****

课程设计报告书

**题目：《机器学习应用与实践》课程设计**

**学 院 经济与金融学院**

**专 业 金融科技**

**学生姓名 欧阳智**

**学生学号 201965442328**

**指导教师 贺建风**

**课程编号**

**课程学分**

**起始日期**

|  |  |
| --- | --- |
| 教  师  评  语 | 教师签名：  日期： |
| 成  绩  评  定 |  |
| 备  注 |  |

**基于机器学习的贷款违约评估**

**摘要：**随着我国经济的快速发展与居民超前消费理念的产生，借贷成为企业和个人解决经济问题的重要方式。为了鼓励居民消费，国家大力支持发展信贷产业及其他金融服务。过去银行等传统金融业对贷款业务的评估主要依靠5C分类法和经验判断法，效率较低，且过于依赖审核人员的主观判断。因此，传统的资信审查和风控模式逐渐无法满足日益增长的庞大贷款需求。机器学习能够在短时间内对大量数据多次模拟训练达到理想效果。银行等贷款机构利用机器学习算法，结合借款人提交的信息和各种替代数据来预测客户的还款能力，既可以为银行构建智能风控模型，又可以为因征信信息不足等原因无法获得贷款但实际具有还款能力的公众提供便利。故本文选取阿里云旗下大数据平台天池中的零基础入门金融风控-贷款违约预测中的样本数据进行研究，通过机器学习分类算法对贷款违约客户进行识别分类，首先对原始数据进行数据处理，包括缺失值和异常值的处理及特征预处理。用了SMOTE过采样算法等解决了非均衡样本数据的问题，建立XGBoost算法模型对贷款违约风险的预测提升对少数违约客户的预测效果。其次结合网格搜索算法对XGBoost模型的相关参数进行修正，提高了模型的泛化能力，给出两个不同数据集训练的模型，对不同企业应用对违约敏感程度不同的模型

**关键词：**贷款违约、机器学习、数据预处理、XGBoost、参数调整

**一、引言**

1.1选题背景及研究意义

1.1.1背景：

进入新世纪以后，随着我国经济过度依赖于出口的弊端不断凸显，为了促进经济健康和良性发展，我国提出了扩大内需的战略选择。在这样的背景下，国内众多商业银行积极开展了个人消费信贷业务。随着经济的发展，公众对贷款的需求越来越大。央行金融机构人民币信贷收支表显示，截止2022年5月，全国总贷款量为2035448.45亿元，其中.住户贷款 (Loans to Households)总量为724315.33亿元；企（事）业单位贷款 (Loans to Non-financial Enterprises and Government Departments & Organizations)总量为1296254.59亿元，二者占全国总贷款量的99.27%。同比往期数据，居民和企业的贷款需求呈增长、稳步上升的趋势。大量的贷款需求使得除了银行、信用社等传统的放贷机构，一些网络借贷平台也应运而生，如P2P，这种直接融资模式的诞生拓宽了传统贷款渠道，弥补了新时代下传统金融机构的不足。

1.1.2研究意义

在承认消费信贷业务促进我国国内需求方面发挥的巨大作用的同时，我们也必须清醒的看到，由于诚信意识不高、有关法律不健全、商业银行内部自身管理存在问题等多方面的原因，在我国个人消费信贷领域产生了一些不良贷款，不仅影响了个人消费信贷业务的开展，而且也给我们金融机构的健康发展带来了很多影响。而且仍有一部分公众因为没有信用记录、记录不全、资料有误无法核实等而难以获得贷款，从而被不可靠的放贷人利用。，不良贷款率增加会导致信贷机构风险提高，每一笔贷款的风险控制情况最终会影响营运、偿付能力，带来金融风险，如果这些金融企业的风险预警未能及时监控落实，就会给其带来不可估量的经济损失。在这种情况下，建立一个科学有效的风险评估模型对信贷业的健康发展显得尤为重要。本文通过构建集成分类模型，帮助信贷平台对客户进行风险评估，识别违约客户，以最大限度地对有可能遭遇的信用风险进行规避。

