实验目录

1. 基于决策树优化模型的泰坦尼克号幸存者预测

实验内容

1. 基于决策树优化模型的泰坦尼克号幸存者预测

知识点

1) 决策树模型可以通过优化超参数来提高模型预测准确率
2) 可以优化的主要超参数包括：叶节点最小样本数、最大树深度、分裂所需最小样本数等。

实验目的

1) 学习使用格搜索优化决策树模型，比较优化前后模型准确率

实验步骤

1) 打开 Jupyter，并新建 python 工程

2) 读取数据

1. Jupyter 输入代码后，使用 shift+enter 执行，下同。
2. 数据集包含泰坦尼克号 891 名乘员的基本信息，及幸存数据。字段说明如下：

   survival: Survival 0 = No, 1 = Yes

   pclass: Ticket class 1 = 1st, 2 = 2nd, 3 = 3rd

   sex: Sex

   Age: Age in years

sibsp: # of siblings / spouses aboard the Titanic

parch: # of parents / children aboard the Titanic

ticket: Ticket number

fare: Passenger fare

cabin: Cabin number

embarked: C = Cherbourg, Q = Queenstown, S = Southampton

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/titanic.csv',index_col=0)
df.head()

```
import pandas as pd

df = pd.read_csv('/root/experiment/datas/titanic.csv',index_col=0)
df.head()
```

	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
PassengerId											
1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

3) 描述性分析与可视化分析

1. 查看数据的统计描述

[Code 002]:

df.describe()

```
df.describe()
```

	Survived	Pclass	Age	SibSp	Parch	Fare
count	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
min	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
50%	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

2. 查看缺失值

[Code 003]:

df.isnull().sum()

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

```
Survived      0
Pclass        0
Name          0
Sex           0
Age         177
SibSp         0
Parch         0
Ticket        0
Fare          0
Cabin       687
Embarked      2
dtype: int64
```

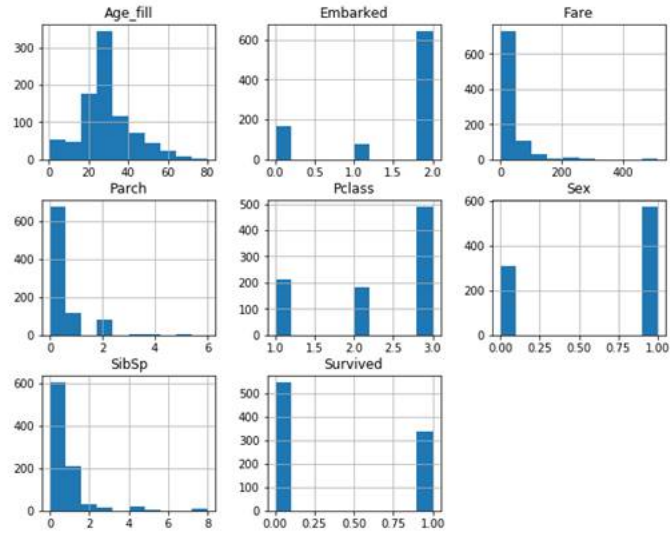
3. 查看数据分布（绘图时，由于 jupyter 的问题，执行时可能需重复执行才能显示绘图结果，下同）

[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. 去掉无关字段

[Code 005]:

df = df.drop(['Name','Ticket','Cabin'],axis=1)
df.columns

```
df = df.drop(['Name','Ticket','Cabin'],axis=1)
df.columns
```

```
Index(['Survived', 'Pclass', 'Sex', 'Age', 'SibSp', 'Parch', 'Fare',
       'Embarked'],
      dtype='object')
```

2. 填充 Age 缺失值

[Code 006]:

mean_age = df['Age'].mean()
tmp = df['Age'].copy()
tmp[df.Age.isnull()] = mean_age
df['Age_fill'] = tmp
del tmp
df = df.drop(['Age'],axis=1)
df.columns

```
mean_age = df['Age'].mean()

tmp = df['Age'].copy()
tmp[df.Age.isnull()] = mean_age
df['Age_fill'] = tmp
del tmp
df = df.drop(['Age'],axis=1)
df.columns
```

```
Index(['Survived', 'Pclass', 'Sex', 'SibSp', 'Parch', 'Fare', 'Embarked',
       'Age_fill'],
      dtype='object')
```

