实验目录

实验内容

1. 基于 Bagging 的糖尿病预测模型

知识点

1) Bagging 算法通过基分类器“投票”选出分类
2) 集成学习需要使用其他算法作为基分类器
3) 当基分类器准确率大于 50%时，集成学习效果优于基分类器
4) 通常来说，基分类器越多，集成学习效果越稳定

实验目的

1) 学习建立基于决策树的 Bagging 模型
2) 学习使用 Bagging 模型预测糖尿病发病情况

实验步骤

2）读取数据

1. Jupyter 输入代码后，使用 shift+enter 执行，下同。
2. 本数据集是“皮马印第安人糖尿病问题”。这个问题包括 768 个对于皮马印第安患者的医疗观测细节，记录所描述的瞬时测量取自诸如患者的年纪，怀孕和血液检查的次数。所有患者都是 21 岁以上（含 21 岁）的女性。字段说明如下：
   preg: Number of times pregnant
   plas: Plasma glucose concentration a 2 hours in an oral glucose tolerance test
   pres: Diastolic blood pressure (mm Hg)
   skin: Triceps skin fold thickness (mm)
   test: 2-Hour serum insulin (mu U/ml)
   mass: Body mass index (weight in kg/(height in m)^2)
   pedi: Diabetes pedigree function
   age: Age (years)
   class: Class variable (0 or 1)
3. 使用 pandas 读取文件，并查看数据前 5 行

[Code 001]:

import pandas as pd
df = pd.read_csv('/root/experiment/datas/pima-indians-diabetes.data.csv',index_col=0)
df.head()

```
import pandas as pd

df = pd.read_csv('/root/experiment/datas/pima-indians-diabetes.data.csv',index_col=0)
df.head()
```

	plas	pres	skin	test	mass	pedi	age	class
preg								
6	148	72	35	0	33.6	0.627	50	1
1	85	66	29	0	26.6	0.351	31	0
8	183	64	0	0	23.3	0.672	32	1
1	89	66	23	94	28.1	0.167	21	0
0	137	40	35	168	43.1	2.288	33	1

3) 数据描述与可视化分析

1. 查看数据统计分布

[Code 002]:

df.describe()

```
df.describe()
```

	plas	pres	skin	test	mass	pedi	age	class
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000
mean	120.894531	69.105469	20.536458	79.799479	31.992578	0.471876	33.240885	0.348958
std	31.972618	19.355807	15.952218	115.244002	7.884160	0.331329	11.760232	0.476951
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.078000	21.000000	0.000000
25%	99.000000	62.000000	0.000000	0.000000	27.300000	0.243750	24.000000	0.000000
50%	117.000000	72.000000	23.000000	30.500000	32.000000	0.372500	29.000000	0.000000
75%	140.250000	80.000000	32.000000	127.250000	36.600000	0.626250	41.000000	1.000000
max	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000	81.000000	1.000000

2. 查看数据缺失值

[Code 003]:

df.isnull().sum()

```
df.isnull().sum()
```

```
plas     0
pres     0
skin     0
test     0
mass     0
pedi     0
age      0
class    0
dtype: int64
```

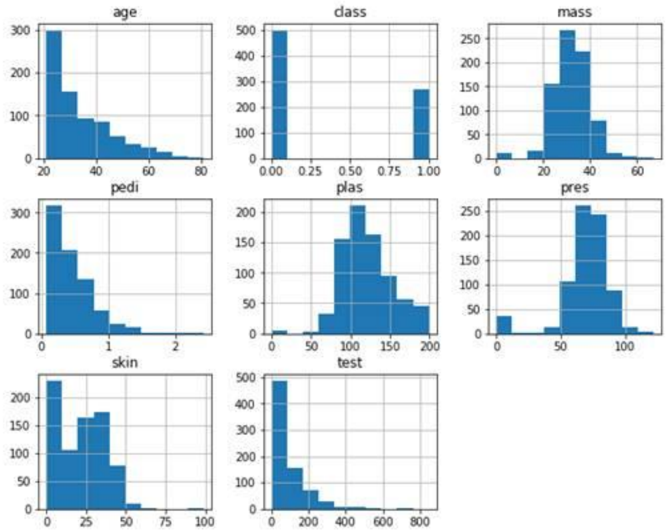
3. 查看数据分布（绘图时，由于 jupyter 的问题，执行时可能需重复执行才能显示绘图结果，下同）

