test,y_test))
    score.append([np.array(_sc).mean(),min_samples_split])
score = np.array(score)

plt.plot(score[:,1],score[:,0],label='score')
plt.xlabel("min_samples_split")
plt.ylabel('score')
plt.legend()
plt.show()

clf = tree.DecisionTreeClassifier(max_depth=2,min_samples_leaf=10,min_samples_split=5)
clf.fit(x_test,y_test)
```



调试 min_samples_split 参数

```
print("score:\t\t\t", round(clf.score(x_test, y_test), 3))
```

```
score:                  1.0
```

输出得分

四、本次实验总结

在使用 graphviz 时发现 anaconda-navigator 的 Jupyter notebook 并不支持调用这个库，在调试无果后我选择了在本地部署运行 ipynb，经过环境配置调试后最终还是生成调用成功了。