1.2研究现状

我国个人消费信贷业务最早可以追溯到20世纪80年代。不过由于当时的政治和经济环境，个人信贷业务在银行业务中的规模较小。1997年亚洲金融危机过后，为了促进国内消费，我国个人消费信贷业务得到了很好的发展。进入新世纪以后，随着国民经济的快速发展，我国银行业加强了金融产品的创新，个人消费信贷业务也为更多的消费者所接受，个人信贷业务不断发展壮大，市场空间不断拓展，个人住房信贷、个人汽车消费信贷、个人大额耐用消费品信贷、助学信贷等个人信贷业务迅速发展起来。借贷行业的健康发展离不开合理的违约预测系统，国内外对这类反欺诈系统已研究多年且成果颇丰。一个良好的金融风控模型或者说反欺诈系统应该具备及时对欺诈活动进行预警的能力，当用户出现可能违约的情况时，系统应及时提醒相应人员，进而阻止违约行为的发生，避免借贷平台遭受损失。目前，保险、电商和金融领域都已经建立了结合自己行业特征的反欺诈系统。国内外也有了相当一部分用机器学习算法作金融风控的研究。如佘朝兵（2018）用Lending Club 2007至2015年贷款期限为3年的贷款数据，选取了逻辑回归、XGBoost以及深度神经网络三个模型分别进行学习评分，并在此基础上建立了堆叠模型。过进一步的优化，堆叠模型准确率较单一模型提升了2%。[1]和沙靖岚（2017）比较了现在机器学习领域前沿的LightGBM与XGBoost算法在对网络借贷违约预测时的特点[2]等。

**二、基本算法和评价指标**

2.1基本算法

（a）XGBoost是Boosting框架下的集成树类算法。和GBDT相比，XGBoost的基学习器不仅可以是CART，而且可以是线性分类器[5]。另一方面，XGBoost算法将规则项添加到目标函数中，以控制模型的准确性和复杂性。和随机森林类似，XGBoost算法也对特征进行随机抽样，有效降低了计算成本。

基本原理：如果boosting算法每一步的弱分类器生成都是依据损失函数的梯度方向，则称之为梯度提升(Gradient boosting)，XGBoost算法是采用分步前向加性模型，只不过在每次迭代中生成弱学习器后不再需要计算一个系数，XGBoost 是由 k 个基模型组成的一个加法运算式：

其中为第k个基模型，为第i个样本的预测值，那么损失函数可由预测值与真实值进行标识：

其中n为样本数量。

XGBoost算法通过优化结构化损失函数（加入了正则项的损失函数，可以起到降低过拟合的风险）来实现弱学习器的生成，并且XGBoost算法没有采用搜索方法，而是直接利用了损失函数的一阶导数和二阶导数值，并通过预排序、加权分位数等技术来大大提高了算法的性能。

2.2分类预测评价指标

（a）ROC曲线：ROC空间将FPR(False Positive Rate)定义为x轴，TPR(TruePositive Rate)定义为y轴。在ROC空间中，预测结果用点进行表示，左上角的点是最为理想的预测结果。100%灵敏在这一空间中用点(0,1)表示，该点表示不存在假阳性和假阴性，我们将其视为一种“完美分类器”。当随机是一种完全随机时，此时意味着会计算得出一条从左上到右下的对角线，该对角线上任意一点的精确率都为50%。

（b）AUC(Area Under Curve)：坐标轴和ROC曲线围成的图形面积即为AUC，一般情况下，ROC曲线位于在y=x斜线上方，因此，AUC所表示的数值区[0.5,1]。数值越高表示检测结果越接近真实值。

AUC的一般评价标准：

|  |  |
| --- | --- |
| 0.5-0.7 | 效果较差 |
| 0.7-0.8 | 效果一般 |
| 0.8-0.95 | 效果较好 |
| 0.95-1 | 效果非常好 |

（c）混淆矩阵

|  |  |  |
| --- | --- | --- |
|  | **Predicted as Postive** | **Predicted as Negative** |
| **Labeled as Positive** | True Positive(TP) | False Negative()FN |
| **Labeled as Negative** | False Positive(FP) | True Negative(TN) |

准确率（Accuracy）:

查准率（Precision）：

召回率（Recall）：

F1-Score:

