|  |  |  |
| --- | --- | --- |
| **列名** | **含义** | **解释** |
| MONTH\_ID | 月份标识符 | 数据记录的月份，例如2016年3月。 |
| USER\_ID | 用户唯一标识符 | 客户的唯一标识符，用于区分不同的用户。 |
| INNET\_MONTH | 内网月份，可能与用户在该月份的活跃度相关。 | 用户在该月份的活跃度标识，具体含义根据业务定义。 |
| IS\_AGREE | 是否同意协议（例如电子合同等），1表示同意，0表示不同意。 | 用户是否同意了某种协议。 |
| AGREE\_EXP\_DATE | 协议到期日期，表示用户协议的过期时间。 | 用户协议的过期日期，例如2016年8月。 |
| CREDIT\_LEVEL | 用户信用等级，通常用于评估客户的信用风险。 | 用户的信用等级，数值越高表示信用越好。 |
| VIP\_LVL | 用户的VIP等级，表示用户的会员等级。 | 用户的VIP等级，通常与用户的忠诚度或购买力相关。 |
| ACCT\_FEE | 账户费用，指用户每月需要支付的费用金额。 | 用户的账户费用金额。 |
| CALL\_DURA | 通话总时长，表示用户在该月的总通话时长（单位：秒）。 | 用户在该月的总通话时长。 |
| NO\_ROAM\_LOCAL\_CALL\_DURA | 本地未漫游通话时长，指用户在本地没有漫游的情况下的通话时长。 | 用户在该月本地未漫游的通话时长。 |
| NO\_ROAM\_GN\_LONG\_CALL\_DURA | 全球未漫游长途通话时长，指用户在全球范围内不漫游的长途通话时长。 | 用户在全球范围内的未漫游长途通话时长。 |
| GN\_ROAM\_CALL\_DURA | 全球漫游通话时长，表示用户在国际漫游状态下的通话时长。 | 用户在全球漫游状态下的通话时长。 |
| CDR\_NUM | 通话记录总数，指用户在该月的通话记录条数。 | 用户该月通话记录的总数。 |
| NO\_ROAM\_CDR\_NUM | 本地未漫游通话记录数，指用户在本地未漫游的通话记录条数。 | 用户在本地未漫游状态下的通话记录数。 |
| NO\_ROAM\_LOCAL\_CDR\_NUM | 本地未漫游本地通话记录数，指用户在本地未漫游时的通话记录条数。 | 用户在本地未漫游时的本地通话记录数。 |
| NO\_ROAM\_GN\_LONG\_CDR\_NUM | 全球未漫游长途通话记录数，指用户在全球未漫游时的长途通话记录数。 | 用户在全球范围内未漫游时的长途通话记录数。 |
| GN\_ROAM\_CDR\_NUM | 全球漫游通话记录数，指用户在全球漫游状态下的通话记录条数。 | 用户在全球漫游状态下的通话记录数。 |
| P2P\_SMS\_CNT\_UP | 点对点短信上传数，表示用户发送的短信数量。 | 用户发送的P2P短信数量。 |
| TOTAL\_FLUX | 总流量，表示用户使用的数据流量总数。 | 用户在该月的总数据流量（单位：MB）。 |
| LOCAL\_FLUX | 本地流量，指用户在本地网络下使用的数据流量。 | 用户在本地网络下的使用流量。 |
| GN\_ROAM\_FLUX | 全球漫游流量，指用户在全球漫游状态下使用的数据流量。 | 用户在全球漫游状态下的流量。 |
| CALL\_DAYS | 通话天数，表示用户在该月内有通话记录的天数。 | 用户有通话记录的天数。 |
| CALLING\_DAYS | 主叫天数，表示用户发起通话的天数。 | 用户发起通话的天数。 |
| CALLED\_DAYS | 被叫天数，表示用户接听通话的天数。 | 用户接听通话的天数。 |
| CALL\_RING | 来电响铃次数，表示用户接听电话时的响铃次数。 | 用户接听电话时的响铃次数。 |
| CALLING\_RING | 主叫响铃次数，表示用户主叫时的响铃次数。 | 用户主叫时的响铃次数。 |
| CALLED\_RING | 被叫响铃次数，表示用户接听电话时的响铃次数。 | 用户接听电话时的响铃次数。 |
| CUST\_SEX | 客户性别，1表示男性，0表示女性。 | 用户的性别，可能需要额外的映射说明。 |
| CERT\_AGE | 客户证件年龄，表示用户的年龄。 | 用户的年龄。 |
| CONSTELLATION\_DESC | 星座，表示用户的星座。 | 用户的星座。 |
| MANU\_NAME | 手机制造商，表示用户使用的手机的品牌名称。 | 用户手机的品牌名称。 |
| MODEL\_NAME | 手机型号，表示用户使用的手机型号。 | 用户使用的手机型号。 |
| OS\_DESC | 操作系统描述，表示用户手机的操作系统。 | 用户手机的操作系统。 |
| TERM\_TYPE | 终端类型，表示用户所用的设备类型。 | 用户设备的类型，具体含义依业务定义。 |
| IS\_LOST | 是否流失，1表示流失，0表示未流失。 | 用户是否流失，0表示未流失，1表示已流失。 |

