**课程编号：A0801052411**

**《数据挖掘原理与算法（2）》**

**消费者人群画像-信用智能评分**



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **姓名** |  | **学号** | |  |
| **班级** |  | **指导教师** | |  |
| **实验名称** | **消费者人群画像-信用智能评分** | | | |
| **开设学期** | **2021-2022秋季学期** | | | |
| **开设时间** | **第6周——第X周** | | | |
| **报告日期** | **2021年11月16日** | | | |
| **评定成绩** |  | | **评定人** | **裴永余** |
| **评定日期** |  |

**东北大学软件学院**

**目录**

[一、项目背景 4](#_Toc1816)

[二、数据集说明 4](#_Toc5556)

[2.1数据集名称 4](#_Toc8246)

[2.2数据集来源 5](#_Toc15070)

[2.3数据集介绍 5](#_Toc25802)

[2.4数据集字段说明 5](#_Toc16400)

[2.5数据集概览 7](#_Toc3854)

[三、实验目的 7](#_Toc31697)

[四、实验环境 8](#_Toc29635)

[五、 实验流程设计 9](#_Toc12010)

[六、实验具体步骤 11](#_Toc15216)

[6.1数据集探索 11](#_Toc10017)

[6.1.1数据集获取 11](#_Toc3758)

[6.1.2 pandas读取数据性能优化 11](#_Toc31955)

[6.1.3数据集基本信息 14](#_Toc5617)

[6.2数据预处理 18](#_Toc27170)

[6.2.1缺失值处理 18](#_Toc28772)

[6.2.2 dask框架并行加速计算 20](#_Toc28884)

[6.2.3基于dask框架和正则表达式的异常值处理 21](#_Toc8911)

[6.2.4无用值处理 22](#_Toc27688)

[6.2.5重复值处理 22](#_Toc18095)

[6.2.6离群点处理-基于异常检测模型 23](#_Toc22040)

[6.3数据校验 25](#_Toc17109)

[6.3.1训练集-反应变量分布分析 25](#_Toc996)

[6.3.2训练集-解释变量分布分析 26](#_Toc25171)

[6.3.3训练集和测试集数据分布一致性判断-基于对抗验证 26](#_Toc31717)

[6.3.4训练集和测试集数据合并 27](#_Toc895)

[6.3.5对测试集和训练集进行分类-基于LightGBM算法模型 27](#_Toc22256)

[6.3.6分类效果评价-基于ROC曲线的AUC值 28](#_Toc10319)

[6.3.6数据集划分方法-基于LightGBM算法模型的概率结果 29](#_Toc29137)

[6.4数据可视化 30](#_Toc14943)

[6.4.1核密度图分布分析 30](#_Toc6056)

[6.4.2特征关系图分析 31](#_Toc10336)

[6.4.3小提琴图分析 32](#_Toc21554)

[6.4.4分类变量分析-用户话费敏感度 32](#_Toc28607)

[6.4.5连续变量分析-用户账单当月总费用 33](#_Toc21820)

[6.4.6相关性分析-用户近6个月平均消费值与信用分 34](#_Toc31943)

[6.4.7相关系数分析 35](#_Toc15574)

[6.5特征工程 36](#_Toc23688)

[6.5.1特征创造-基于聚类算法 36](#_Toc4611)

[6.5.2特征创造-基于实际场景 36](#_Toc22348)

[6.5.3特征过滤 37](#_Toc11252)

[6.5.4特征标准化 37](#_Toc32611)

[6.6 XGBoost算法模型构建 38](#_Toc5138)

[6.6.1划分数据集与测试集 38](#_Toc24451)

[6.6.2 模型初步建模 39](#_Toc32099)

[6.6.3 模型初步建模效果评估-LightGBM选取和随机选取测试集的比较 41](#_Toc3357)

[6.6.4 模型初步建模效果评估-样本数据量对结果的影响 42](#_Toc9230)

[6.6.5使用Embedded嵌入法确定threshold参数 43](#_Toc17998)

[6.6.6构建学习曲线调节参数 45](#_Toc15882)

[6.6.6.1 n\_estimators参数 45](#_Toc16464)

[6.6.6.2 subsample参数 47](#_Toc11838)

[6.6.6.3 learning\_rate参数 49](#_Toc12963)

[6.6.6.4 booster参数 51](#_Toc30104)

[6.6.6.5 objective参数 52](#_Toc19520)

[6.6.6.6 max\_depth参数 53](#_Toc31204)

[6.6.6.7 min\_samples\_leaf参数 55](#_Toc27535)

[6.6.6.8 min\_samples\_split参数 57](#_Toc14453)

[6.6.6.9 gamma参数 59](#_Toc30467)

[6.6.6.10 min\_child\_weight参数 61](#_Toc27342)

[6.6.6.11 colsample\_bytree参数 63](#_Toc25467)

[6.6.7学习曲线调参效果分析 65](#_Toc5624)

[6.6.8随机搜索优化参数-Randomized SearchCV 66](#_Toc7909)

[6.6.9网格搜索优化参数-Grid SearchCV 67](#_Toc296)

[6.6.10 网络调参效果分析 68](#_Toc2223)

[6.7模型评价与测试-基于MAE的结果 68](#_Toc14042)

[七、总结 70](#_Toc12042)

# 一、项目背景

随着社会信用体系建设的深入推进, 社会信用标准建设飞速发展，相关的标准相继发布，包括信用服务标准、信用数据釆集和服务标准、信用修复标准、城市信用标准、行业信用标准等在内的多层次标准体系亟待出台，社会信用标准体系有望快速推进。社会各行业信用服务机构深度参与广告、政务、涉金融、共享单车、旅游、重大投资项目、教育、环保以及社会信用体系建设，社会信用体系建设是个系统工程，通讯运营商作为社会企业中不可缺少的部分，同样需要打造企业信用评分体系，助推整个社会的信用体系升级。同时国家也鼓励推进第三方信用服务机构与政府数据交换，以增强政府公共信用信息中心的核心竞争力。现在通过使用人工智能技术，它将会从日常生活、学校、工作、休闲度假等多个维度来评定个人的信用度。每个人的社会活动将会被记录，例如如果你与一个人深度接触后再没有了联系，智能系统会认为你们双方兴趣爱好或者价值观不同，但是如果一个人深度接触过很多人之后，这些人后来都远离了他，这时智能系统会认为这个人的信誉很差。同样的道理，无论在学校和工作中任何的社交往来都将成为评定参考标准之一。**在本项目中我首先对于数据预处理尝试了一些性能优化，包括减小数据读取时的内存开销和使用dask框架加速运算。在数据预处理中我使用了正则表达式作为异常值检验的方式，除此之外我还建立了一个离群点检测模型用来识别离群点。在数据预处理之前我基于对抗验证的原理，使用LightGBM算法对于训练集和案例给的用来预测的预测集进行了数据一致性分布分析，并基于分布分析的结果构造了和预测集相似程度最高的测试集用来作为调参的数据集，而不采用交叉验证的方式。最后我基于有标签的用户信用智能评分的数据，进行了特征工程实现了特征创造，最后使用XGBoost算法构建消费者行为的评价模型，对他们的身份特征、消费能力、人脉关系、位置轨迹、应用行为偏好等特征与他们信用智能评分的分数的关联性进行预测。**

# 二、数据集说明

## 2.1数据集名称

消费者人群画像-信用智能评分数据集

## 2.2数据集来源

https://www.datafountain.cn/competitions/337/datasets

## 2.3数据集介绍

数据的原始来源是中国移动福建公司提供2018年某月份的样本数据（脱敏），包括客户的各类通信支出、欠费情况、出行情况、消费场所、社交、个人兴趣等丰富的多维度数据。该数据集包含了2个csv文件，train\_dataset.csv文件和test\_dataset.csv文件。其中train\_dataset.csv文件是用来给我们提供训练机器学习算法模型的原始数据文件，包含了某月中各个用户的上网行为信息及其信用评价的分数。数据集大小为5.43MB。整个数据集有50000条记录，用户编码，用户实名制是否通过核实，用户年龄，是否大学生落户，是否黑名单落户，是否4G不健康客户，用户网龄，用户最近一次缴费距今时长，缴费用户最近一次缴费金额，用户近6个月平均消费话费，用户账单当月总费用，用户当月账户余额，缴费用户当前是否欠费缴费，用户话费敏感度等30个特征，其中信用分作为反应变量，其他29个特征作为解释变量，在解释特征中有一个用来标识不同用户上网行为记录的编码。每条记录都由这30个字段组成，代表着一名用户在该月的行为特征及其获得的信用分数。

## 2.4数据集字段说明

数据集的字段说明如下表所示。

|  |  |  |
| --- | --- | --- |
| **序号** | **字段名** | **含义** |
| 1 | 用户编码 | 数值 唯一性 |
| 2 | 用户实名制是否通过核实 | 1为是0为否 |
| 3 | 用户年龄 | 数值 |
| 4 | 是否大学生客户 | 1为是0为否 |
| 5 | 是否黑名单客户 | 1为是0为否 |
| 6 | 是否4G不健康客户 | 1为是0为否 |
| 7 | 用户网龄（月） | 数值 |
| 8 | 用户最近一次缴费距今时长（月） | 数值 |
| 9 | 缴费用户最近一次缴费金额（元） | 数值 |
| 10 | 用户近6个月平均消费话费（元） | 数值 |
| 11 | 用户账单当月总费用（元） | 数值 |
| 12 | 用户当月账户余额（元） | 数值 |
| 13 | 缴费用户当前是否欠费缴费 | 1为是0为否 |
| 14 | 用户话费敏感度 | 用户话费敏感度一级表示敏感等级最大。  根据极值计算法、叶指标权重后得出的结果，根据规则，生成敏感度用户的敏感级别：  先将敏感度用户按中间分值按降序进行排序，前5%的用户对应的敏感级别为一级：  接下来的15%的用户对应的敏感级别为二级；  接下来的15%的用户对应的敏感级别为三级；  接下来的25%的用户对应的敏感级别为四级；  最后40%的用户对应的敏感度级别为五级。 |
| 15 | 当月通话交往圈人数 | 数值 |
| 16 | 是否经常逛商场的人 | 1为是0为否 |
| 17 | 近三个月月均商场出现次数 | 数值 |
| 18 | 当月是否逛过福州仓山万达 | 1为是0为否 |
| 19 | 当月是否到过福州山姆会员店 | 1为是0为否 |
| 20 | 当月是否看电影 | 1为是0为否 |
| 21 | 当月是否景点游览 | 1为是0为否 |
| 22 | 当月是否体育场馆消费 | 1为是0为否 |
| 23 | 当月网购类应用使用次数 | 数值 |
| 24 | 当月物流快递类应用使用次数 | 数值 |
| 25 | 当月金融理财类应用使用总次数 | 数值 |
| 26 | 当月视频播放类应用使用次数 | 数值 |
| 27 | 当月飞机类应用使用次数 | 数值 |
| 28 | 当月火车类应用使用次数 | 数值 |
| 29 | 当月旅游资讯类应用使用次数 | 数值 |
| 30 | 信用分 | 数值 |

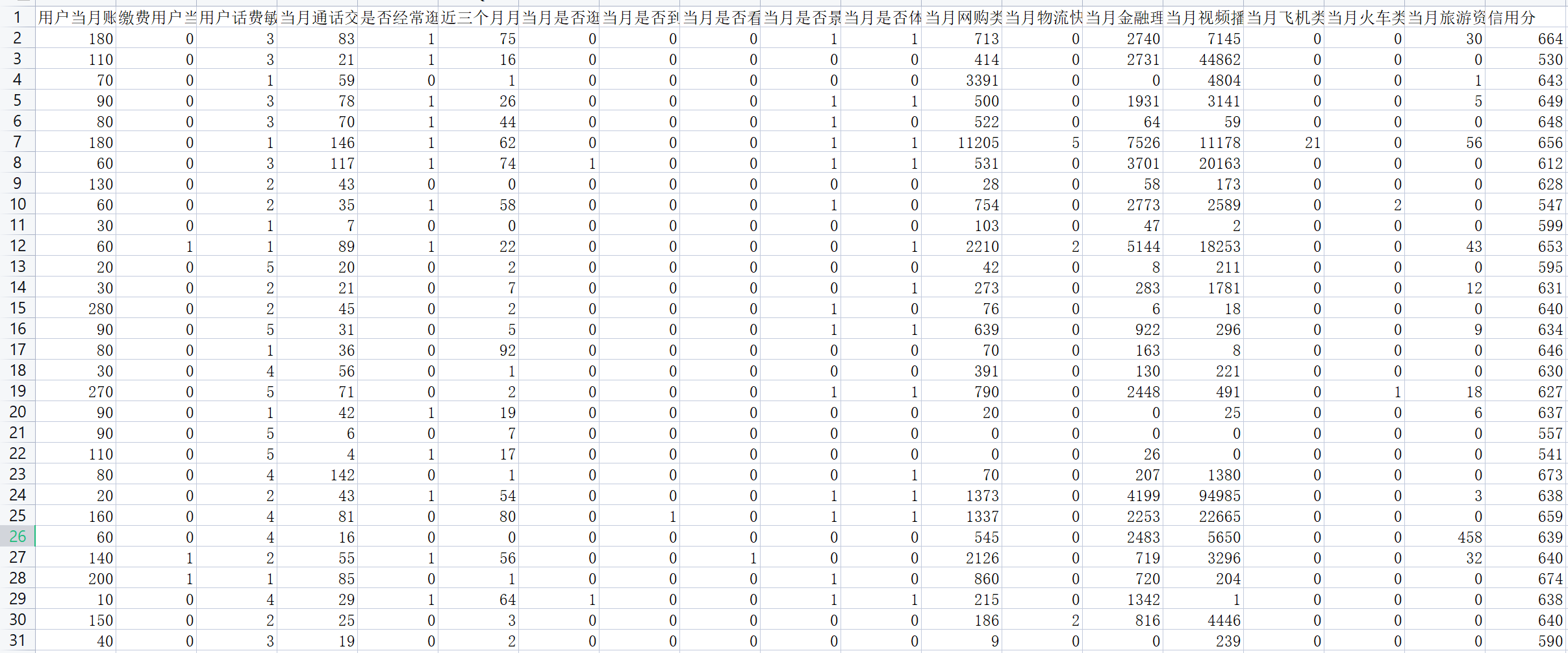
**表1 train\_data数据集各字段含义解释**

## 2.5数据集概览

为了方便查看数据集，先截取了数据集的一部份用来观察数据集的内容。截取train\_dataset.csv文件前50条记录如下图所示。



**图1 train\_dataset数据集概览一**



**图2 train\_dataset数据集概览二**

# 三、实验目的

使用Python数据分析的技术和机器学习的各种算法构建用户日常行为的评价模型。首先基于对抗验证的原理使用LightGBM算法进行数据集划分，然后再使用机器学习算法中的XGBoost算法完成用户行为的评价模型，并进行调参，获取最佳的模型。并且通过实验进一步加深对机器学习流程中以下步骤的理解。

* 学会一些机器学习中常见的性能优化技巧，包括**减小内存空间开销、多核cpu加速等**。
* 学会使用一些机器学习的开源框架进行机器学习任务，从而提高运算速度，例如**dask**框架。
* 明白交叉验证的底层原理，**学会根据项目的应用案例场景构建合适的模型评价方法，而不是单单套交叉验证**。
* 明白**对抗验证**的原理，能够使用合适的分类算法模型，如**LightGBM**算法实现对抗验证的步骤。
* 掌握数据集划分方式，学会根**据对抗验证中分类算法的分类结果对数据集进行划分。**
* 数据预处理中如何检测离群点，不单单是简单地观察箱型图和散点图上的数据，而是通过**构建异常检测模型来检测离群点**。
* 数据预处理中如何对缺失值，异常值，重复值，无用值进行针对性的处理。例如如何用**正则表达式进行检测复杂形式的字符串是否具有正确的形式**。
* 如何对复杂的数据进行可视化分析。例如包括**核密度图、小提琴图、热力图**等各种图形的画法**。**
* 如何进行特征工程，例如**特征创造**。
* 如何使用**Embedded嵌入法**进行特征选择。
* 如何构建**学习曲线**调节**XGBoost**算法的参数。
* 如何使用**网格搜索法**优化**XGBoost**算法模型。