**三、数据描述与建模分析**过程论述

3.1数据描述和探索性分析

3.1.1数据来源

本文数据主要来源于该数据来自某信贷平台的贷款记录[零基础入门金融风控-贷款违约预测-天池大赛-阿里云天池 (aliyun.com)](https://tianchi.aliyun.com/competition/entrance/531830/information)，，总数据量超过100w，包含47列变量信息，其中15列为匿名变量。从中抽取80万条作为训练集，20万条作为测试集A，同时会对employmentTitle、purpose、postCode和title等信息进行脱敏（即用编号等分类信息区分对应的涉及隐私的敏感信息）。本研究过程主要采用2007-2018年的客户贷款数据进行分析，样本数据总体数量为80万，为了更充分地了解数据，本章节对样本数据采用探索性分析方法，更好地将特征工程前期工作做好。在这一过程中，运用简便的统计量来达到了解整体数据的目的，并对特征相互关联性加以分析，选取理想的图形对数据进行可视化展示。为使原始数据充分应用到试验中，我们需要使用Python对数据进行读取和处理，通常在处理数据时使用pandas，numpy，seaborn等函数库；在建立模型时需要用到了sklearn，XGBoost等函数库。

3.1.2数据描述

本文样本数据共包含47列特征信息，其中匿名特征变量共有15列。表3.1对原始样本数据的特征进行了详细解释说明。n0-n14表示匿名特征，是一些贷款人行为计数特征的处理。



