# 决策树实现过程

用到的游戏用户数据直接进行过脱敏处理，且已经经过数据清洗与特征工程。用处理好的数据建立决策树模型。主要讲解使用scikit-learn来跑决策树算法，结果的可视化以及一些参数调参的关键点。（在实际的项目分析过程中，需要先结合业务需求对数据进行预处理，使之变成模型要求的形式，然后进行具体的建模。）

## 1、数据准备

首先载入需要的库，准备好建模数据

```
import pandas as pd
import numpy as np
#数据准备
df=pd.read_csv(r'D:\aboutfile\数据挖掘之旅\mycode\markdown准备\data.csv')
#按照7：3的比例划分训练数据集与测试数据集
from sklearn.model_selection import train_test_split#数据划分
x=df[['day','notLoginDays','dayVitality','intervalMean']]#特征属性
y=df['isLoss']#类别属性
x_train,x_test,y_train,y_test = train_test_split(x, y,
                                                 test_size = 0.3,
                                                 random_state = 1,
                                                 shuffle=True)
df.head(10)
```

查看前十条记录：

| id | day | notLoginDays | dayVitality | intervalMean | isLoss |
|---|---|---|---|---|---|
| 0 | 1 | 4 | 0 | 12 | 1 |
| 1 | 1 | 4 | 1 | 18 | 1 |
| 2 | 1 | 4 | 1 | 0 | 0 |
| 3 | 1 | 4 | 1 | 4 | 0 |
| 4 | 0 | 6 | 0 | 0 | 1 |
| 5 | 0 | 6 | 1 | 0 | 1 |
| 6 | 0 | 6 | 0 | 0 | 1 |
| 7 | 0 | 6 | 0 | 18 | 1 |
| 8 | 0 | 6 | 0 | 18 | 1 |
| 9 | 0 | 6 | 0 | 0 | 0 |

## 2、模型建立：sklearn库

**sklearn**是一个**Python**第三方提供的非常强力的机器学习库，它包含了从数据预处理到训练模型的各个方面。在实战使用**scikit-learn**中可以极大的节省我们编写代码的时间以及减少我们的代码量，使我们有更多的精力去分析数据分布，调整模型和修改超参。

调用sklearn库里的分类决策树模型（tree.DecisionTreeClassifier)。利用默认参数进行建模，查看模型效果。模型对实际流失用户的预测效果并不是很好，需要进一步调整模型参数，用到格点搜索。

```
from sklearn import tree
dtree_t=tree.DecisionTreeClassifier() #默认参数
dtree_t=dtree_t.fit(x_train, y_train) #学习决策树
#输出测试集的混淆矩阵
dtree_t_pred = dtree_t.predict(x_test) #对测试集的预测结果
dtree_t_train_pred = dtree_t.predict(x_train) #对训练集的预测结果
from sklearn import metrics
print("Decision Tree")
print(metrics.classification_report(y_test, dtree_t_pred)) #模型分类报告
#横为 true label  竖为 predict
print("混淆矩阵，横为真实，竖为预测")
print(metrics.confusion_matrix(y_test, dtree_t_pred)) #混淆矩阵
```

# 3、参数调整：网格搜索（GridSearch）

Grid Search：是一种调参手段，会对候选参数进行**穷举搜索**。在所有候选的参数选择中，通过循环遍历，尝试每一种可能性，表现最好的参数就是最终的结果。以有两个参数的模型为例，参数a有3种可能，参数b有4种可能，把所有可能性列出来，可以表示成一个3*4的表格，其中每个cell就是一个网格，循环过程就像是在每个网格里遍历、搜索，所以叫网格搜索（grid search）。

利用“网格搜索”（GridSearch）寻找最佳参数。对模型结果影响最大的参数（树深：max_depth）优先进行调整，确定参数的大致范围，缩小范围继续调参。

```
#创建决策树模型，利用“格点搜索”（GridSearch）寻找最佳参数
from sklearn.model_selection import GridSearchCV #格点搜索
from sklearn import tree #树模型
#设置目标参数的范围
param_test1 = {
 'max_depth':range(5,15),
 'min_samples_split':range(10,100,10),
 'min_samples_leaf':range(10,100,10)
}
#通过格点搜索寻找最佳模型参数
gsearch1 = GridSearchCV(estimator =tree.DecisionTreeClassifier(
                                    class_weight='balanced',
                                    criterion='gini',
                                    splitter='best',
                                    max_features=None,
                                    min_samples_split=20,
                                    min_samples_leaf=5,
                                    max_leaf_nodes=None,
                                    min_impurity_split=None,
                                    random_state=1),
 param_grid = param_test1, scoring='roc_auc',n_jobs=4,iid=False, cv=10)
gsearch1.fit(x_train, y_train)
#打印最佳模型参数
print(gsearch1.best_params_, gsearch1.best_score_)
```

# 4、建立最佳决策树模型

根据格点搜索得到的最佳模型参数建立分类决策树模型

```
from sklearn import tree #树模型
dtree =tree.DecisionTreeClassifier(max_depth=9,
                                   class_weight='balanced',
                                   criterion='gini',
                                   splitter='best',
                                   max_features=None,
                                   min_samples_split=10,
                                   min_samples_leaf=80,
                                   max_leaf_nodes=None,
                                   min_impurity_split=None,
                                   random_state=1)
dtree=dtree.fit(x_train, y_train) #学习决策树
```

# 5、输出模型分类报告和混淆矩阵

在分析问题时，需要结合业务情况。在游戏用户流失的问题中，决策人员更关心的是流失用户的情况——用户是否会流失，针对预测的情况去制定对应的策略。所以，在评价模型时，流失用户的召回率会是更重要的一个指标，代表实际流失的用户被预测出来的概率越高，它的含义类似：宁可错杀一千，绝不放过一个。本例中，流失用户的召回率达到了90%。

```
#输出测试集的混淆矩阵
dtree_pred = dtree.predict(x_test) #对测试集的预测结果
dtree_train_pred = dtree.predict(x_train) #对训练集的预测结果
from sklearn import metrics
print("Decision Tree")
print(metrics.classification_report(y_test, dtree_pred)) #模型分类报告
#横为 true label  竖为 predict
print("混淆矩阵，横为真实，竖为预测")
print(metrics.confusion_matrix(y_test, dtree_pred)) #混淆矩阵
```