# 四、实验环境

操作系统：Windows 10

编译环境：Jupyter Notebook 5.6.0

语言：Python 3.7.9

科学库：Numpy 1.19.2

Pandas 1.3.4

Matplotlib 2.2.3

Seaborn 0.9.0

Sklearn 1.0.1

Scipy 1.6.2

Xgboost 1.4.2

Dask 2021.10.0

# 实验流程设计

本实验共包括两个部分，第一部分为数据的初步探索，第二部分为用户日常上网行为基线的构建，第三部分位为用户上网行为评价模型的构建。具体内容如下：

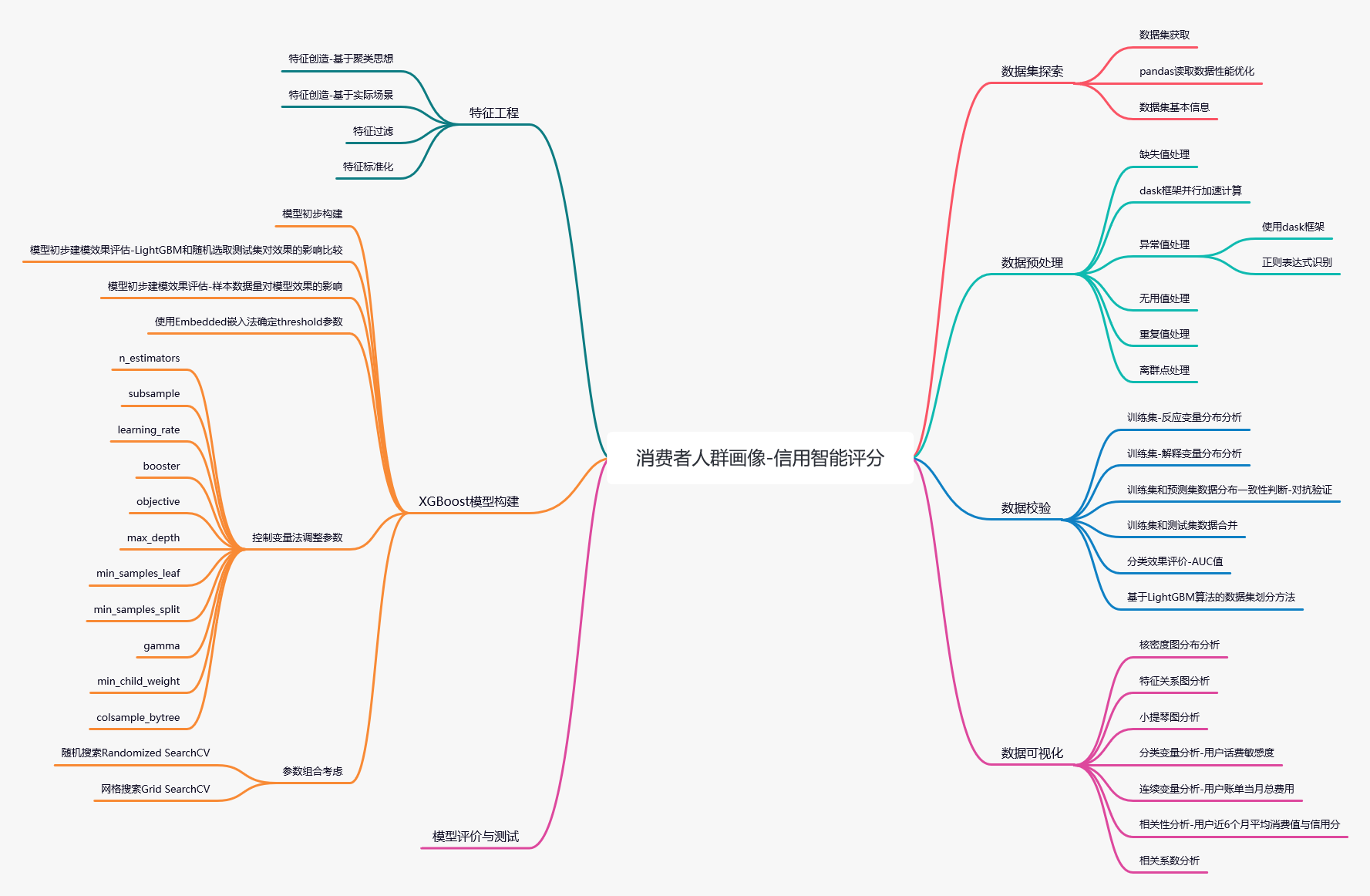
1. 数据的初步探索

* 数据集的初步探索
* 数据的预处理
* 数据的可视化分析

1. 决策树模型的构建

* 数据的特征工程
* 控制变量法调节**XGBoost**算法参数
* 网格搜索法优化**XGBoost**算法模型

该实验的具体的流程设计如以下思维导图所示：



**图3 项目思维导图**

# 六、实验具体步骤

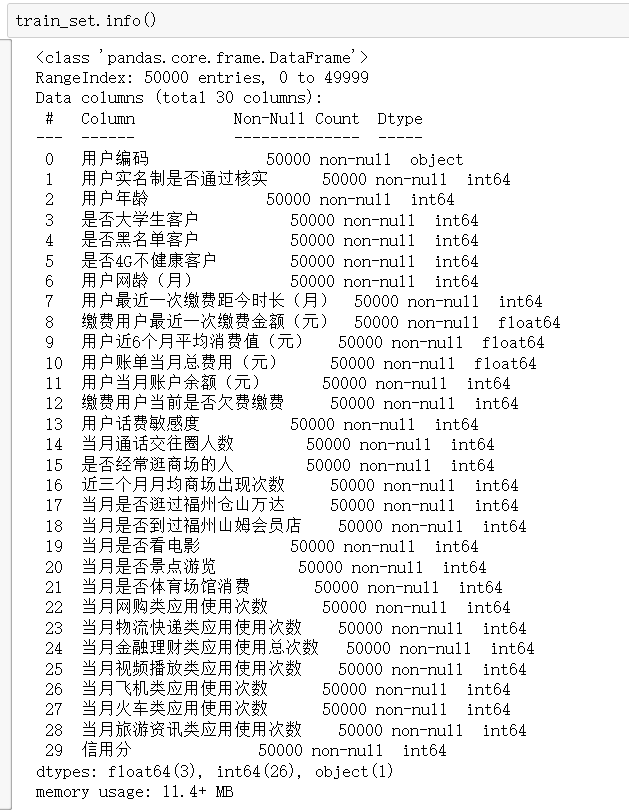
## 6.1数据集探索

### 6.1.1数据集获取

DataFountain是国内领先的数据科学及人工智能协同创新平台，提供AI 竞赛/大数据竞赛、人工智能数据集、开源分享社区、AI实验室等服务。是CCF大赛官方指定竞赛平台。我在DataFountain平台上寻找到用于用户上网异常行为预测的数据集UEBA异常行为分析数据集，数据集来源网址：https://www.datafountain.cn/competitions/337/datasets。由于元素数据集的文件是csv文件，文件名是train\_dataset.csv，另外由于该文件的编码是gbk，所以通过pandas库的read\_csv方法读取原始数据到train\_set中，编码参数使用read\_csv这个方法的默认参数能直接将数据读取进来。

1. train\_set = pd.read\_csv('train\_dataset.csv')

### 6.1.2 pandas读取数据性能优化

Pandas是进行机器学习算法的常见的库，我们使用Pandas可以进行快速读取数据、分析数据、构造特征。但Pandas在进行操作时候如果处理的好可以提高运行效率，降低代码运行时间和节省内存空间。

在上一小节中，我们不加处理的直接使用read\_csv方法读取了原始数据，读取进来的数据存进dataframe格式的数据。我们可以使用train\_set.info()方法来查看该dataframe的信息，我们可以发现该dataframe使用了11.4MB的内存，但是我们的原始数据集train\_dataset.csv总共才5.43MB。

**图4 train\_set信息**

我们从csv文件读取dataframe时，它文件的大小扩大了一倍，这个时候浪费了空间，这在工业界是不能接受的，一个10个TB的文件读进来需要耗费20个TB的存储空间。进一步观察train\_set的信息可以发现该dataframe各个字段的数据类型，如果该字段的数据是数值类型的，对应的数据类型不是int64就是float64。但是64位的精度远远大于我们获得的原始数据集中的数据的精度，我们实际上并不需要使用这么大的精度来表示我们数据集中的数据精度。所以对于类似0-1判断的特征字段，不需要使用64位的数据类型。只需要使用8位或者16位的数据类型就性了，这个时候这个dataframe就能省下大量无用的内存空间。

所以我们应该指定每个数值类型字段的数据类型。比如说用户网龄可以指定数据类型为int32，用户年龄为int16，是否黑名单客户为int8，是否4g不健康客户为int8，是否大学生客户int8，用户实名制是否通过核实为int8等。



**图5 优化过后的读取原始数据文件的代码**

Pandas默认情况下读取数据各个字段确定数据类型时不会替你优化内存开销。通过指定数据字段精度的方式能有效压缩消耗的内存空间的大小。



**图6 经过优化pandas操作后的读取进来的数据集信息**

通过比较图4和图6，我们可以看出指定字段数据精度后，数据集消耗的内存空间开销为5.1MB，已经比原始数据集train\_dataset.csv的大小相比没指定字段数据精度之前数据集消耗的内存空间开销11.4MB，数据集消耗的内存空间压缩了55.26%，这个提升是十分巨大的。对于十几MB的数据集，这个效果可能并不显著。但是如果数据集的大小达到GB级别时，没有压缩的数据集可能压根都读取不进来。而且通过压缩数据集的大小也能提升dataframe读取数据的速度。

### 6.1.3数据集基本信息

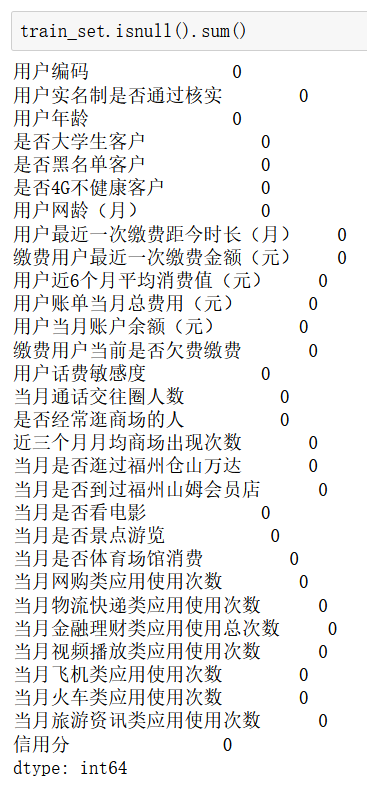
通过train\_set.shape查看数据集的大小：

IMG_256

**图7 train\_set形状**

train\_set数据集有528690行，10列。

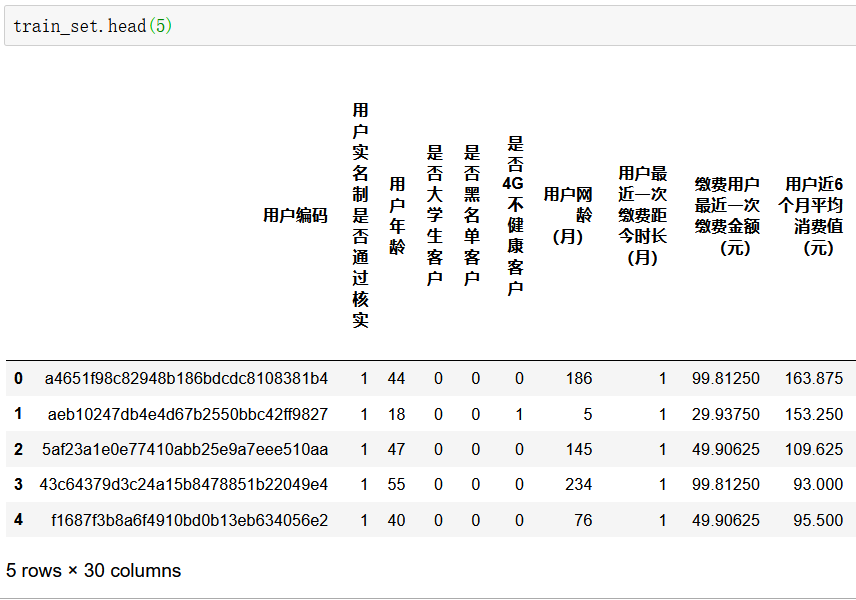
通过train\_set.isnull().sum()可以打印出数据集中每一个特征下缺失值的数量：



**图8 train\_set缺失值数量**

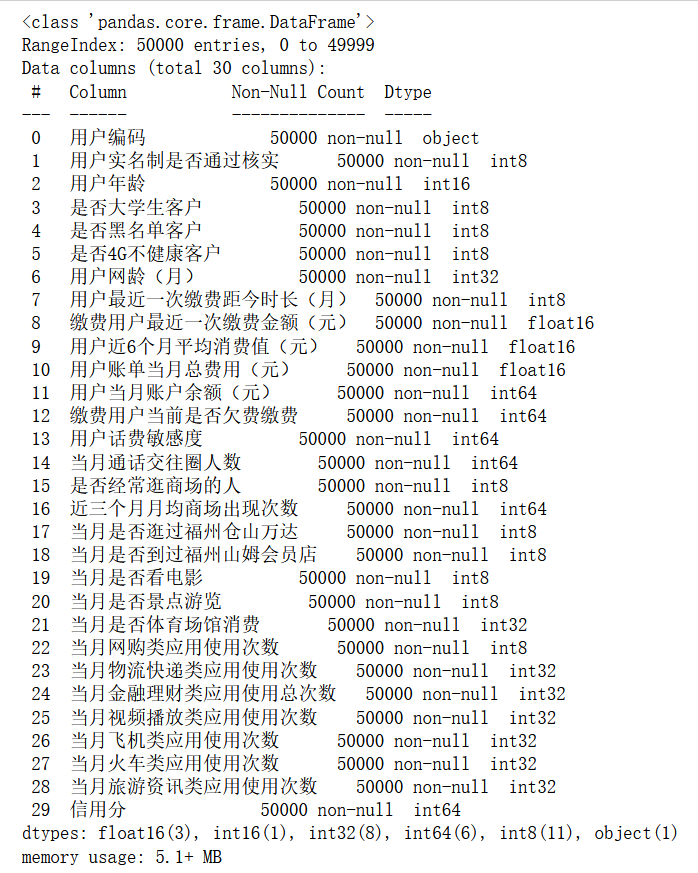
从结果中可以看出，数据集每一个特征下都不存在缺失值，所以我们在数据预处理时不需要对数据集中的缺失值进行处理。

train\_set.head()可以打印出来数据集前5行的数据，通过打印出来的数据我们可以看出，整个数据集有50000条记录，30个字段，其中信用分字段作为反应变量，其他29个特征字段作为解释变量，在解释特征字段中有一个用来标识不同用户上网行为记录的编码。每条记录都由这30个字段组成，代表着一名用户在该月的行为特征及其获得的信用分数。



**图9 train\_set前5行数据**

通过调用train\_set.info()，可以打印出来该每数据集每个特征下数据的类型。



**图10 train\_set的各个字段类型**

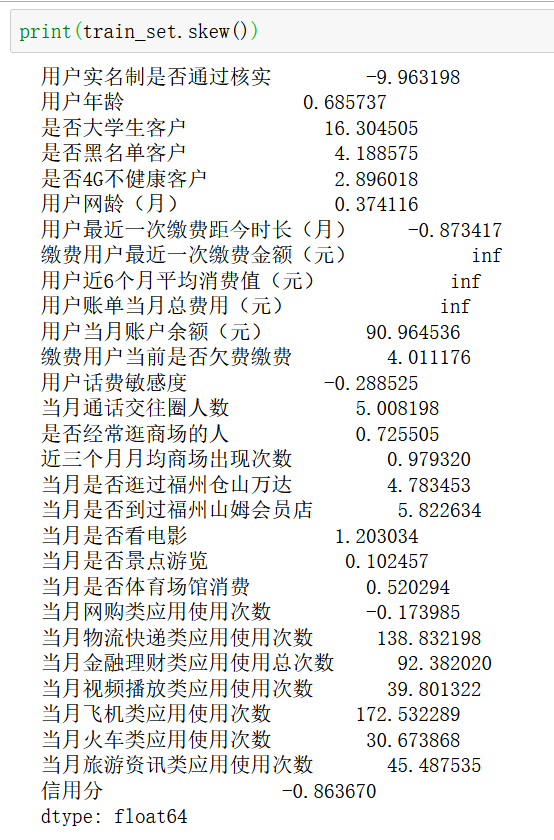
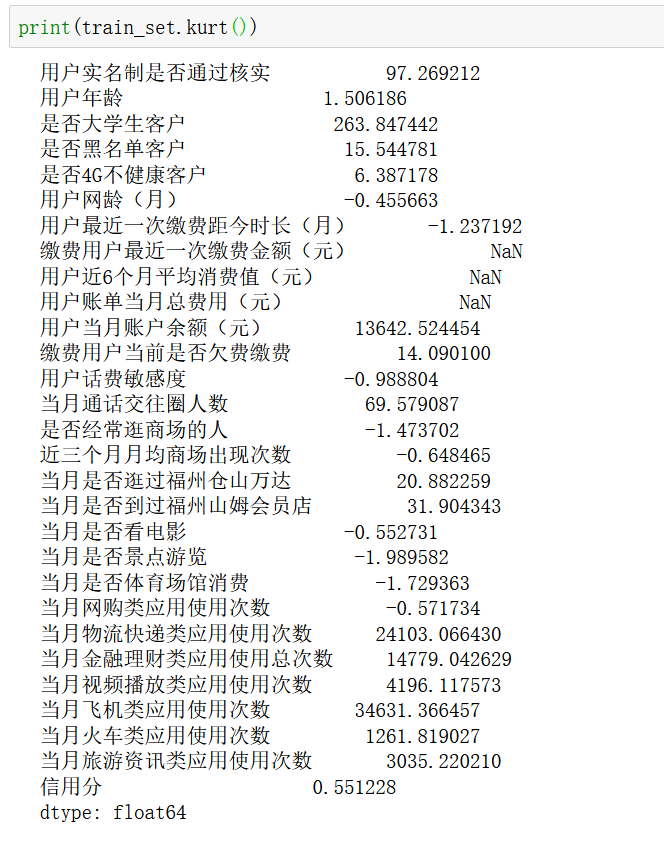
从结果中看出，有不少特征的数据类型是object，存在字符型的数据，所以我们需要在数据预处理时对这些字段的数据类型进行转换。

Pandas库中提供了skew函数和kurt函数可以用来查看原始数据的偏度和峰度。偏度Skewness是描述数据分布形态的统计量，其描述的是某总体取值分布的**对称性**，简单来说就是数据的不对称程度：

1. Skewness = 0 ，分布形态与正态分布偏度相同。
2. （2）Skewness > 0 ，正偏差数值较大，为正偏或右偏。长尾巴拖在右边，数据右端有较多的极端值。
3. （3）Skewness < 0 ，负偏差数值较大，为负偏或左偏。长尾巴拖在左边，数据左端有较多的极端值。
4. （4）数值的绝对值越大，表明数据分布越不对称，偏斜程度大。