表3-1特征说明

图3-2和3-3显示变量的基本信息和部分统计量

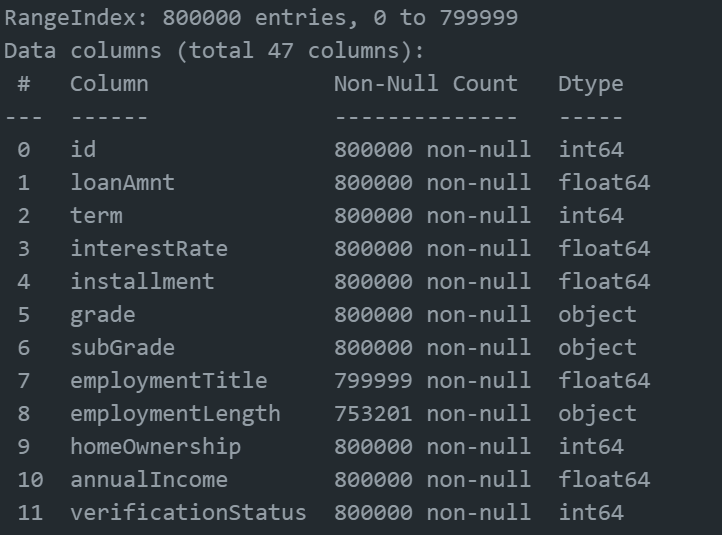


图3-2部分数据的一些基本信息（缺失情况、类型等等）

图3-3部分数据的描述性统计分析

图3-5到图3-8是对连续性特征、离散特征和分类特征等的可视化呈现

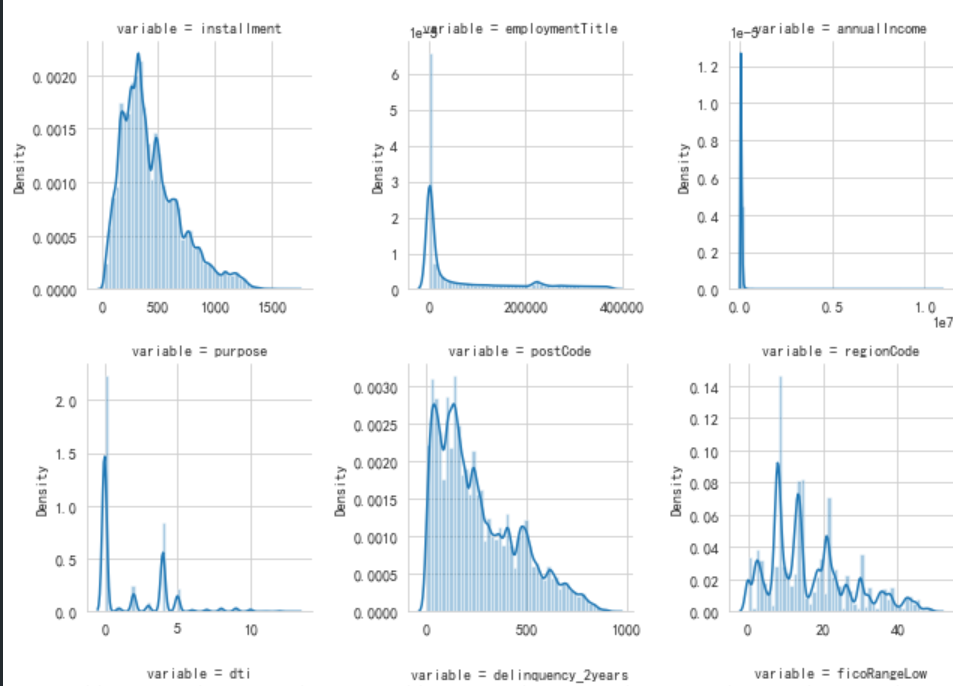


图3-4部分连续型变量的可视化

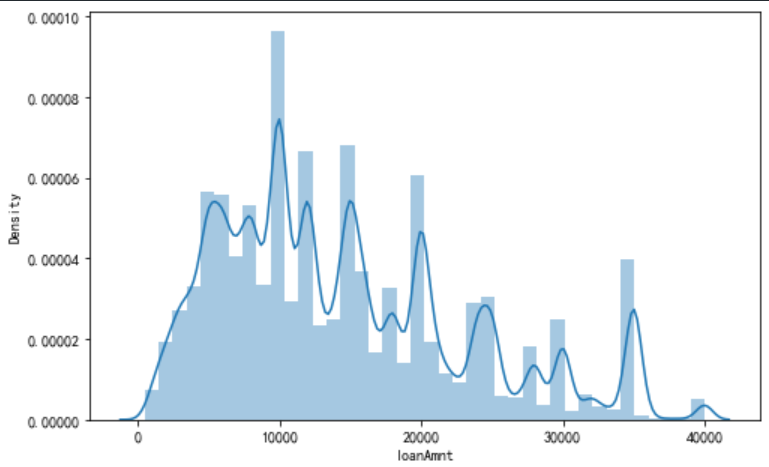


图3-5贷款金额 loanAmnt 的分布情况

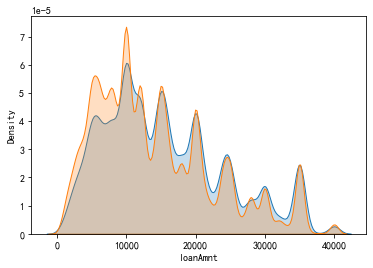


图3-6违约与不违约两类样本的贷款金额分布情况