3. 使用 LabelEncoder 将离散变量转换为编码

[Code 007]:

from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
df_sex = le.fit(df['Sex'])
df['Sex'] = df_sex.transform(df['Sex'])
df = df.dropna()
df_embarked = le.fit(df['Embarked'])
df['Embarked'] = df_embarked.transform(df['Embarked'])
df.info()

```
from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()
df_sex = le.fit(df['Sex'])
df['Sex'] = df_sex.transform(df['Sex'])
df = df.dropna()

df_embarked = le.fit(df['Embarked'])
df['Embarked'] = df_embarked.transform(df['Embarked'])

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 889 entries, 1 to 891
Data columns (total 8 columns):
Survived    889 non-null int64
Pclass      889 non-null int64
Sex         889 non-null int64
SibSp       889 non-null int64
Parch       889 non-null int64
Fare        889 non-null float64
Embarked    889 non-null int64
Age_fill    889 non-null float64
dtypes: float64(2), int64(6)
memory usage: 62.5 KB
```

4. 划分自变量和因变量，训练集和测试集

[Code 008]:

 # 划分自变量和因变量

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

 # 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!='Survived']
y = df.loc[:,df.columns=='Survived']
```

```
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
```

```
((666, 7), (223, 7))
```

5) 建立模型

1. 建立决策树模型

[Code 009]:

from sklearn.tree import DecisionTreeClassifier
dtc = DecisionTreeClassifier()
dtc = dtc.fit(X_tr,y_tr)

```
from sklearn.tree import DecisionTreeClassifier

dtc = DecisionTreeClassifier()
dtc = dtc.fit(X_tr,y_tr)
```

6) 模型预测与评估

1. 使用 5 折交叉验证计算决策树准确率

[Code 010]:

from sklearn.model_selection import cross_val_score
scores = cross_val_score(dtc,X,y,cv=5,scoring='accuracy')
scores.mean()

```
from sklearn.model_selection import cross_val_score

scores = cross_val_score(dtc,X,y,cv=5,scoring='accuracy')
scores.mean()
```

```
0.7716879324573097
```

7) 模型优化

1. 使用格搜索优化决策树模型

[Code 011]:

from sklearn.model_selection import GridSearchCV
parameters={'min_samples_split' : list(range(2,6,1)),
'max_depth': list(range(6,19,2)),
'criterion':('gini','entropy'),
'min_samples_leaf':list(range(2,9,2))}
clf = DecisionTreeClassifier()
gs = GridSearchCV(estimator=clf, param_grid = parameters,
scoring='accuracy',iid=False,cv=5,return_train_score=True)
gs.fit(X_tr, y_tr)
*gs.best_params_, gs.best_score_*

```
from sklearn.model_selection import GridSearchCV

parameters={'min_samples_split' : list(range(2,6,1)),
            'max_depth': list(range(6,19,2)),
            'criterion':('gini','entropy'),
            'min_samples_leaf':list(range(2,9,2))}

clf = DecisionTreeClassifier()
gs = GridSearchCV(estimator=clf, param_grid = parameters, scoring='accuracy',iid=False,cv=5,return_train_score=True)
gs.fit(X_tr, y_tr)
gs.best_params_, gs.best_score_
```

```
({'criterion': 'entropy',
  'max_depth': 8,
  'min_samples_leaf': 8,
  'min_samples_split': 2},
 0.8453448750777894)
```

2. 使用优化后的超参数建模并预测

[Code 012]:

*dtc_best = gs.best_estimator_*
dtc_best.fit(X_tr,y_tr)
y_pred = dtc_best.predict(X_ts)
y_pred[0:5]

```
dtc_best = gs.best_estimator_
dtc_best.fit(X_tr,y_tr)

y_pred = dtc_best.predict(X_ts)
y_pred[0:5]
```

```
array([1, 0, 0, 0, 1])
```

3. 使用 5 折交叉验证计算优化模型准确率

[Code 013]:

scores = cross_val_score(dtc_best,X,y,cv=5,scoring='accuracy')
scores.mean()

```
scores = cross_val_score(dtc_best,X,y,cv=5,scoring='accuracy')
scores.mean()
```

```
0.8133180981400369
```

8) 实验结论

1. 本试验中，决策树模型 5 折交叉验证准确率 0.772。

2. 本试验中，优化后的决策树模型 5 折交叉验证准确率 0.813。

3. 使用格搜索得到的决策树模型最优超参数如下：

   {'max_depth': 8,'min_samples_leaf': 8,'min_samples_split': 2}