峰度Kurtosis是描述某变量所有取值分布形态陡缓程度的统计量，简单来说就是数据分布顶的**尖锐程度：**

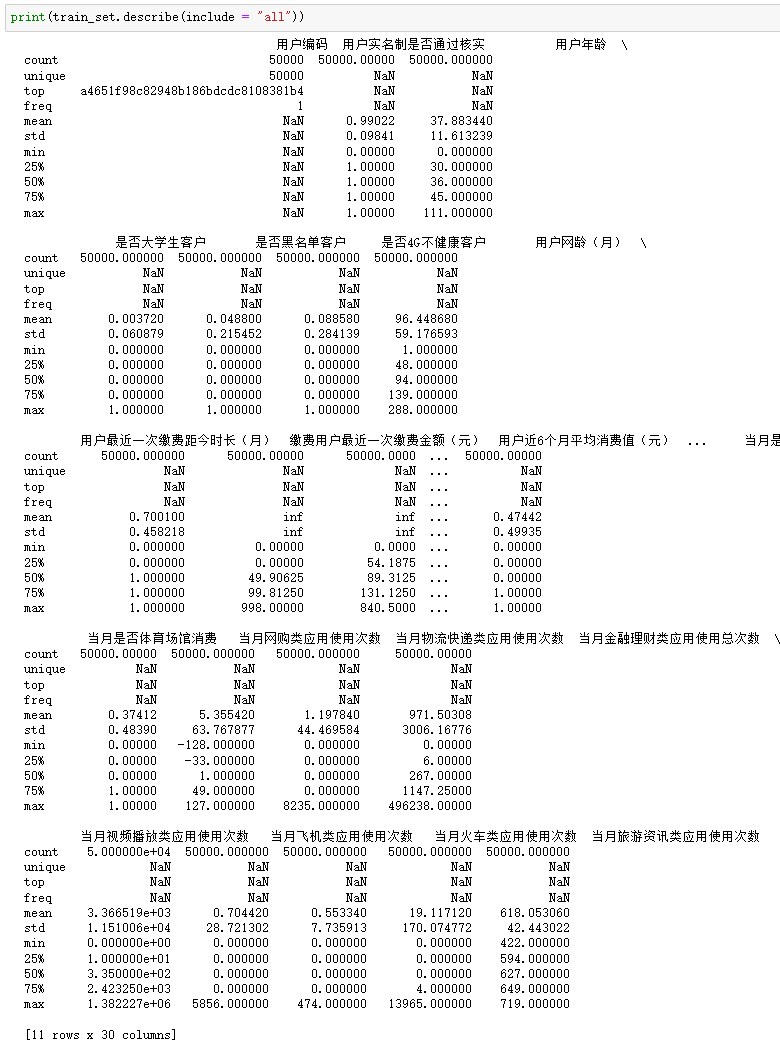
1. Kurtosis = 0 与正态分布的陡缓程度相同。
2. （2）Kurtosis > 0 比正态分布的高峰更加陡峭——尖顶峰
3. （3）Kurtosis < 0 比正态分布的高峰来得平台——平顶峰

**图11 train\_set的偏度 图12 train\_set的峰度**

由上面的结果我们可以看出信用分这个反应变量的字段数据的偏度是-0.863670，说明信用分在低分段会出现长尾巴，在低分段部份会出现很多数据偏低的极端值。相比于用户当月账户余额等字段来说，它的偏度的绝对值更接近0，说明它的形状更像正态分布。而它的峰度是0.551228，说明信用分的字段数据相比于正态分布更加尖锐。由于连续型变量类型的数据的峰度和偏度都过高，可能数据并不符合正态分布，所以应该要对连续型变量数据进行正态分布检验。

Pandas库中提供describe函数可以用来初步探寻数据集的特征，所以通过使用train\_set.describe()，我们可以得出来数据集的基本特征。



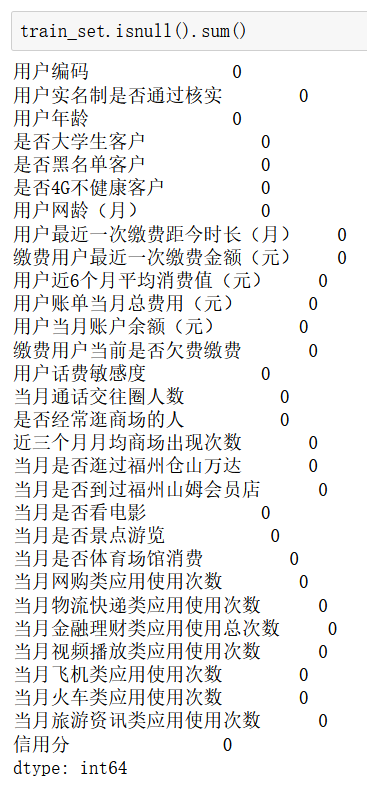
**图13 train\_set的描述性特征**

从结果中我们可以得到作为反应变量的用户信用智能评分的信用分数值是618.053060，最大值是719，最低值是422，50%的信用分数小于627。数据的分布符合用来训练机器学习模型的特征，不需要进行特征重采样，所以数据可以直接用来建立机器学习的算法模型。

## 6.2数据预处理

### 6.2.1缺失值处理

Pandas库中提供isnull函数可以用来生成所有数据的true/false矩阵。其中元素为空或者NA时对应元素就显示为true，否则显示为false。而sum函数可以用来获取所请求轴的值之和，所以通过调用train\_set.isnull().sum()，我们可以得到原始数据集中每一个特征下的缺失值的个数之和。



**图14 train\_set缺失值**

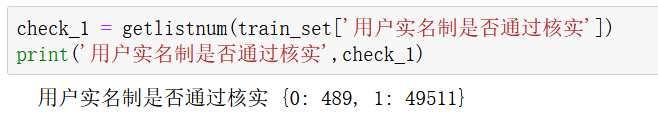
我们通过观察得到的结果可以知道原始数据集每一列都不存在缺失值，我们不需要进行缺失值的处理，填补或者删除缺失值。

**6.2.2统计数据分布情况**

由于大部分字段的数据都是离散型的。为了得到所有取值情况，把每一个特征的所有取值情况放进字典里。通过把每一个特征对应的所有取值放入一个list中，然后遍历list的每一个元素，每遍历一个元素就用字典自带的update函数，自动往字典里插入新的键值对或者更新已经有的键值对中元素的个数的数据。函数返回的字典就是该字段的各种不同的取值和对应的记录个数。

1. def getlistnum(li):
2. li = list(li)
3. set1 = set(li)
4. dict1 = {}
5. for item in set1:
6. dict1.update({item:li.count(item)})
7. return dict1

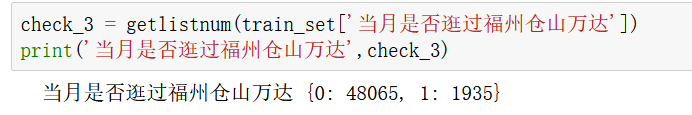
统计出来的用户实名制是否通过核实特征的取值和对应取值的个数如下图所示。



**图15 用户实名制是否通过核实字段数据分布**

通过该字段的数据分布可以看出它的分布是不均匀的，通过实名制检验的数据有接近5万条，但是没通过实名制检验的数据只有500条。数据分布接近100比1，这对于算法模型预测的效果会产生一定的影响，所以在特征工程时应该对它的结果进行处理。

统计出来的用户当月是否到过福州仓山万达店特征和对应取值的个数如下图所示。



**图16 用户是否到过福州仓山万达店字段数据分布**

通过该字段的数据分布可以看出它的分布是不均匀的，通过实名制检验的数据有接近5万条，但是没通过实名制检验的数据只有2000条。数据分布接近100比1，这对于算法模型预测的效果会产生一定的影响，所以在特征工程时应该对它的结果进行处理。

但是Pandas库中提供了现成的可以用来显示字段数据分布的函数value\_counts()，调用这个函数能直接观察数据分布。统计出来的是否经常逛商场的人特征和对应取值的个数如下图所示。



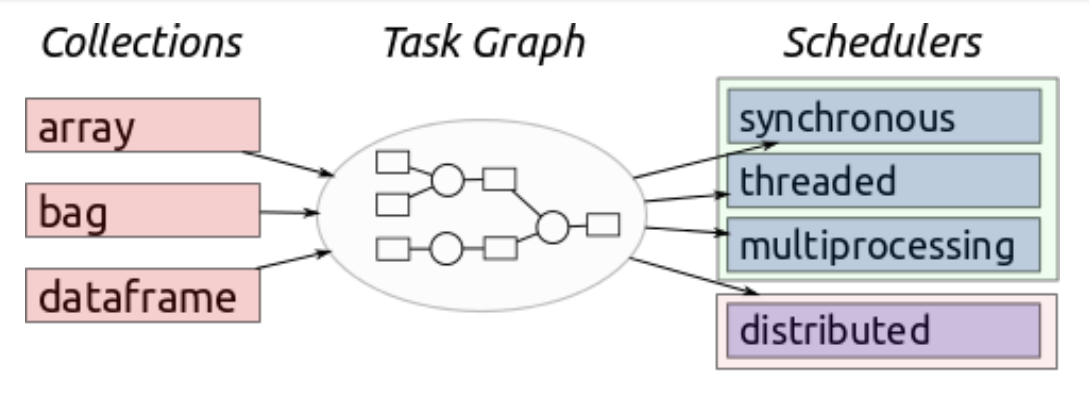
**图17 是否经常逛商场的人字段的数据分布**

该字段的数据分布接近2：1，是分布情况比较合理的字段特征。其他字段的数据分布没有明显的便于观察的现象，不方便观测是否出现异常值，而且字段不同的取值情况非常多，所以不在这里列出其他字段的数据分布情况。

### 6.2.2 dask框架并行加速计算

dask是一种分布式集群系统，支持pandas、numpy、sklearn、xgboost等机器学习的科学库的主流api，由于python GIL的限制，运用多线程时，在同一时刻，只能有一个线程在执行，导致了运用多线程并不会使程序运行速度明显加快，反而由于线程之间的数据传输实现效果并不好。通常在调用模型进行计算时，能控制n\_jobs参数为-1，就能使得在跑算法模型时成功调用所有cpu的核心。但是在对numpy和pandas这些科学库对象进行操作时不能控制调用所有cpu核心，这个时候使用dask框架就能加速计算，调用cpu的所有核心。

除此之外，dask具有延时计算的能力，它还能用于优化计算的动态任务调度和大数据集合，如并行数组、数据帧和列表，它能将NumPy、Pandas或Python迭代器等常见接口扩展到比内存大的环境或分布式环境，这些环境运行在动态任务调度程序之上。使得它能在本地处理它能处理的最大内存的能力。



**图18 dask算法框架原理**

### 6.2.3基于dask框架和正则表达式的异常值处理

由于部份字段的数据是形式比较复杂的数值型数据。例如用户年龄字段对应的是1-120的正整数，如果该字段的数据是特殊字符或者是字符串，或者该年龄字段的数据不是1-120的正整数，则该字段就是异常值。所以这个时候如果字段出现异常值时不好直接观测到，这个时候检测异常值更好地办法是通过使用正则表达式对train\_df（train\_df是由上一节对应的dask读取的数据）对应字段的字符串进行模式匹配，判断是否有不符合条件的字符串存在。

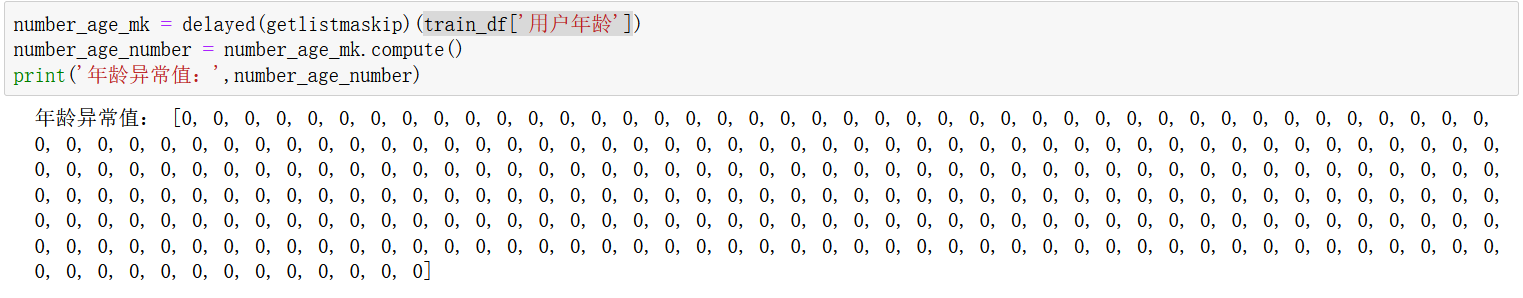
对年龄格式进行模式匹配的正则表达式为：

1. (^([1-9]\d?|1[01]\d|140)$)

通过函数getlistmask函数能够得到用户年龄字段索引不符合正则表达式的字符串取值，其中先通过调用re包的compiled方法将正则表达式字符串转化为对象再进行模式匹配，能够比直接通过正则表达式字符串进行模式匹配具有更高的效率。

1. def getlistmask(li):
2. L=[]
3. mask=0
4. rule\_age=r"(^([1-9]\d?|1[01]\d|140)$)"
5. compiled\_rule=re.compile(rule\_age)
6. lx = list(li)
7. for i in lx:
8. if compiled\_rule.match((str)(i)) is not None:
9. mask=mask
10. else:
11. Mask=mask+1
12. L.append(i)
13. return L

调用getlistmask函数发现该字段的异常值数量为290，异常的用户年龄字段数据为0。



**图19 用户年龄字段异常值**

在统计数据异常值的时候我使用了dask计算框架。首先是导入原始数据，dask支持替换主流的pandas的API接口。

1. import dask.dataframe as dd
2. train\_df = dd.read\_csv('train\_dataset.csv')

函数使用的是上面可以直接支持pandas的函数。下面的代码是在dask框架下执行函数的过程。delayed标签是将该计算任务放入计算图中，在dask框架下该任务就变成准备执行的状态，再调用compute命令，就能智能调度所有cpu的核心执行计算任务。

1. number\_age\_mk = delayed(getlistmaskip)(train\_df['用户年龄'])
2. number\_age\_number = number\_age\_mk.compute()
3. print('年龄异常值：',number\_age\_number)

由于年龄描述的用户的现实特征，如果改变了用户的现实特征不合理，而且年龄为0的用户数量不多。所以直接将年龄为0的用户记录全部去除就行了。使用loc函数，里面的条件是年龄字段数据大于0。

1. model\_train = model\_train.loc[model\_train['用户年龄']>0]
2. train\_df = train\_df.loc[train\_df['用户年龄']>0]

### 6.2.4无用值处理

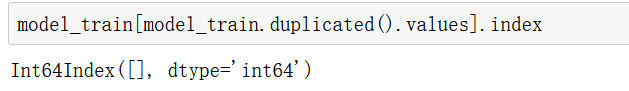
通过观察数据发现，用户编码特征只是用来唯一标识不同的用户的一个主键，只为了用来区分不同的用户，并不具有实际意义。通过del可以直接删去model\_train中的该字段的数据。

1. del model\_train['用户编码']
2. del train\_df['用户编码']

### 6.2.5重复值处理

在数据集中，由于把用来唯一标识不同用户行为信息的用户编码特征删除掉了，所以可能出现某些条记录中所有特征的数据完全一样的情况，这将导致在后续的数据分析和建模的过程中产生异常，影响数据分析结论的可靠性和正确性，为了规避由于存在重复值而产生的问题，在数据预处理阶段也需要对重复值进行处理。通过model\_train.duplicated()能找到当前model\_train所有每个字段都一样的记录。

1. train\_df[train\_df.duplicated().values].index



**图20 查询重复值位置**

由以上结果可知，数据集model\_train中不存在重复值，所以并不需要删除一些重复的记录。

### 6.2.6离群点处理-基于异常检测模型

根据下一节进行数据分布校验时可以知道一些连续型的特征字段的数据大致满足正态分布。在这种情况下如何检测连续型的特征字段的异常值，通常情况下我们是使用箱型图或者散点图观察数据点的分布情况，查看是否有一些数据点偏离集中的数据点。但是这种方法还是有一定缺陷的，在这里我建立了一种异常检测的数学模型。离群点的数值相对于正常点的数据的数值会发生巨大的变化，而数据数值发生了极大的变化可以使用那组数据的方差变大来表示。当我们原始数据分割成两组数据，此时两组数据的方差和应该是将原始数据任意划分成两组数据后的两组数据方差和中最小的。

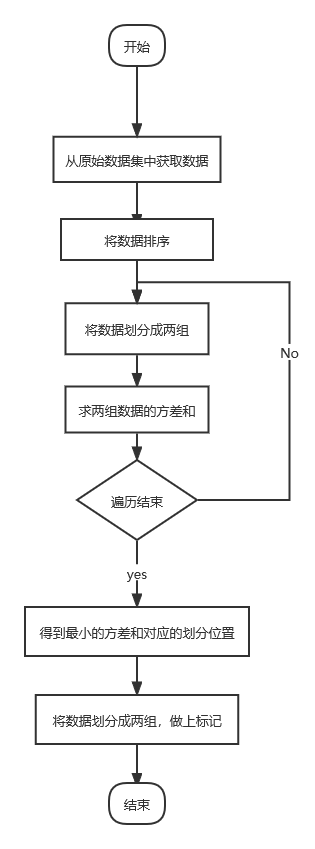
以检测指标偏大值为例，在检测指标异常数值的过程中，我们首先先得到原始数据，然后调用numpy库提供的**argsort**函数得到升序排序后的原数组索引所在的位置的数组。然后按照原数组将数据集拆分成两组，计算两组数据的方差和。

|  |  |
| --- | --- |
|  | (1) |

当两组数据的方差和取到最小值时说明此时两组数据内部的方差和是最小的，而两组数据之间的差异是非常大的。即偏高的那组数据是指标异常数值，偏低的那组数据是正常值。并在原始数据集里对相应的数据打上标签，即0或者1，1代表指标数据为异常值，0代表指标数据为正常值。

|  |  |
| --- | --- |
|  | (2) |

该异常检测模型的算法流程图如下所示：

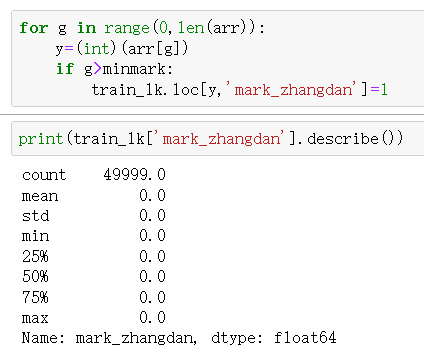


**图21 异常检测模型算法流程图**

进行异常检测算法的代码如下所示。对数据进行排序可以使用numpy库中提供的argsort方法，argsort方法能将原数组进行排序，并返回排序后的原数组元素对应的索引。而求方差能直接调用numpy库的std方法。