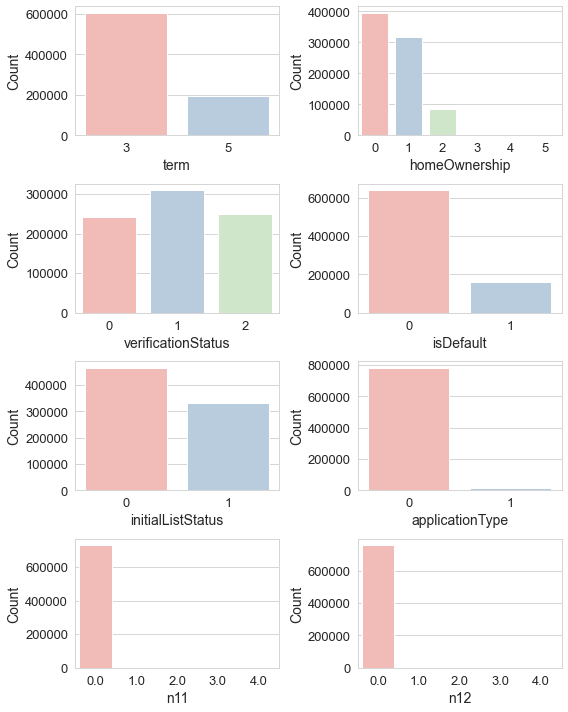


图3-7离散型表量的可视化呈现

图3-8显示模型样本是不平衡的，故特征工程需要平衡数据集

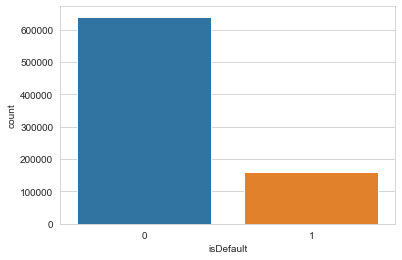


图3-8贷款违约与不违约的样本数量分布

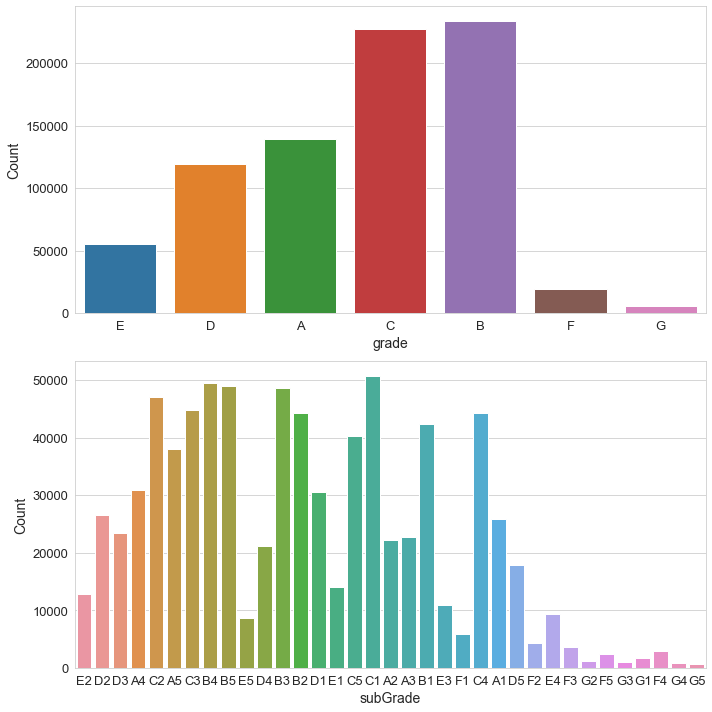


图3-9部分分类型特征的可视化呈现

图3-10显示那些特征值有缺失值

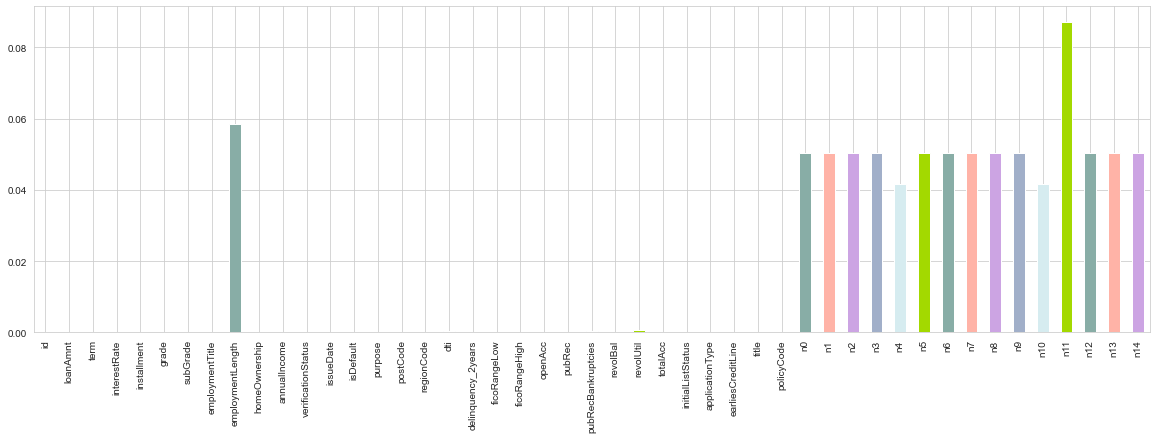


图3-10缺失特征的可视化呈现

图3-11可以看到，有些变量之间的相关性还是很强的，比如贷款总额loanAmnt 和分期付款金额installment 相关性为1，ficoRangeLow he ficoRangeHigh 相关性为1。