数据集具体解释如上，由于存在数值型数据，文本型数据（分类数据），我考虑采用**随机森林分类模型**（Random Forest Classifier）。

**1. 输入与目标**

**特征变量 X**：我选择了除 IS\_LOST 和其他不相关列（如 USER\_ID 和 AGREE\_EXP\_DATE）外的其他所有列，作为模型的输入特征。

**目标变量 y**：我将 IS\_LOST 作为目标变量，它表示是否流失的标签（0表示未流失，1表示已流失）。

**2. 数据拆分**

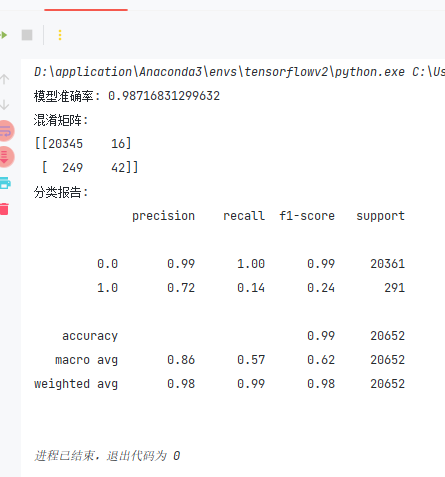
使用 train\_test\_split 将数据集按 80% / 20% 划分为训练集和测试集。随机种子 random\_state=42 保证每次拆分时相同的结果。

**3. 数据处理**

**数值特征**：包括所有可以进行算术计算的字段，如通话时长、费用、数据流量等。数值特征通常需要进行标准化处理，以便不同特征对模型的贡献更加平衡。对数值特征，我们使用了SimpleImputer来填充缺失值（填充均值），然后通过StandardScaler进行标准化。标准化的目的是让所有特征具有相似的尺度，这样可以让模型更容易学习。

**类别特征**：包括字符串字段和分类变量，如性别、星座、手机品牌等。类别特征采用OneHotEncoder，转换成数值类型特征。使用SimpleImputer填充缺失值（填充最频繁的值），然后通过OneHotEncoder对其进行独热编码，将每个类别变量转化为一组二进制变量。

**4. 训练与结果**



import pandas as pd  
from sklearn.model\_selection import train\_test\_split  
from sklearn.preprocessing import StandardScaler, OneHotEncoder  
from sklearn.compose import ColumnTransformer  
from sklearn.pipeline import Pipeline  
from sklearn.impute import SimpleImputer  
from sklearn.ensemble import RandomForestClassifier  
from sklearn.metrics import accuracy\_score, confusion\_matrix, classification\_report  
  
# 1. 读取数据  
data = pd.read\_csv("./user\_info\_screen.csv",encoding='gbk')  
  
# 2. 分割数据集为特征（X）和目标变量（y）  
X = data.drop(columns=['IS\_LOST', 'USER\_ID', 'AGREE\_EXP\_DATE']) # 删除不相关的列  
y = data['IS\_LOST']  
  
# 3. 分割训练集和测试集  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)  
  
# 4. 数值特征和类别特征区分  
numerical\_features = ['CREDIT\_LEVEL', 'VIP\_LVL', 'ACCT\_FEE', 'CALL\_DURA', 'NO\_ROAM\_LOCAL\_CALL\_DURA',  
 'NO\_ROAM\_GN\_LONG\_CALL\_DURA', 'GN\_ROAM\_CALL\_DURA', 'CDR\_NUM', 'NO\_ROAM\_CDR\_NUM',  
 'NO\_ROAM\_LOCAL\_CDR\_NUM', 'NO\_ROAM\_GN\_LONG\_CDR\_NUM', 'GN\_ROAM\_CDR\_NUM', 'P2P\_SMS\_CNT\_UP',  
 'TOTAL\_FLUX', 'LOCAL\_FLUX', 'GN\_ROAM\_FLUX', 'CALL\_DAYS', 'CALLING\_DAYS', 'CALLED\_DAYS',  
 'CALL\_RING', 'CALLING\_RING', 'CALLED\_RING', 'CERT\_AGE', 'TERM\_TYPE']  
  
categorical\_features = ['IS\_AGREE', 'INNET\_MONTH', 'CUST\_SEX', 'CONSTELLATION\_DESC', 'MANU\_NAME', 'MODEL\_NAME', 'OS\_DESC']  
  
# 5. 数值特征预处理  
# 使用SimpleImputer处理缺失值，并使用StandardScaler进行标准化  
numerical\_transformer = Pipeline(steps=[  
 ('imputer', SimpleImputer(strategy='mean')), # 用均值填充缺失值  
 ('scaler', StandardScaler()) # 标准化数据  
])  
  
# 6. 类别特征预处理  
# 使用SimpleImputer填补缺失值，使用OneHotEncoder进行独热编码  
categorical\_transformer = Pipeline(steps=[  
 ('imputer', SimpleImputer(strategy='most\_frequent')), # 用最频繁值填充缺失值  
 ('onehot', OneHotEncoder(handle\_unknown='ignore')) # 独热编码  
])  
  
# 7. 使用ColumnTransformer来将预处理应用到不同的列  
preprocessor = ColumnTransformer(  
 transformers=[  
 ('num', numerical\_transformer, numerical\_features),  
 ('cat', categorical\_transformer, categorical\_features)  
 ])  
  
# 8. 创建模型并加入预处理步骤  
model = Pipeline(steps=[  
 ('preprocessor', preprocessor),  
 ('classifier', RandomForestClassifier(random\_state=42)) # 使用随机森林作为分类器  
])  
  
# 9. 训练模型  
model.fit(X\_train, y\_train)  
  
# 10. 预测  
y\_pred = model.predict(X\_test)  
  
# 11. 输出模型评估  
accuracy = accuracy\_score(y\_test, y\_pred)  
print(f"模型准确率: {accuracy}")  
  
# 混淆矩阵  
conf\_matrix = confusion\_matrix(y\_test, y\_pred)  
print("混淆矩阵:")  
print(conf\_matrix)  
  
# 分类报告  
class\_report = classification\_report(y\_test, y\_pred)  
print("分类报告:")  
print(class\_report)