1. arr = np.argsort(radio)
2. for g in range(1,len(arrg)):
3. sk = np.std(arrg[arr[0:g]])+np.std(arrg[arr[g:len(arr)]])
4. if(min>=sk):
5. min = sk;
6. minmark = g
7. result1=[]

接下来我针对当月网购类应用使用次数这个特征进行离群点处理。处理结果如下所示。从结果我们可以知道该当月网购类应用使用次数并不存在离群点。所以我们不需要对该字段进行特殊处理。



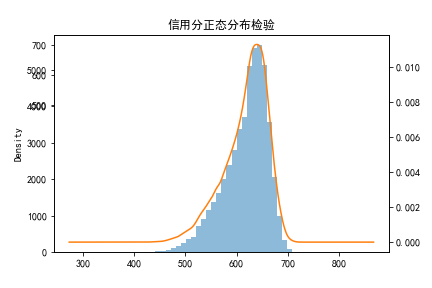
**图22 异常检测字段的结果**

## 6.3数据校验

### 6.3.1训练集-反应变量分布分析

通过观察数据信用分可能具有正态分布的特性，我对信用分的分布情况进行了正态分布的检验。通过代码作出信用分的正态分布曲线和对应的回归拟合的图像情况。通过以下代码可以做出信用分和置信度在0.5时，对应的正态分布拟合的曲线。这个分布较为合理，所以不需要对训练集数据进行重采样。

1. target.hist(bins=30,alpha = 0.5,ax = ax2)
2. target.plot(kind = 'kde', secondary\_y=True,ax = ax2)

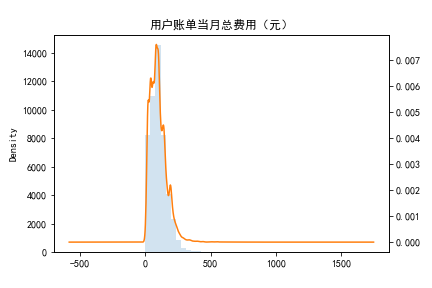
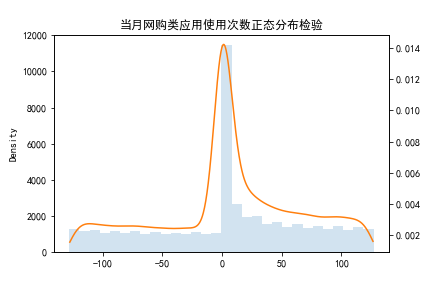
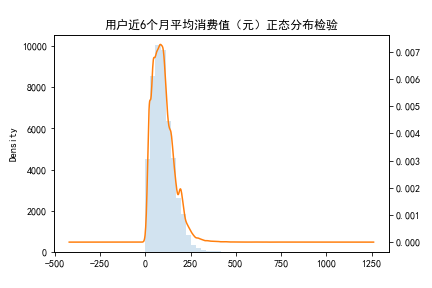


**图23 信用分正态分布检验图**

通过观察可以知道信用分数据在置信度为0.5的条件下近似满足正态分布。

### 6.3.2训练集-解释变量分布分析

我接下来对用户近6个月平均消费值，当月网购类应用使用次数，用户账单当月总费用等指标进行了正态分布分析。



**图24 解释变量正态分布检验图**

通过作出来的图，我们可以发现用户近6个月平均消费值和用户账单当月总费用所作出的条形图更接近于对应的正态分布曲线。而当月网购类应用使用次数的条形图低于对应的正态分布曲线图较多，说明它不能简单地看成正态分布。

### 6.3.3训练集和测试集数据分布一致性判断-基于对抗验证

工业界有一个大家公认的看法，“数据和特征决定了机器学习项目的上限，而算法只是尽可能地逼近这个上限”。在实战中，特征工程几乎需要一半以上的时间，是很重要的一个部分。缺失值处理、异常值处理、数据标准化、不平衡等问题是比较明显地问题，但是还有一个问题需要特别注意-数据一致性。在本节中我使用了对抗验证的方法去检验训练集和测试集的数据分布是否一致。

对抗验证并不是一种评估模型效果的方法，而是一种用来确认测试集和训练集的分布是否变化的方法。它的思路是：构建一个分类器去分类测试集和训练集，如果模型能清楚分类测试集和训练集，说明测试集和训练集存在明显区别，否则反之。具体步骤如下:

* 测试集和训练集合并，同时新增标签‘Is\_Test’去标记训练集样本为0，测试集样本为1。
* 构建分类器去训练混合后的数据集，可采用交叉验证的方式，拟合目标标签‘Is\_Test’。
* 输出交叉验证中最优的AUC分数。AUC越大(越接近1)，越说明训练集和测试集分布不一致。

### 6.3.4训练集和测试集数据合并

读取训练集和测试集的数据：

1. test\_dui = pd.read\_csv('test\_dataset.csv')
2. train\_dui = pd.read\_csv('train\_dataset.csv')

删除训练集里的标签字段：

1. del(train\_dui['信用分'])

新增是训练集还是测试集的标签：

1. test\_dui['Is\_Test']=1
2. train\_dui['Is\_Test']=0

最后使用dataframe的append方法将数据合并：

1. duikang = test\_dui.append(train\_dui)

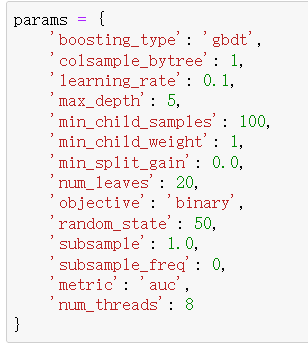
重置新的datframe的索引：

1. duikang=duikang.reset\_index()
2. del(duikang['index'])

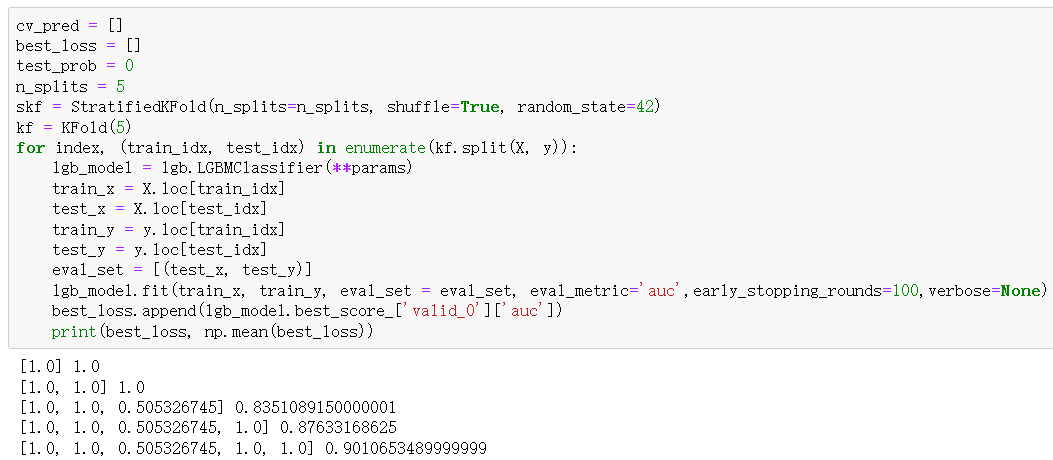
### 6.3.5对测试集和训练集进行分类-基于LightGBM算法模型

LightGBM和XGBoost一样是对GBDT的高效实现，原理上它和GBDT及XGBoost类似，都采用损失函数的负梯度作为当前决策树的残差近似值，去拟合新的决策树。

LightGBM的算法参数的选择如下所示：



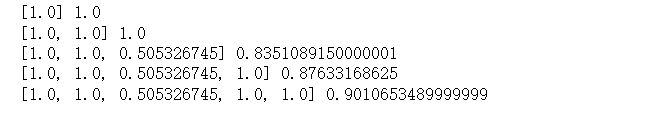
**图25 LightGBM算法参数**



**图26 LightGBM算法代码及执行结果**

### 6.3.6分类效果评价-基于ROC曲线的AUC值

如果我们用ROC曲线来评估分类器的分类质量，我们就可以通过计算AUC，即ROC曲线下的面积来评估分类器的质量，ROC的值越高说明分类器的效果越好。通常认为AUC值大于0.7时，LightGBM具有良好的分类效果，即训练集和测试集间的数据分布差异较大。



**图27 不同情况下的AUC取值**

由于在交叉验证折数确定的情况下，本问题中LightGBM算法得到的分类器的分类效果AUC均值的分类效果都大于0.7，说明本题提供的训练集和测试集数据分布情况有较大差别。

### 6.3.6数据集划分方法-基于LightGBM算法模型的概率结果

根据之前的结果我们可以分析出题目给的数据集的验证集和测试集的数据分布差距过大，如果直接随机选取数据进行交叉验证，在测试集上模型预测的效果将会过低，所以我们在跑模型算法之前应该要改变训练集和测试集的划分方法，不再使用交叉验证法划分测试集和训练集。当我们进行对抗验证时，模型预测样本是预测集的概率。概率越高，则说明和验证集越相似。所以我们选用训练集中和测试集样本越相似的样本作为模型挑参的测试集。

通过对抗验证中的模型，得到各个样本属于测试集的概率：

1. lgb\_model.fit(df\_adv.drop('Is\_Test', axis=1), df\_adv.loc[:, 'Is\_Test'])
2. preds\_adv = lgb\_model.predict\_proba(df\_adv.drop('Is\_Test', axis=1))[:, 1]

只需要训练样本的概率：

1. df\_train\_copy = train\_dui.copy()
2. df\_train\_copy['is\_test\_prob'] = preds\_adv[:len(train\_dui)]

根据概率排序：

1. df\_train\_copy = df\_train\_copy.sort\_values('is\_test\_prob').reset\_index(drop=True)

将概率最大的20%作为测试集：

1. df\_validation\_2 = df\_train\_copy.iloc[int(0.8 \* len(train\_dui)):, ]
2. df\_train\_2 = df\_train\_copy.iloc[:int(0.8 \* len(train\_dui)), ]

结果导出：

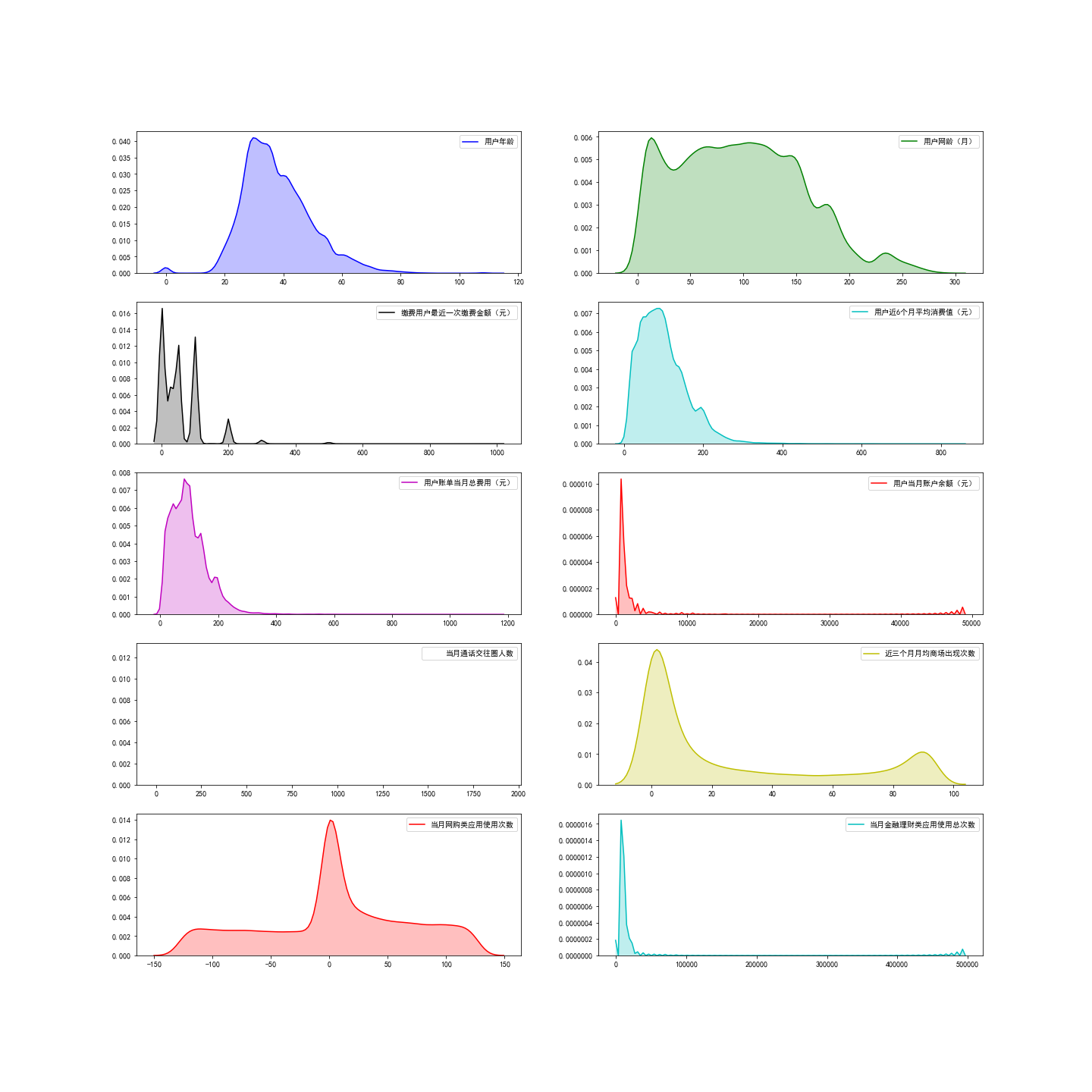
1. df\_validation\_2.to\_csv("model\_test.csv")
2. df\_train\_2.to\_csv("model\_train.csv")

## 6.4数据可视化

### 6.4.1核密度图分布分析

使用seaborn库提供的kdeplot方法能作出较为美观的核密度图。下面作出10个字段的核密度图进行分析。

1. sns.kdeplot(data=train\_set['用户年龄'], color='b', shade=True, ax=ax[0,0])



**图28 各连续变量核密度图分析**

我们知道频率直方图是统计学中表示频率分布的图形。在直角坐标系中，用横轴表示随机变量的取值，横轴上的每个小区间对应一个组的组距，作为小矩形的底边；纵轴表示频率与组距的比值，并用它作小矩形的高，所以一个小矩形的面积表示落入该小区间的频率。而核密度图是条形图取极限时候的情况，每个点的取值对应落入该点的概率。通过做出来的图我们可以发现，网购类应用使用次数最多的人都是0，金融理财类应用使用次数最多的人还是在接近0的位置，用户网龄在0-150个月内的用户的数量最高，而且各个月份的用户数量差不多。

### 6.4.2特征关系图分析

使用seaborn库提供的pairplot方法能作出较为美观的特征关系图。下面作出10个字段的核密度图进行分析。

1. axis=sns.pairplot(model\_train)



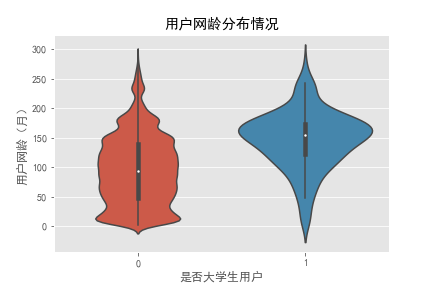
**图29 两两特征之间的关系图**

从结果中我们可以看出绝大部分变量之间的关系是较为离散的，没有很多关系是很密集的变量，这对于离群点的判断增加了难度。因为我们判断离群点的时候通常是参照成群密集点才能判断出离群孤立点。

### 6.4.3小提琴图分析

使用seaborn库提供的violinplot方法能作出较为美观的小提琴图。下面作出一些字段对比的小提琴图进行分析。

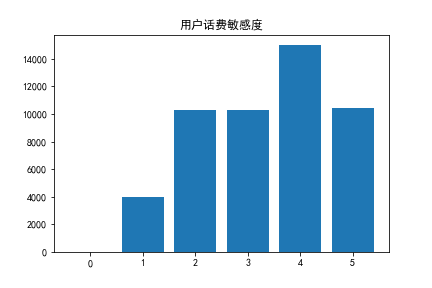
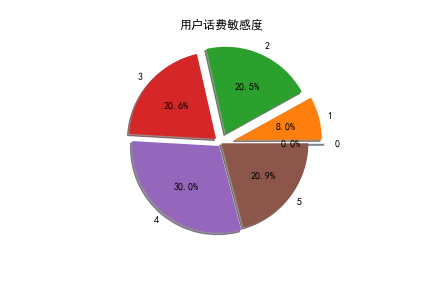
1. sns.violinplot(x = "是否大学生客户", y = "用户网龄（月）", data = train\_set)