[Code 004]:

import matplotlib.pyplot as plt
df.hist(figsize=(10,8))
plt.show()

```
import matplotlib.pyplot as plt

df.hist(figsize=(10,8))
plt.show()
```



4) 数据预处理

1. 划分自变量和因变量，训练集和测试集

[Code 005]:

 # 划分自变量和因变量
X = df.loc[:,df.columns!='class']
y = df.loc[:,df.columns=='class']
 # 划分训练集和测试集
from sklearn.model_selection import train_test_split
X_tr,X_ts,y_tr,y_ts = train_test_split(X,y)
X_tr.shape,X_ts.shape

```
X = df.loc[:,df.columns!='class']
y = df.loc[:,df.columns=='class']
```

```
from sklearn.model_selection import train_test_split

X_tr,X_ts,y_tr,y_ts = train_test_split(X,y)
X_tr.shape,X_ts.shape
```

```
((576, 7), (192, 7))
```

5) 建立模型

1. 分别建立决策树模型和 Bagging 模型

[Code 006]:

from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import BaggingClassifier
dtc = DecisionTreeClassifier()
bgc = BaggingClassifier(base_estimator=dtc,n_estimators=100)
dtc = dtc.fit(X_tr,y_tr)
bgc = bgc.fit(X_tr,y_tr.values.ravel())

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import BaggingClassifier
```

```
dtc = DecisionTreeClassifier()
bgc = BaggingClassifier(base_estimator=dtc,n_estimators=100)

dtc = dtc.fit(X_tr,y_tr)
bgc = bgc.fit(X_tr,y_tr.values.ravel())
```

6) 模型预测与评估

1. 对两个模型分别使用测试集预测，并计算 f1-score 和 accuracy

[Code 007]:

y_dtc_pred = dtc.predict(X_ts)
y_bgc_pred = bgc.predict(X_ts)
from sklearn.metrics import f1_score,accuracy_score
print('f1-score:%.4f'%f1_score(y_ts,y_dtc_pred))
print('accuracy:%.4f \n'%accuracy_score(y_ts,y_dtc_pred))
print('f1-score:%.4f'%f1_score(y_ts,y_bgc_pred))
print('accuracy:%.4f'%accuracy_score(y_ts,y_bgc_pred))

```
y_dtc_pred = dtc.predict(X_ts)
y_bgc_pred = bgc.predict(X_ts)
```

```
from sklearn.metrics import f1_score,accuracy_score
```

```
print('f1-score:%.4f'%f1_score(y_ts,y_dtc_pred))
print('accuracy:%.4f \n'%accuracy_score(y_ts,y_dtc_pred))
print('f1-score:%.4f'%f1_score(y_ts,y_bgc_pred))
print('accuracy:%.4f'%accuracy_score(y_ts,y_bgc_pred))
```

```
f1-score:0.6309
accuracy:0.7135

f1-score:0.6715
accuracy:0.7656
```

2. 使用 10 折交叉验证计算决策树模型 f1-socre

[Code 008]:

from sklearn.model_selection import cross_val_score
cross_val_score(dtc,X,y,scoring='f1',cv=10).mean()

```
from sklearn.model_selection import cross_val_score

cross_val_score(dtc,X,y,scoring='f1',cv=10).mean()
```

```
0.576927757131499
```

3. 使用 10 折交叉验证计算 Bagging 模型 f1-socre

[Code 009]:

cross_val_score(bgc,X,y.values.ravel(),scoring='f1',cv=10).mean()

```
cross_val_score(bgc,X,y.values.ravel(),scoring='f1',cv=10).mean()
```

```
0.625295867179608
```

7) 实验结论