**图30 用户网龄对比分布分析**

从结果中我们可以看出大学生用户的平均用户网龄会比非大学生用户平均网龄更高。因为通常大学生用户具有更高的知识文化水平，相对来讲会更有可能有更多时间在网络上冲浪。

### 6.4.4分类变量分析-用户话费敏感度

用户话费敏感度是一个是否重要的特征，下面对它进行分析。

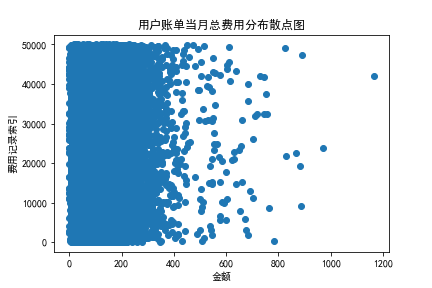
**图31 用户话费敏感度饼状图、柱状图**

从对于特征名称的解读中可以很明显的发现，用户对于话费的敏感度这个特征可以较为直观地反映出用户当月的收入情况。如果用户对于话费很敏感的话说明该用户该月的收入肯定不高，收入不高会导致该用户该月的信用评分也不高。

从统计结果可以看出来用户话费敏感度较高的人的数量少，敏感度一级的人只占了8%，相对来讲敏感度较低的人数量更多，敏感度四级和五级的人占了50.9%。说明大部分人的月收入还是比较高的，对应的信用分也会较高。

### 6.4.5连续变量分析-用户账单当月总费用

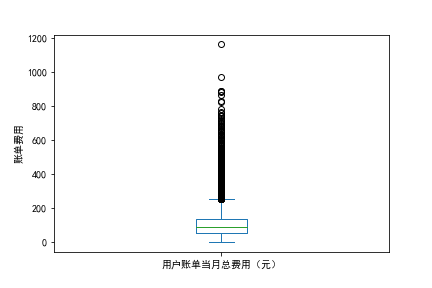
用户账单当月总费用也是一个重要的特征。不同的人具有不同的消费能力，往往消费能力高的用户的收入水平也较高，它具有比较高的可支配财富，所以他抵御风险的能力也较强。即用户账单当月总费用越高的人收入也应该越高，信用分也应该越高。



**图32 用户账单当月总费用分布散点图**

根据作出来的用户账单当月总费用分布散点图可以看出绝大部分的用户账单当月总费用金额在小于400元，用户账单当月总费用大于400元的用户少之又少不占大头。从结果可以看出来绝大部分的人收入还不是特别高，对应的信用分数也大概率偏低。

下面使用pandas包中提供的dataframe的plot方法能绘制出dataframe对应特征的箱型图。通过做出来的图像可以看出来该特征数据中大部分数据都分布在400以下，只有少部分数据在400以上。

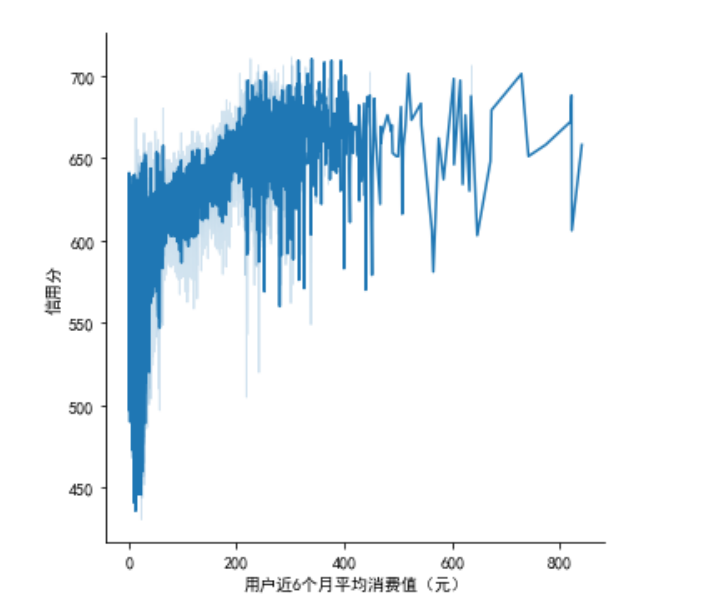


**图33 用户账单当月总费用分布箱型图**

### 6.4.6相关性分析-用户近6个月平均消费值与信用分

使用seaborn库提供的relplot方法能作出较为美观的折线图。relplot方法可以让你直接指定作折线图的自变量和因变量。下面作出用户近6个月平均消费值与信用分关系折线图进行分析。

1. sns.relplot(x="用户近6个月平均消费值（元）",y="信用分",kind="line",data=train\_set)



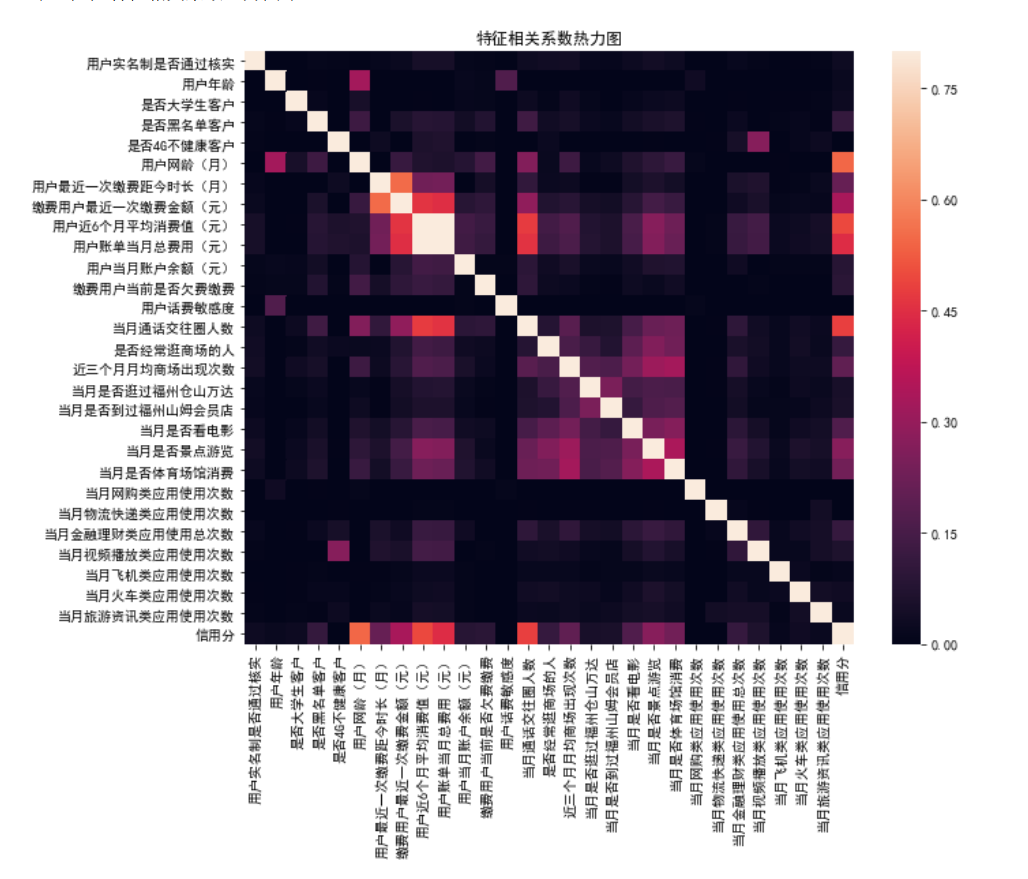
**图34 用户近6个月平均消费值折线图**

由折线图可以看出来用户消费额低的用户的信用分的下限可以相当低接近450，这说明低消费额的用户容易出现违约的情况，对应的信用分低。而用户消费额高的用户的信用分上限虽然和消费额低的用户差不多，但是信用分下限会比消费额低的用户高出很多，最低的也超过550了。这和实际情况相符，高消费额的用户的收入水平也高，抵御突然资金风险的能力也越强，所以信用分评级也会越高。

### 6.4.7相关系数分析

使用seaborn库提供的heatmap方法能作出较为美观的相关系数热力图。首先使用corr函数，能求出dataframe各列间的相关系数，然后使用sns.heatmap命令就作出了dataframe各列间的相关系数热力图。

1. correlation=lelitu.corr()
2. sns.heatmap(correlation,vmin=0,vmax=0.8)



**图35 训练集中各个特征之间相关系数的热力图**

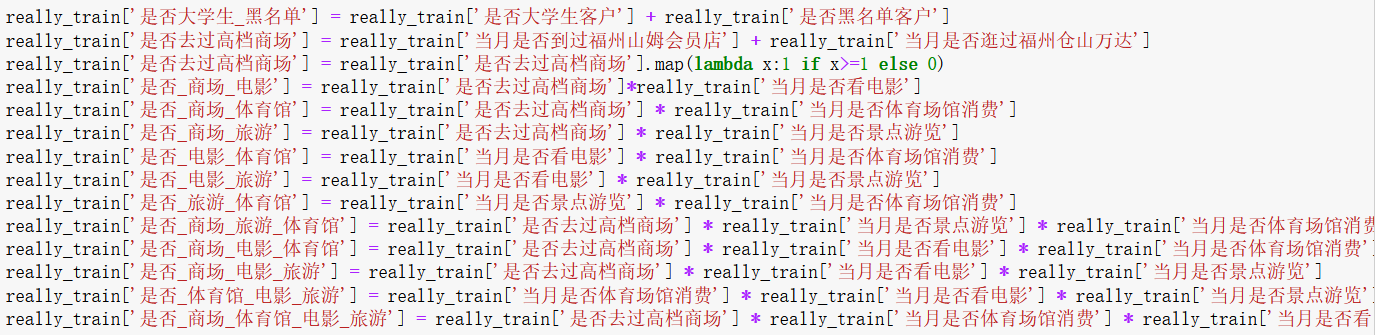
为了研究自变量和解释变量这些不同变量之间的相关性，用Person相关系数量化计算变量之间的相关性。从上面作出的Person相关系数热力图中可以看出信用分与用户网龄的正相关的关系最大，与当月通话交往圈人数，用户当月账单总费用，用户近6个月平均消费额，等字段具有比较强的相关性。而不同特征之间的相关性都是比较小的，各个特征之间不容易出现共线性的情况。

## 6.5特征工程

### 6.5.1特征创造-基于聚类算法

在我看来，聚类算法的本质在于依据样本的特征将具有相类似特征的样本都尽量划到同一类里，而同时也能保证同一类里的样本都具有了非常类似的特征。聚类算法可以说是相当于把一堆无监督的数据打上标签的算法。所以聚类算法可以用来特征升维或者特征降维。在这个问题中，我对0-1的是否型变量进行特征创造。在不同维度上的数据将对应的单维度的字段特征进行数据相乘，相乘后的数据如果是1，代表原来的两条记录是在这几个特征字段的高维特征上是一样的，即高维特征对应的低维特征同时具备。这样生成的高维特征数据能代表很多低维数据特征字段并不具备的特征。

进行特征创造的代码如下所示：



**图36 基于聚类算法实现的特征创造**

### 6.5.2特征创造-基于实际场景

在本问题中根据有些关于欠费和账单的特征能进行符合实际生活场景的特征创造，生成新的特征。

对于缴费用户最近一次缴费金额和用户近6个月平均消费值这两个特征。我的理解是对于用户来说，如果用户缴纳的费用比用户账单金额高，说明该用户收入水平比较高足以付清该用户的账单。所以进行特征创造的时候可以使用缴费用户最近一次缴费金额减去用户近6个月平均消费值作为一个新的特征字段用来代表用户的收入水平支付当月消费的账单的能力，这个值是正数代表用户该月收入足以偿还当月的账单，这个值如果是负数代表用户该月的收入不足以偿还当月消费的账单。而且这个只越大代表用户偿还账单的能力越强，可以反映出该用户的信用情况是越好的。

上面的特征创造我们是基于用户的收入和花销间的关系，除此之外我们可以考虑用户的花销本身。如果用户的花销越来越小，说明用户违约的可能性也越来越小。基于次我考虑使用用户账单当月总费用减去用户近6个月平均消费值作为一个新的特征字段用来代表用户花销的变化情况。这个值越小代表用户的花销的变化情况是越来越小的。

进行特征创造的代码如下所示：

1. really\_train['缴费金额是否能覆盖当月账单'] = really\_train['缴费用户最近一次缴费金额（元）'] - really\_train['用户近6个月平均消费值（元）']
2. really\_train['当月账单是否超过平均消费额'] = really\_train['用户账单当月总费用（元）'] - really\_train['用户近6个月平均消费值（元）']

### 6.5.3特征过滤

我们已经知道用户编码特征只是用来唯一标识不同的用户的一个主键，只用来区分不同的用户，并不具有实际意义。通过del可以直接删去really\_train中的该字段的数据。

1. del(really\_train['用户编码'])

### 6.5.4特征标准化

许多机器学习的算法中目标函数的基础都是假设所有的特征都是零均值并且具有同一阶数上的方差。如果某个特征的方差比其他特征大几个数量级，那么它就会在学习算法中占据主导位置，会影响学习器的效率，导致学习器并不能像我们说期望的那样，从其他特征中学习。特征标准化是让不同维度之间的特征在数值上有一定比较性，可以大大提高分类器的准确性。但是XGBoost模型是一个梯度集成算法，它本质上是由单棵树模型构成。而树模型是由决策点、策略点及结果构成的树形图，一般应用于序列决策中，通常以最大收益期望值或最低期望成本作为决策准则，通过图解方式求解在不同条件下各类方案的效益值，然后通过比较，做出决策。其中，树模型的优点有：一是浅层的树视觉上非常直观，而且容易解释。二是对数据的结构和分布不需作任何假设。三是可以捕捉住变量间的相互作用。所以在机器学习的过程中使用除了树之外的其他模型时往往需要对数据进行标准化。

在本次机器学习的项目中构建的是XGBoost模型，该模型对于数据没有任何先验要求，不需要数据集具有任何特殊的结构，所以不需要对该数据集进行数据标准化。

## 6.6 XGBoost算法模型构建

### 6.6.1划分数据集与测试集

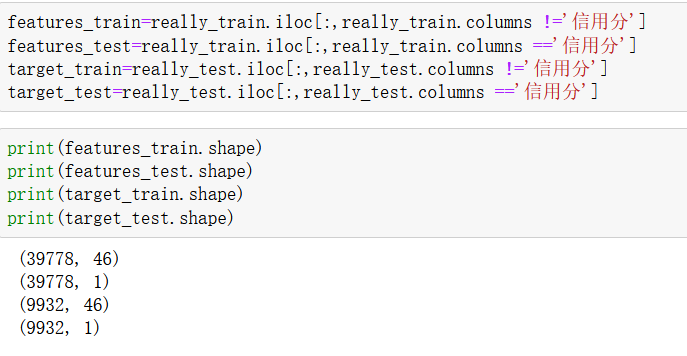
我在之前的6.3节中探讨了训练集和测试集的数据分布，通过使用以LightGBM模型为基础的对抗验证方法，我们发现训练集和测试集之间的数据分布差距过大。所以我们在模型构建时不考虑使用随机选取测试集的方法，而是使用以LightGBM模型结果为基础的选取测试集的方法，即选择和测试集相似程度最高的20%的训练集中的数据作为调整参数的测试集。然后将信用分特征字段作为反应变量，除了信用分特征字段的其他字段作为解释变量。但是之前在进行对抗验证实现训练集和测试集的数据检验时，对于提取出来的数据删除了训练集的反应字段特征信用分，所以应该在之前的代码位置把信用分字段拼接回去，从而使得数据集不会发生字段缺失的错误。

1. train\_dui = train\_dui.join(add\_train\_dui)

然后对数据集进行划分，划分成训练集反应变量，训练集解释变量，测试集反应变量，测试集解释变量。

1. features\_train=really\_train.iloc[:,really\_train.columns !='信用分']
2. features\_test=really\_train.iloc[:,really\_train.columns =='信用分']
3. target\_train=really\_test.iloc[:,really\_test.columns !='信用分']
4. features\_train,features\_test,target\_train,target\_test=train\_test\_split(features,t

划分后各个数据集的形状如下：



**图37 对抗验证数据集划分以后的形状**

为了实现使用对抗验证方法处理数据集和使用交叉验证的方法处理数据集时对模型处理效果的比较。所以我还对模型同样进行了交叉验证。将数据集中的20%作为测试集，80%作为训练集。通过调用sklearn库中的model\_selection提供的train\_test\_split函数来实现划分数据集的功能。此时所有的数据预处理已经进行完成，直接划分数据集就能保证测试集和训练集都经过了所有操作完全相同的数据预处理。将信用分字段作为目标变量，除了信用分字段的其他字段作为自变量。划分数据集的代码如下：

1. from sklearn.model\_selection import train\_test\_split
2. qfeatures = jiaocha.iloc[:,jiaocha.columns !='信用分']
3. qtarget = jiaocha.iloc[:,jiaocha.columns =='信用分']
4. qfeatures\_train,qfeatures\_test,qtarget\_train,qtarget\_test=train\_test\_split(qfeatures,qtarget,test\_size=0.20)print(features\_train.shape)

### 6.6.2 模型初步建模

在建立XGBoost模型之前，需要进行导库的操作。首先，我们这个数据集是回归的数据集，因此需要在安装完XGBoost库后导入XGBRegressor类。然后设置n\_estimators为100，其他参数都不设置使用默认的原始参数数据进行初步建模。

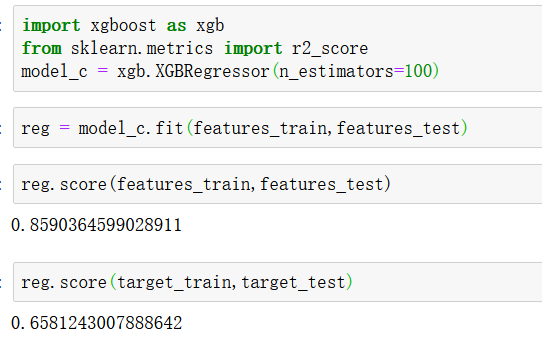
1. import xgboost as xgb
2. from sklearn.metrics import r2\_score
3. model\_c = xgb.XGBRegressor(n\_estimators=100)
4. reg = model\_c.fit(features\_train,features\_test)
5. reg.score(features\_train,features\_test)
6. reg.score(target\_train,target\_test)

XGBoost库提供了对应的方法plot\_importance可以让我们查看训练好的模型的特征的重要性。因为原始特征有40多个，在这里我选择了在图中只显示重要性前10的特征。



**图38 对抗验证初步模型建模时特征的重要性**

模型初步建模跑出的结果如下所示，在训练集上模型表现的效果比较好准确率为0.85，但是在测试集上模型的效果只有0.65，这说明模型针对预测集预测的效果可能会较差。



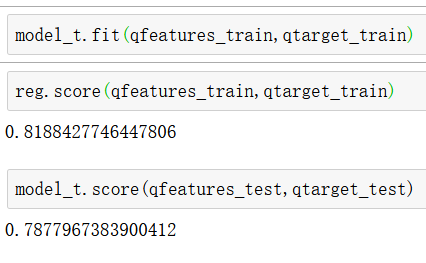
**图39 对抗验证模型初步构建训练集和测试集评分**

### 6.6.3 模型初步建模效果评估-LightGBM选取和随机选取测试集的比较

随机选取测试集并对测试集进行评价时只能评价模型针对一般情况下的泛化能力，并不能针对具体实际的案例场景，这种评价方式并不适合具体的案例场景。接下来我使用随机选取测试集的方式进行模型的初步构建并与之前进行对抗验证选取的测试集的方式进行对比，说明随机划分训练集和测试集针对模型的泛化能力的评价作用远远不如对抗验证划分测试集的效果显著。进行随机选取时建立的模型也是设置n\_estimators为100。

1. model\_t = xgb.XGBRegressor(n\_estimators=100)
2. model\_t.fit(qfeatures\_train,qtarget\_train)
3. once = cross\_val\_score(model\_t,qfeatures\_train,qtarget\_train,cv=5).mean()
4. model\_t.score(qfeatures\_test,qtarget\_test)

模型初步建模跑出的结果如下所示，在训练集上模型表现的效果一般准确率为0.81，但是在测试集上模型的效果居然比之前高很多有0.78。，说明模型针对不同案例场景的泛化能力是非常强悍的。这和之前使用对抗验证的方法对模型在训练集上的预测效果和在测试集上评估的模型泛化能力差别很大。对抗验证的模型在训练集上准确率有0.85，在测试集上只有0.65，说明模型的泛化能力是比较弱小的。这是因为本案例场景中的训练集和预测集数据分布差距过大。如果使用随机选取测试集后，这种数据分布的差距会被多次重复实验中和训练集分布一致的数据给稀释掉。

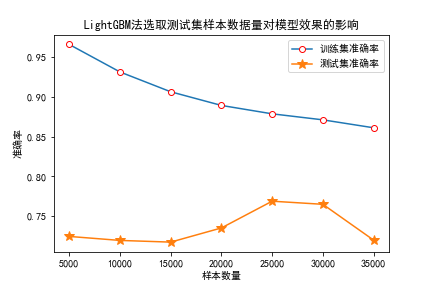


**图40 交叉验证模型初步构建训练集和测试集评分**

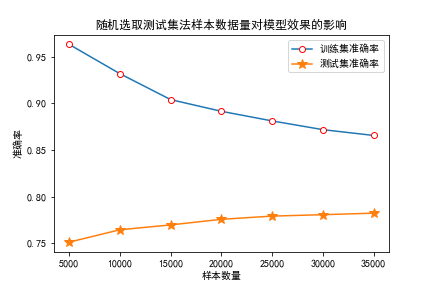
### 6.6.4 模型初步建模效果评估-样本数据量对结果的影响

在之前的节中我先初步建立了XGBoost模型，并探讨了随机选取测试集和使用LightGBM算法选取测试集对模型的预测效果和模型对于测试集上预测的泛化能力的影响。接下来我研究了训练集数据的选取对于模型效果的影响。我在训练集中选取了不同数量的训练数据来训练模型，观察模型的准确率是如何变化的。代码如下所示：

1. for i in range(5000,len(features\_train),5000):
2. samples.append(i)
3. feat\_train=features\_train.iloc[:int(i),]
4. feat\_test=features\_test.iloc[:int(i),]
5. reg = model\_c.fit(feat\_train,feat\_test)
6. xunlian.append(reg.score(feat\_train,feat\_test))
7. ceshi.append(reg.score(target\_train,target\_test))



**图41 LightGBM法构建训练集和测试集评分和数据集大小关系**

我们可以对使用LightGBM算法确定模型测试集的情况下的模型的训练集分数和测试集分数进行分析，发现随着样本数的提升，测试集的分数是先小幅度下降然后大幅度提升然后再下降，而训练集的分数在下降。测试集分数能大幅度提升说明了模型在随着样本数的提升，过拟合的程度在有所改善。测试集分数的下降可能是因为测试集数据分布特殊性导致的。

**图42 随机选取测试集法构建训练集和测试集评分和数据集大小关系**

我作出了随机选取测试集法测试集分数和数据集大小的关系图，在这个图中随着样本数量的增加模型在测试集上的准确率是一直上升的，而此时训练集上的准确率是一直在下降的，这证实随着样本数量增加模型过拟合的现象是一直在减轻的。但是这只是一个模型泛化能力粗放的衡量标准。因为具体数据中的数据分布不确定，所以才会在图39中在测试集里准确率呈现出先下降后上升再下降的变化趋势。

### 6.6.5使用Embedded嵌入法确定threshold参数

嵌入法是一种让算法自己决定使用哪些特征的方法，即特征选择和算法训练同时进行。在使用嵌入法时，我们先使用某些机器学习的算法和模型进行训练，得到各个特征的权值系数，根据权值系数从大到小选择特征。这些权值系数往往代表了特征对于模型的某种贡献或某种重要性，我们就可以基于这种贡献的评估，找出对模型建立最有用的特征。

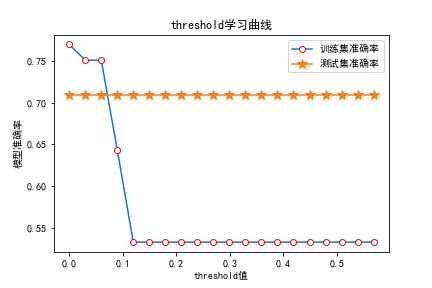
对于如何选取合适的特征这个问题，传统的过滤法只能通过机械地分析原始数据集的数据分布特征，如方差来淘汰一些特征所含信息熵小的字段。而通过调用SelectFromModel方法能自动地在算法训练过程中选择出贡献大于阈值的特征进行模型训练。然后我们可以通过绘制学习曲线的方式来获得threshold对应的最佳参数。

通过使用SelectFromModel方法，我们能够实现使用Embedded嵌入法进行特征选择的功能，将原始特征按照特征的重要性重新选择新的特征。接下来直接使用新生成的特征就能进行XGBoost算法。



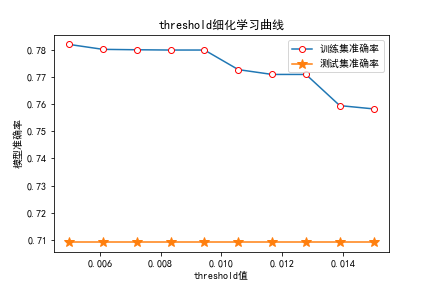
**图43 执行选择threshold参数的代码**

执行上述代码能够得到粗放的threshlod学习曲线图，如下图所示。但是由于模型准确率最高值位于取值区间的端点，所以应该在端点附近进一步作出细化的threshlod学习曲线图。



**图44 threshold学习曲线**

将上述学习曲线的范围缩小，作出细化的threshold学习曲线图如下所示。



**图45 threshold细化学习曲线**

通过上图绘制出的threshold的细化的学习曲线可以发现threshold的最佳取值应该是为0，即不进行特征选择的时候模型的预测效果是最好的。根据这个结果我们可以知道不应该使用Embedded进行特征选择等其他降维方法，因为使用降维方法后模型在训练集和预测集上的准确性反而会下降，每个特征都对于结果来讲是有意义的。

### 6.6.6构建学习曲线调节参数

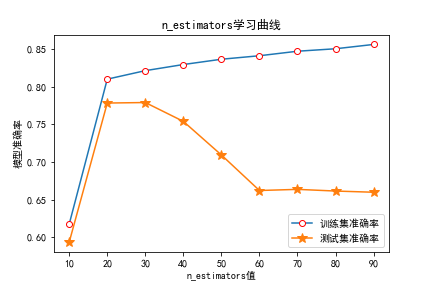
XGBoost本身的核心是基于梯度提升树实现的集成算法，整体来说可以有三个核心部分：集成算法本身，用于集成的弱评估器，以及应用中的其他过程。所以我们进行调参时应该也要关注这几方面。集成算法本身应该关注弱评估器的数量n\_estimators、控制随机抽样的subsample、控制迭代数率的learning\_rate等参数，而用于集成的弱评估器应该关注选择弱评估器的booster、目标函数objective、单棵树的max\_depth、min\_child\_weight、colsample\_bytree、控制正则化参数的gamma等。

### 6.6.6.1 n\_estimators参数

n\_estimators参数代表集成算法中弱评估器的数量，弱评估器的数量越多用来预测的效果就会越好。不过n\_estimators参数设的太大，模型的学习能力会更强，这容易导致过拟合现象的发生，n\_estimators参数设的太小，模型的学习能力较弱，容易导致欠拟合现象的发生。我们可以通过绘制学习曲线的方式来获得n\_estimators对应的最佳参数。对n\_estimators参数进行10到90，以10为间隔的方式进行初步调参的代码如下所示。其中随机种子参数设置为100。

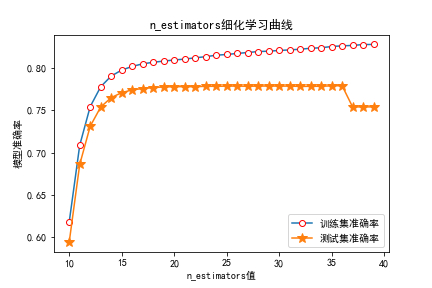


**图46 n\_estimators调参代码**

n\_estimators参数初步调参绘制出来的学习曲线如下所示。

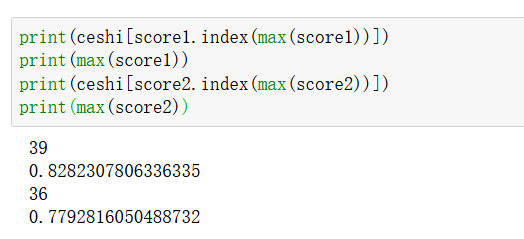
**图47 n\_estimators初步调参学习曲线**

由图可知最佳的n\_estimators取值应该在20附近。进下来对n\_estimators参数进行10到20，以1为间隔的方式进行详细调参。



**图48 n\_estimators详细调参学习曲线**

n\_estimators详细调参的结果如下所示。



**图49 n\_estimators详细调参结果**

通过上图绘制出的n\_estimators详细学习曲线和结果可以得到n\_estimators最佳的取值是36。一方面n\_estimators超过36以后模型的准确率在测试集上由断崖式下降。另一方面n\_estimators不到36以前模型的准确率都在不断提升，而由于弱评估器的增加使得计算成本增加的速度不是特别快。所以n\_estimators的最佳取值是36。

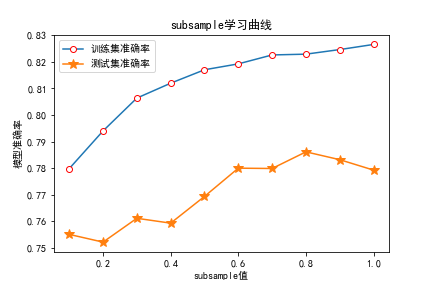
### 6.6.6.2 subsample参数

subsample参数代表随机抽样时抽取样本的比例。对模型来说，数据量越少模型学习越容易，学到的规则也会越具体越不适用于测试样本。所以subsample参数通常是在样本量本身很大的时候来调整和使用。我们可以通过绘制学习曲线的方式来获得subsample对应的最佳参数。对subsample参数进行0.1到1，以0.1为间隔的方式进行初步调参的代码如下所示。其中随机种子参数设置为100，n\_estimators设置为36。



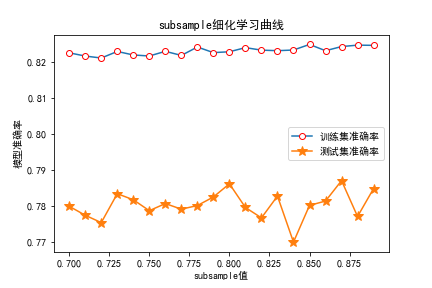
**图50 subample调参代码**

subsample参数初步调参绘制出来的学习曲线如下所示。



**图51 subample初步调参学习曲线**

由图可知最佳的subsample取值应该在0.8附近。进下来对subsample参数进行0.7到0.9，以0.01为间隔的方式进行详细调参。



**图52 subample详细调参学习曲线**

subsample详细调参的结果如下所示。



**图53 subample详细调参结果**

通过上图绘制出的subsample详细学习曲线和结果可以得到subsample最佳的取值是0.87。

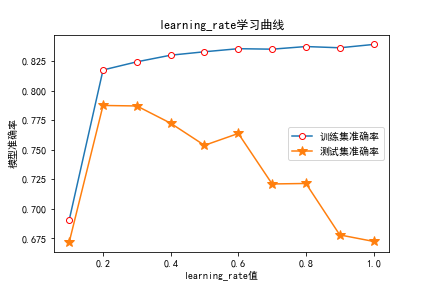
### 6.6.6.3 learning\_rate参数

learing\_rate参数代表集成算法中的学习率，又称为步长。以控制迭代速率，常用于防止过拟合。和逻辑回归类似，集成算法中的学习率越大，迭代的速度越快，算法的极限很快被达到，有可能无法收敛到真正的最佳值。集成算法中的学习率越小，越有可能找到更精确的最佳值。但是更多的空间被留给了后面建立的弱评估器，但迭代速度会比较缓慢。对learing\_rate参数进行0.1到1，以0.1为间隔的方式进行初步调参的代码如下所示。其中随机种子参数设置为100，n\_estimators设置为36，subsample设置为0.87。



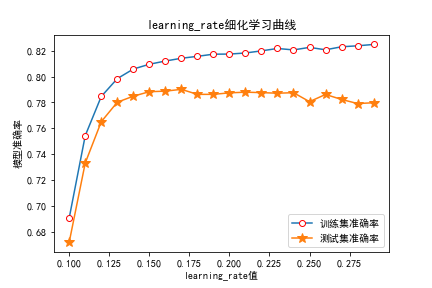
**图54 learning\_rate调参代码**

learing\_rate参数初步调参绘制出来的学习曲线如下所示。



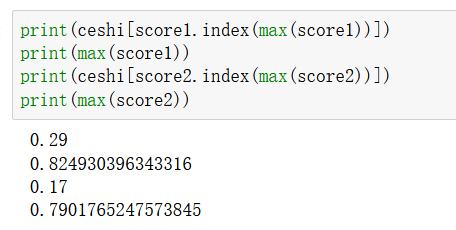
**图55 learning\_rate初步调参学习曲线**

由图可知最佳的learing\_rate取值应该在0.2附近。进下来对learing\_rate参数进行0.1到0.3，以0.01为间隔的方式进行详细调参。



**图56 learning\_rate详细调参学习曲线**

learing\_rate详细调参的结果如下所示：

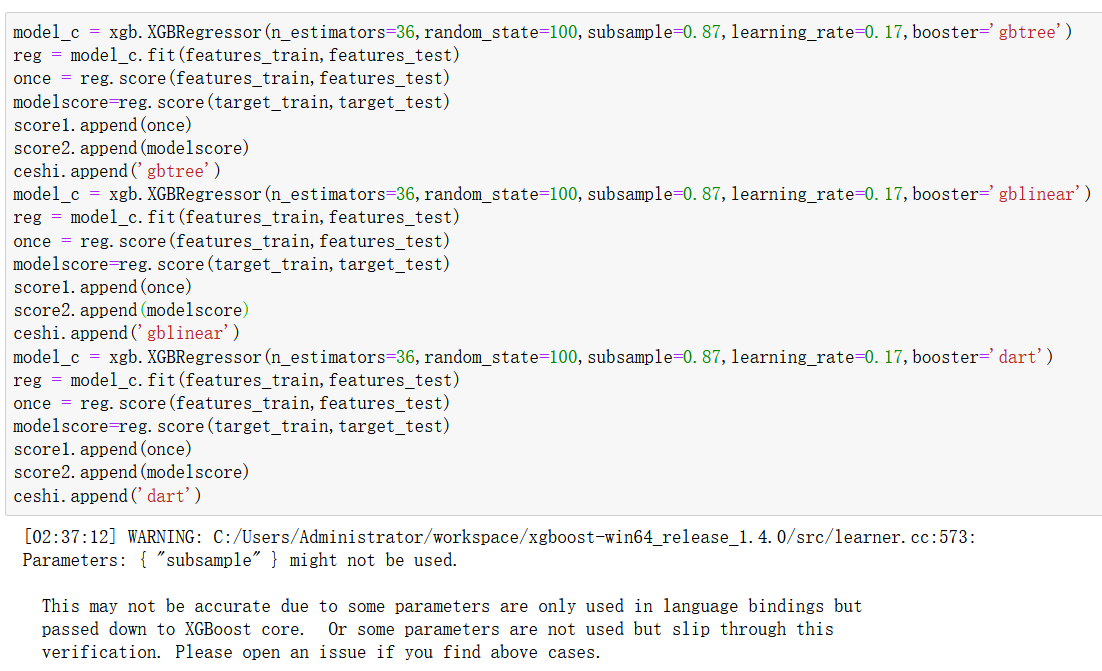


**图57 learning\_rate详细调参的结果**

通过上图绘制出的learning\_rate详细学习曲线和结果可以得到learning\_rate最佳的取值是0.17。

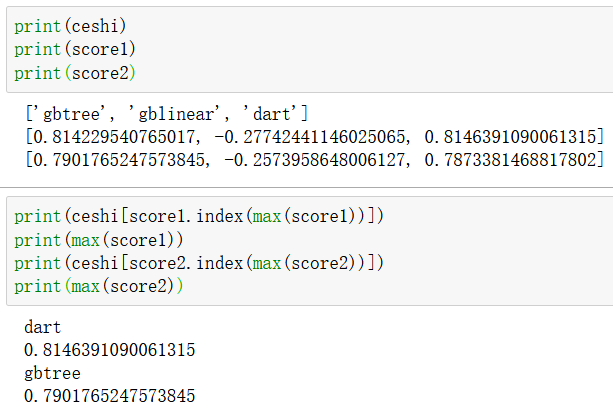
### 6.6.6.4 booster参数

梯度提升算法中不只有梯度提升树，XGBoost算法作为梯度提升算法的进化，自然也不只有树模型一种弱评估器。在XGBoost算法中，除了树模型，我们还可以选用线性模型，比如线性回归，来进行集成。虽然主流的XGB依然是树模型，但我们也可以使用其他的模型。基于XGBoost算法的这种性质，我们有参数“booster"来控制我们究竟使用怎样的弱评估器。对booster参数调参的代码如下所示。其中随机种子参数设置为100，n\_estimators设置为36，subsample设置为0.87，learning\_rate设置为0.17。



**图58 booster调参代码**

booster进行调参的结果如下所示：



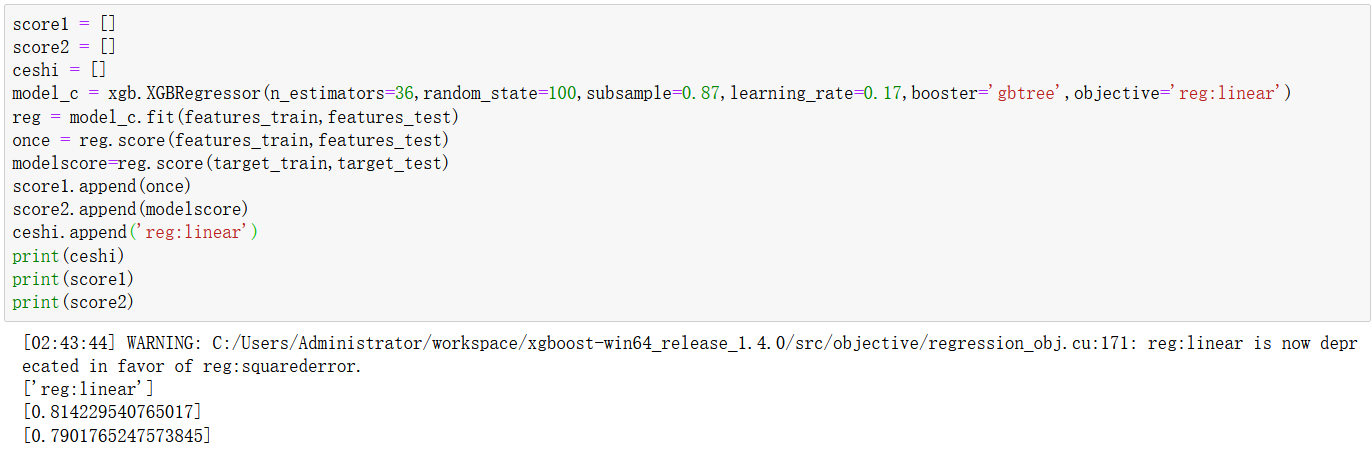
**图58 booster调参结果**

通过上图得到的booster进行调参的结果可以看出booster应该选择最佳的方法是’gbtree’。

### 6.6.6.5 objective参数

集成算法中的损失函数是可选的，要选用什么损失函数取决于我们希望解决什么问题，以及希望使用怎样的模型。比如说，如果我们的目标是进行回归预测，那我们可以选择调节后的均方误差RMSE作为我们的损失函数。如果我们是进行分类预测，那我们可以选择错误率error或者对数损失log\_loss。只要我们选出的函数是一个可微的，能够代表某种损失的函数，它就可以是我们XGBooster中的损失函数。对于回归类问题，XGBooster的模型能取的参数已经固定好了，只能使用‘reg:linear’。随机种子参数设置为100，n\_estimators设置为36，subsample设置为0.87，learning\_rate设置为0.17，booster设置为’gbtree’。

代码如下所示：



**图59 objective代码**

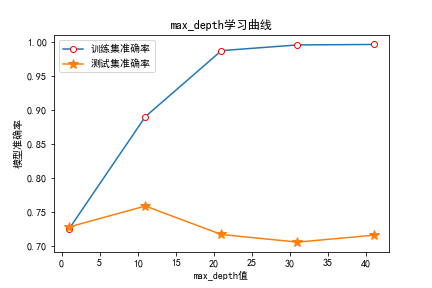
### 6.6.6.6 max\_depth参数

在上面我们研究了集成算法中对应集成过程的各个参数，接下来我们开始研究集成算法中带有的弱评估器本身的参数。max\_depth是树模型中用得最广泛的剪枝参数，在高维度低样本量时非常有效。可以用来限制集成算法中每棵树模型的最大深度，超过设定深度的树枝全部剪掉。因为树模型多生长一层，对样本量的需求会增加一倍，所以树深度过于高时容易导致发生过拟合现象。实验中固定XGBooster的随机种子参数设置为100，n\_estimators设置为36，subsample设置为0.87，learning\_rate设置为0.17，booster设置为’gbtree’，obective设置为‘reg:linear’。对max\_depth参数进行1到51，以10为间隔的方式进行初步调参的代码如下所示。



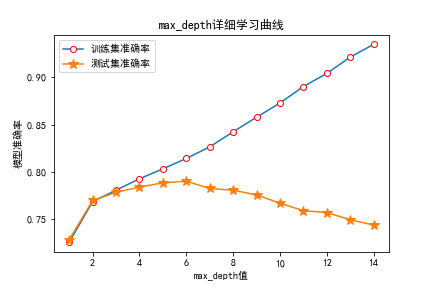
**图60 max\_depth调参代码**

max\_depth参数初步调参绘制出来的学习曲线如下所示。



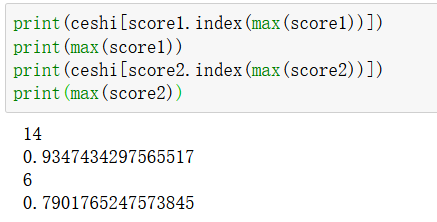
**图61 max\_depth初步调参学习曲线**

由图可知最佳的max\_depth取值应该在10附近。进下来对max\_depth参数进行1到15，以1为间隔的方式进行详细调参。



**图62 max\_depth详细调参学习曲线**

max\_depth详细调参的结果如下所示：



**图63 max\_depth详细调参的结果**

通过上图绘制出的max\_depth初步调参学习曲线我们可以看出随着模型深度的增加，在训练集上的准确率无线增加直到接近1，但是这只是我们拿来训练参数的数据集上的表现，在我们调参的数据集上的表现就不好了，是先上升后下降了。因为深度增加模型在训练的数据集上准确率肯定很高，但是对于没参与模型训练的数据集它的准确率就很低了。通过上图绘制出的max\_depth详细学习曲线和结果可以得到max\_depth最佳的取值是6。

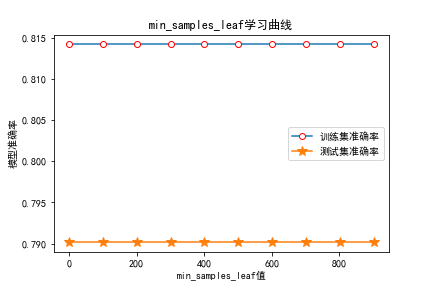
### 6.6.6.7 min\_samples\_leaf参数

min\_samples\_leaf参数是指叶子节点上应有的最少样例数，意指划分后的集成算法中的每一棵弱评估器的叶子节点最少有这么多样本。实验中固定XGBooster的随机种子参数设置为100，n\_estimators设置为36，subsample设置为0.87，learning\_rate设置为0.17，booster设置为’gbtree’，obective设置为‘reg:linear’，max\_depth固定为6。对min\_samples\_leaf参数进行1到101，以10为间隔的方式进行初步调参的代码如下所示。



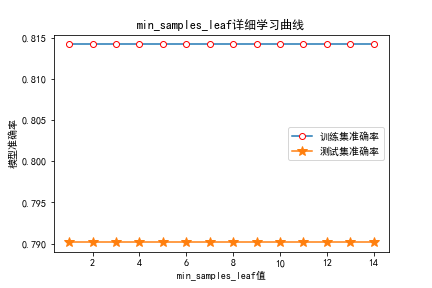
**图64 min\_samples\_leaf调参的代码**

min\_samples\_leaf参数初步调参绘制出来的学习曲线如下所示。



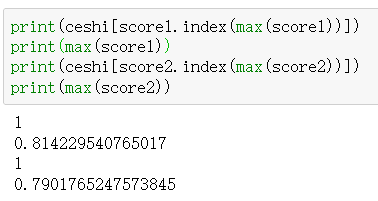
**图65 min\_samples\_leaf初步调参学习曲线**

由图可知最佳的min\_samples\_leaf取值可能在1附近。进下来对min\_samples\_leaf参数进行1到15，以1为间隔的方式进行详细调参。



**图66 min\_samples\_leaf详细调参学习曲线**

min\_samples\_leaf详细调参的结果如下所示：



**图67 min\_samples\_leaf详细调参的结果**

通过上图得到的min\_samples\_leaf进行调参的结果可以看出该参数对于模型建模影响不大。

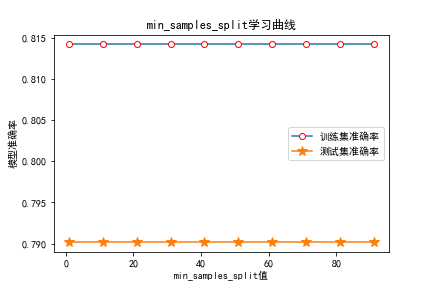
### 6.6.6.8 min\_samples\_split参数

实验中固定XGBooster的随机种子参数设置为100，n\_estimators设置为36，subsample设置为0.87，learning\_rate设置为0.17，booster设置为’gbtree’，obective设置为‘reg:linear’，max\_depth固定为6，min\_samples\_leaf固定为1。对min\_samples\_split参数进行1到101，以10为间隔的方式进行初步调参的代码如下所示。



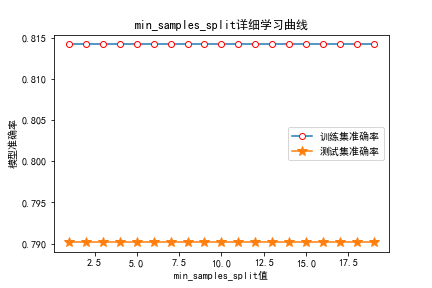
**图68 min\_samples\_split调参的代码**

min\_samples\_split参数初步调参绘制出来的学习曲线如下所示。



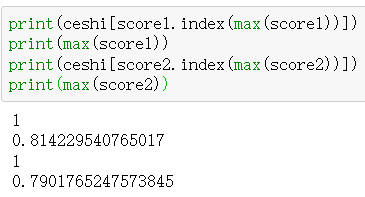
**图69 min\_samples\_split初步调参学习曲线**

由图可知最佳的min\_samples\_split取值可能在1附近。进下来对min\_samples\_split参数进行1到20，以1为间隔的方式进行详细调参。



**图70 min\_samples\_split详细调参学习曲线**

min\_samples\_split详细调参的结果如下所示：

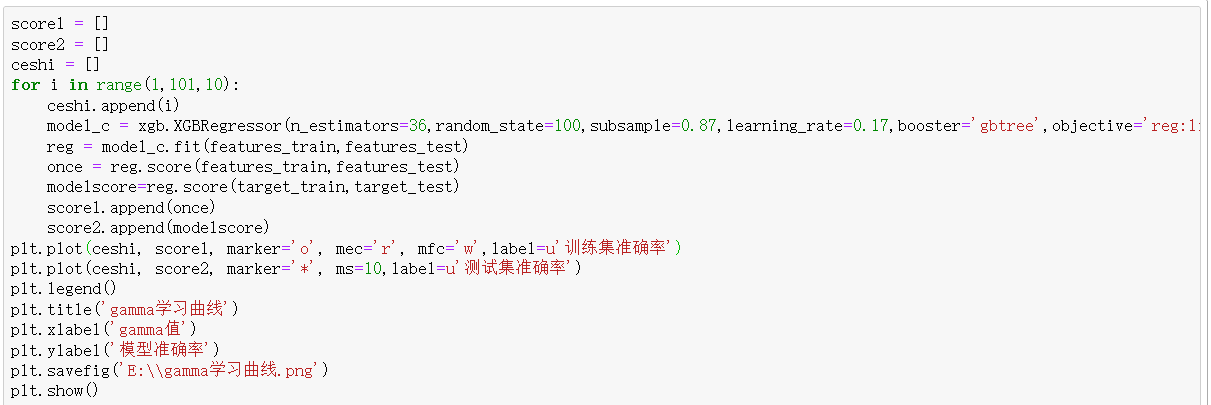


**图71 min\_samples\_split详细调参结果**

通过上图得到的min\_samples\_split进行调参的结果可以看出该参数对于模型建模影响不大。

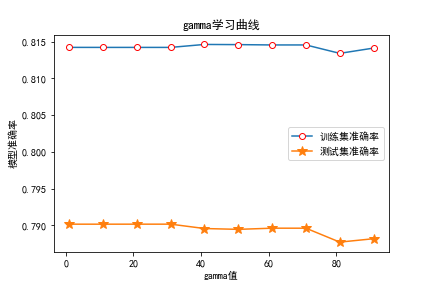
### 6.6.6.9 gamma参数

gamma参数指的是复杂项的惩罚因子，是指在树的叶节点上进行进一步分枝所需的最小目标函数减少量。该值设定越大，算法就越保守，树的叶子数量就越少，模型的复杂度就越低。实验中固定XGBooster的随机种子参数设置为100，n\_estimators设置为36，subsample设置为0.87，learning\_rate设置为0.17，booster设置为’gbtree’，obective设置为‘reg:linear’，max\_depth固定为6，min\_samples\_leaf固定为1，min\_samples\_leaf固定为1。对gamma参数进行1到101，以10为间隔的方式进行初步调参的代码如下所示。



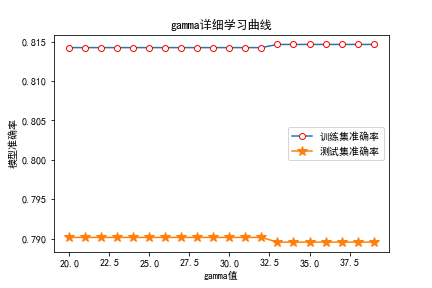
**图72 gamma调参代码**

gamma参数初步调参绘制出来的学习曲线如下所示。



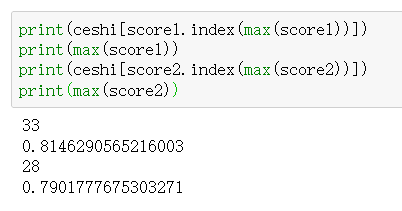
**图73 gamma初步调参学习曲线**

由图可知最佳的gamma取值可能在30附近。进下来对gamma参数进行20到40，以1为间隔的方式进行详细调参。



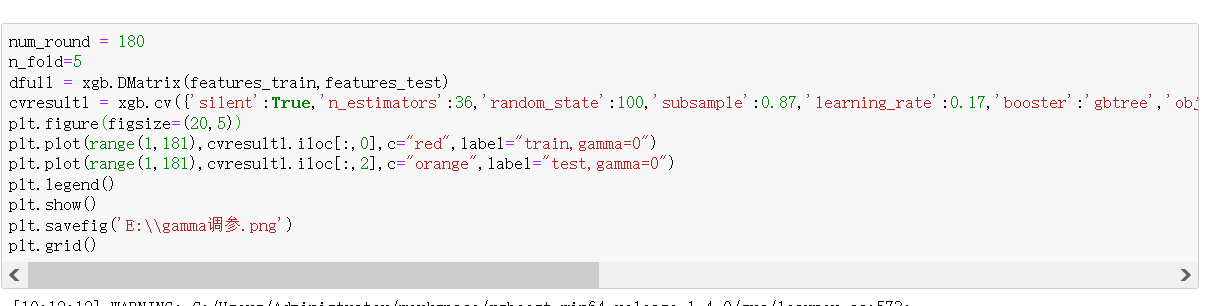
**图74 gamma详细调参学习曲线**

gamma详细调参的结果如下所示：

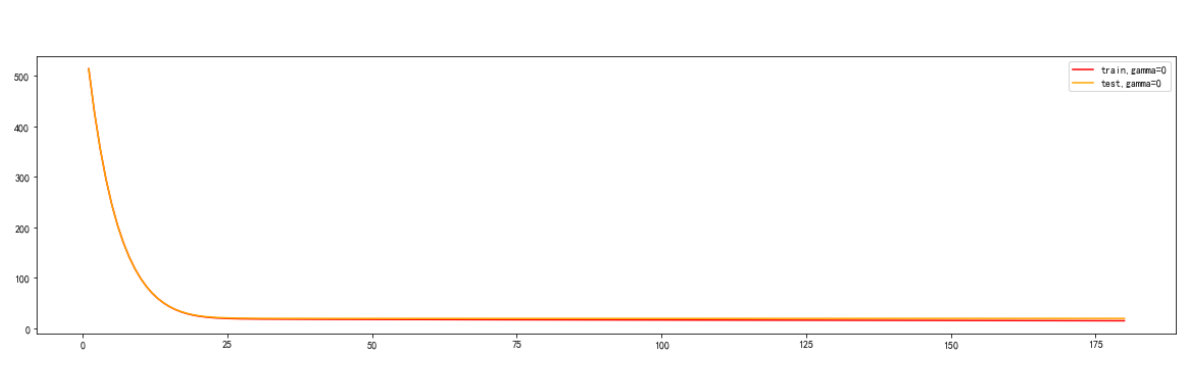


**图75 gamma详细调参结果**

通过gamma详细调参的结果和学习曲线可以知道得到的gamma的最佳取值是28。我们也可以使用XGBoost库自带的调参工具，使用XGBoost库自带的调参工具代码如下所示。



**图76 gamma自动调参**

绘制出的学习曲线如下所示：

**图77 gamma自动调参结果**

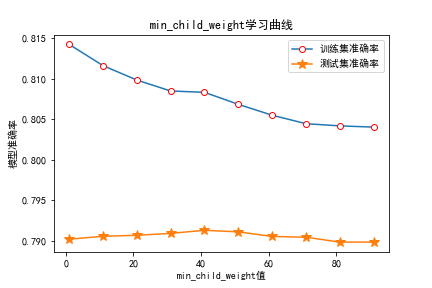
### 6.6.6.10 min\_child\_weight参数

实验中固定XGBooster的随机种子参数设置为100，n\_estimators设置为36，subsample设置为0.87，learning\_rate设置为0.17，booster设置为’gbtree’，obective设置为‘reg:linear’，max\_depth固定为6，min\_samples\_leaf固定为1，min\_samples\_leaf固定为1，gamma参数固定为28。对min\_child\_weight参数进行1到101，以10为间隔的方式进行初步调参的代码如下所示。

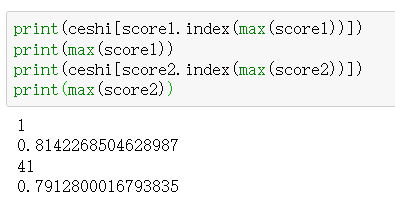


**图78 min\_child\_weight调参的代码**

min\_child\_weight参数初步调参绘制出来的学习曲线如下所示。



**图79 min\_child\_weight初步调参学习曲线**



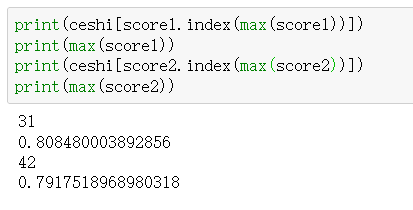
**图80 min\_child\_weight初步调参的结果**

由图可知最佳的min\_child\_weight取值可能在41附近。进下来对gamma参数进行30到50，以1为间隔的方式进行详细调参。



**图81 min\_child\_weight详细调参学习曲线**

min\_child\_weight详细调参的结果如下所示：



**图82 min\_child\_weight详细调参的结果**

通过上图得到的min\_child\_weight详细调参的结果和学习曲线可以知道该参数的最佳取值是42。

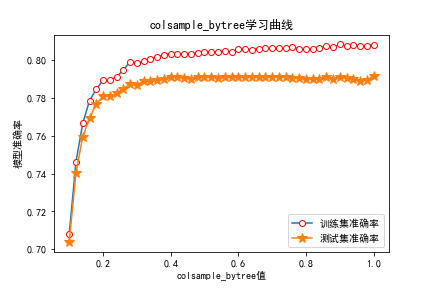
### 6.6.6.11 colsample\_bytree参数

实验中固定XGBooster的随机种子参数设置为100，n\_estimators设置为36，subsample设置为0.87，learning\_rate设置为0.17，booster设置为’gbtree’，obective设置为‘reg:linear’，max\_depth固定为6，min\_samples\_leaf固定为1，min\_samples\_leaf固定为1，gamma参数固定为28，min\_child\_weight固定为42。对colsample\_bytree参数进行0.1到1，以0.1为间隔的方式进行初步调参的代码如下所示。

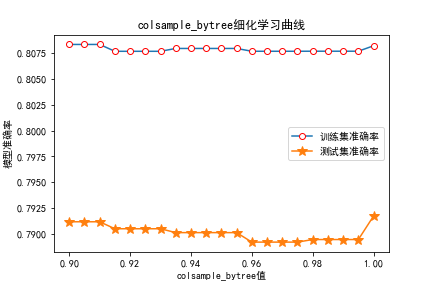


**图83 colsample\_bytree调参的代码**

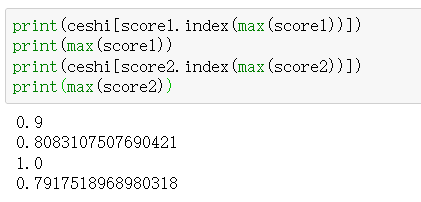
colsample\_bytree参数初步调参绘制出来的学习曲线如下所示。



**图84 colsample\_bytree初步调参的学习曲线**



**图85 colsample\_bytree详细调参的学习曲线**



**图86 colsample\_bytree详细调参的结果**

通过上图得到的colsample\_bytree详细调参的结果和学习曲线可以知道该参数的最佳取值是1。

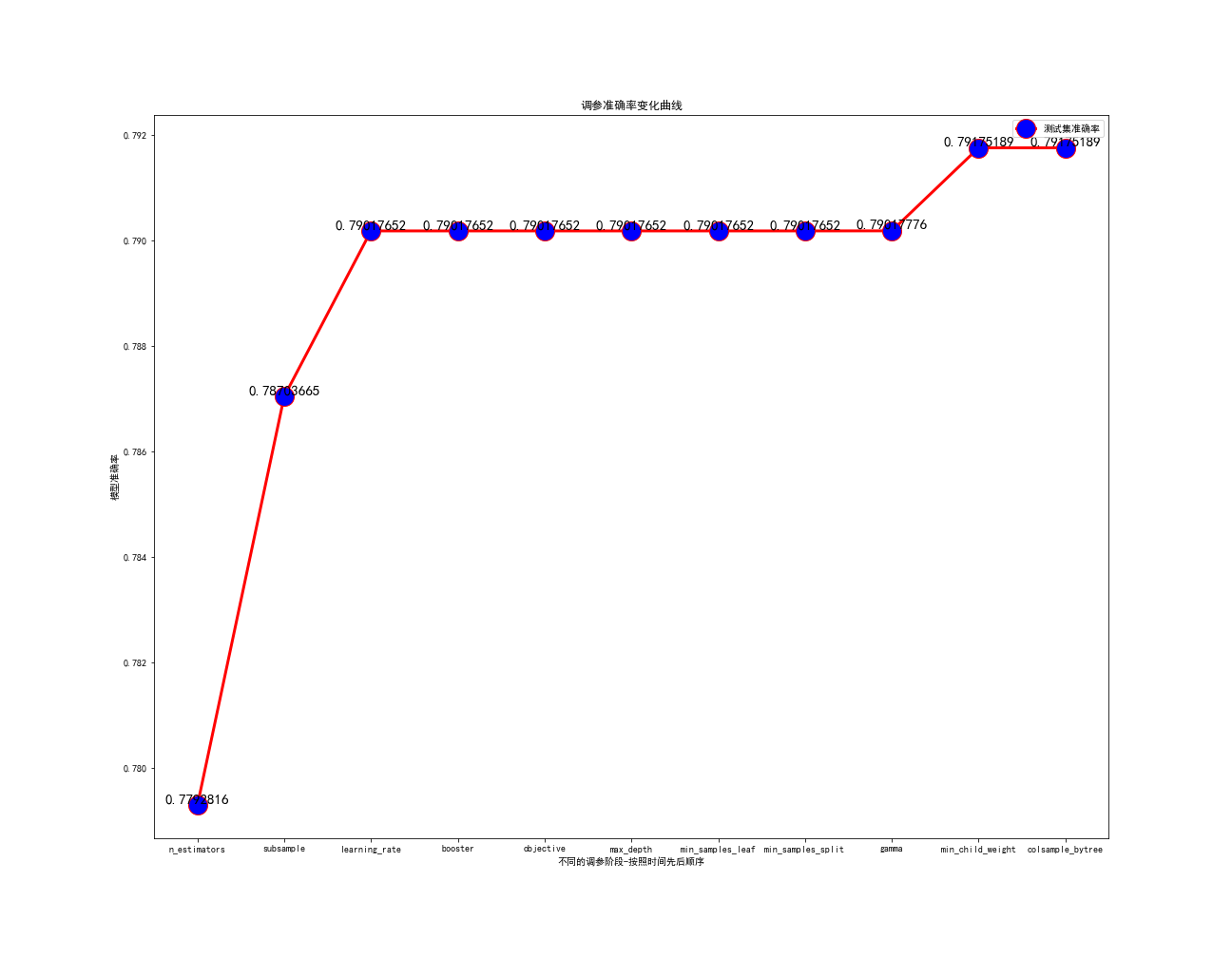
### 6.6.7学习曲线调参效果分析

我们之前使用控制变量法按照顺序对模型的参数进下了一个个地调整，获得了模型的最优解所对应的模型各个参数，在测试集上模型最高的得分为0.7917518968980318。这个时候XGBoost模型各个参数如下表所示。

|  |  |
| --- | --- |
| 参数 | 取值 |
| n\_estimators | 36 |
| random\_state | 100 |
| subsample | 0.87 |
| learning\_rate | 0.17 |
| booster | gbtree |
| objective | reg:linear |
| max\_depth | 6 |
| min\_samples\_leaf | 1 |
| min\_samples\_split=1 | 1 |
| gamma | 28 |
| min\_child\_weight | 42 |
| colsample\_bytree | 1 |

**表2 XGBoost算法模型的参数**

我对通过使用学习曲线法调参的过程进行了总结分析，对于调参过程，不同的参数对于模型准确率的影响是差别非常大的，有些参数对于模型的准确率影响微乎其微，有些参数可能模型默认的参数就是最适合的参数，而有些参数调参以后可能模型的准确率会发生巨大的变化。对于集成算法XGBoost来说，它的参数主要可以分为3类，一类参数是用来决定弱评估器本身的性质特征的参数，如max\_depth，一类参数是用来决定弱评估器集成模型的过程的参数，如n\_estimators，还有一类参数是用来控制算法模型过拟合的程度，如gamma。下面作出来的模型准确率随时间变化关系的折线图。



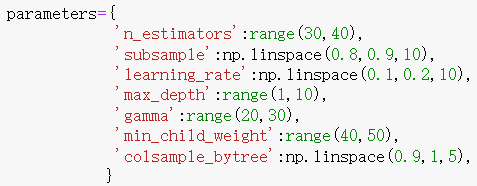
**图87 XGBoost调整不同参数各阶段对模型准确率的影响**

通过折线图我们可以看出来调整n\_estimators、subsample等对应集成算法的过程的参数对于提升模型的准确率有很大的帮助。而调整对应弱评估器本身性质的参数，如max\_depth等几乎不改变模型准确率，说明弱评估器本身默认的参数已经优化的足够好了，模型参数对应的模型准确率已经是最优的了。

### 6.6.8随机搜索优化参数-Randomized SearchCV

在之前使用学习曲线调整参数中分别对各个参数进行遍历求解时，每个参数都是独立出来看的，考虑一个参数时就不考虑其他的参数了，这可以看作是逐步修正的固定变量法。这样求出来的每个参数在当前情况下是最优解，但是如果再考虑当前步骤之后的参数时可能就不是全局最优解了。所以同时遍历模型各个参数获得模型的最优解时更可靠的方法。首先我使用了随机搜索法来优化模型参数。

1. 首先创建网格参数，包括subsample、max\_depth、n\_estimators等参数，为了减少机器计算时间，取值范围缩小了一些。



**图88 Randomized SearchCV模型参数设置**

1. 调用sklearn中的model\_selection库的Randomized SearchCV函数来初始化XGBoost，设定random\_state为90，n\_jobs为-1。

IMG_256

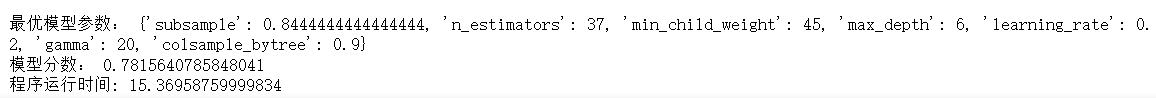
**图89 初始化XGBoost**

1. 实例化随机搜索。

IMG_256

**图90 实例化随机搜索**

1. 使用随机搜索进行拟合数据。
2. 查看最佳参数，subsample、max\_depth、n\_estimators等参数分别的数值和得分。



**图91 随机搜索的结果和参数取值**

### 6.6.9网格搜索优化参数-Grid SearchCV

使用随机搜索优化参数的效果并没有比使用控制变量法用学习曲线调整模型参数的效果好，这是十分正常的，因为随机搜索只是使用随机对分布取值，它大概率能逼近函数的最优解，而不能直接得到最优解。接下来我使用了网格搜索法来优化模型参数。

1. 首先创建网格参数，包括subsample、max\_depth、n\_estimators等参数，为了减少机器计算时间，取值范围缩小了一些。



**图92 Grid SearchCV模型参数设置**

1. 调用sklearn中的model\_selection库的Randomized SearchCV函数来初始化XGBoost，设定random\_state为90，n\_jobs为-1。

IMG_256

**图93 初始化XGBoost**

1. 实例化网格搜索。

IMG_256

**图94 实例化网格搜索**

1. 使用网格搜索进行拟合数据。
2. 查看最佳参数，subsample、max\_depth、n\_estimators等参数分别的数值和得分。



**图95 网格搜索的结果和参数取值**

### 6.6.10 网络调参效果分析

随机搜索和网格搜索都没办法到达使用学习曲线单变量调整模型参数的模型效果的最优解，一方面随即搜索鉴于取参数方法的限制，没办法遍历到参数的每一个取值。另一方面网格搜索由于参数的数量过于大，还有笔记本性能的限制，我只取了部份参数和缩小参数的取值范围来进行网格搜索，这可能是网格搜索没达到最优解的一个原因。

除此之外，通过比较随机搜索和网格搜索，我们可以看出来。随机搜索只需要在15秒内就能实现在百万级参数选择组合上进行5重交叉验证，而网格搜索在45秒内只能进行81钟参数组合的2重交叉验证。但是随机搜索的准确率却比网格搜索的更高。这说明我们不一定非要使用网格搜索法进行验证，随机搜索法也是可以考虑使用的。

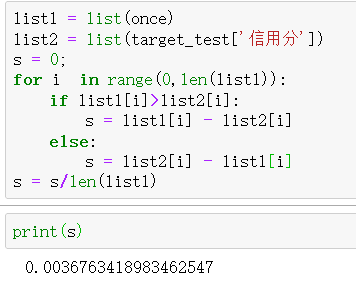
## 6.7模型评价与测试-基于MAE的结果

平均绝对差值MAE是用来衡量模型预测结果对标准结果的接近程度一种衡量方法。MAE的计算公式如下所示。

|  |  |
| --- | --- |
|  | (3) |

由于sklearn没有提供直接计算MAE指标的函数，所以需要我们手写计算MAE指标的函数。

1. list1 = list(once)
2. list2 = list(target\_test['信用分'])
3. s = 0
4. for i in range(0,len(list1)):
5. if list1[i]>list2[i]:
6. s = list1[i] - list2[i]
7. else:
8. s = list2[i] - list1[i]
9. s = s/len(list1)



**图96 MAE评价指标取值**

将XGBoost参数设置为100，n\_estimators设置为36，subsample设置为0.87，learning\_rate设置为0.17，booster设置为’gbtree’，obective设置为‘reg:linear’，max\_depth固定为6，min\_samples\_leaf固定为1，min\_samples\_leaf固定为1，gamma参数固定为28，min\_child\_weight固定了42，colsample\_bytree为1。通过MAE评估知道，它的MAE值是0.0036，MAE值越小越好，说明模型效果很好。

本项目要做的是一个回归模型。用来实现消费者行为画像，对其行为进行信用分评分。鉴于给定的情景中训练集和测试集的数据分布差异相当大，使用对抗验证的办法将训练集进行切分，使得用来作为调参依据的数据集的分布和测试集具有较高的相似性。在这个过程能有效地提高模型的训练效果，相比于对抗验证来说。

接下来我进行的特征工程，基于聚类算法的原理和基于实际场景的特征创造也能有效提高模型的准确率。

# 七、总结

在本次消费者人群画像-信用值智能评分的项目中，我先进行了正态分布检验和数据分布一致性检验，针对数据分布不一致的问题我通过使用LightGBM算法依据对抗验证的原理实现了数据集的重组，针对正态分布的问题，我建立了异常检测模型。在特征工程中我基于实际场景和聚类算法的原理进行了特征创造。在整个项目流程中，我完整地独立实现了一遍数据挖掘所需要的各个步骤，包括数据集获取，数据一致性检验，数据预处理，数据可视化，特征工程，建立模型，模型优化，模型评价，从而使得我对数据挖掘原理有了更深的理解。对于不同的数据集，要采取适合数据集特点的不同的数据预处理方法和算法模型才能更好地完成目标。通过这次消费者人群画像-信用值智能评分的项目，我对交叉验证、数据分布检验、特征工程和Embedded嵌入法和LightGBM算法以及XGBoost算法模型有了更深的理解和掌握。以及对如何通过使用学习曲线和网格搜索法来寻找最优参数有了更好的掌握。除此之外我对于如何进行模型融合有了初步的理解，在进行模型融合时，要对应用场景有着深刻的理解，自然地将模型融合在一起，而不是简单的堆砌模型。

评价表格

|  |  |
| --- | --- |
| 考核标准 | 得分 |
| （1）正确理解和掌握实验所涉及的概念和原理（20%）； |  |
| （2）按实验要求合理设计数据结构和程序结构（20%）； |  |
| （3）运行结果正确（20%）； |  |
| （4）认真记录实验数据，原理及实验结果分析准确（20%）； |  |
| （5）实验过程中，具有严谨的学习态度和认真、踏实、一丝不苟的科学作风（10%）； |  |
| （6）实验报告规范（10%）。 |  |
| 合计 |  |