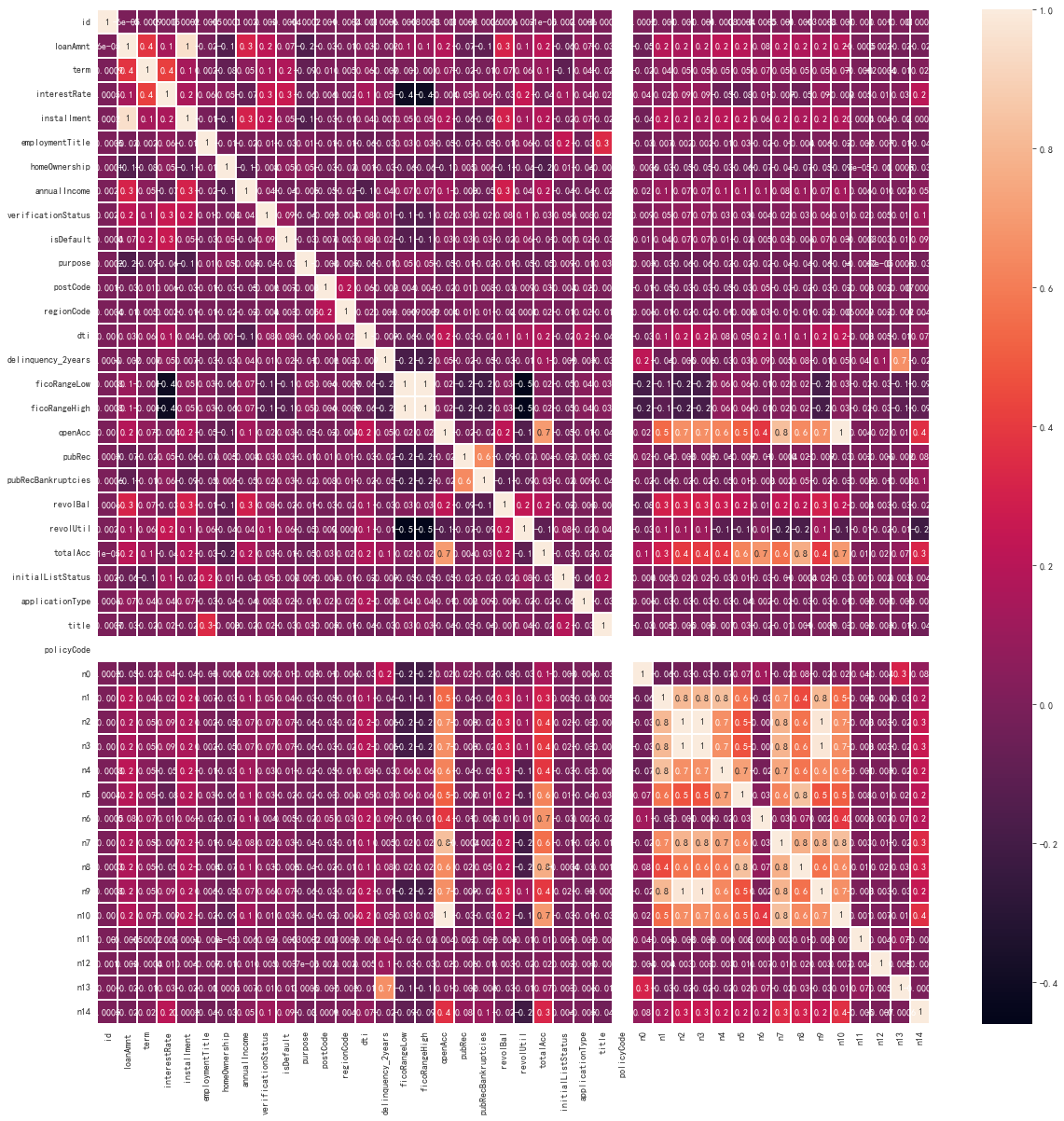


图3-11特征的相关关系可视化呈现

3.2数据预处理

3.2.1数据清洗

（a）重复值处理（本次训练集中不存在重复值）

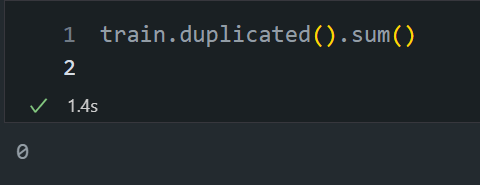


图3-12训练集中的重复值数量

（b）缺失值填补（缺失值填补后的结果如图3-13所示）

一般来说如果是分类型特征采用众数进行填补，如果是连续型还要①考虑均值一般适用于近似正态分布数据，观测值较为均匀散布均值周围②中位数一般适用于偏态分布或者有离群点数据，中位数是更好地代表数据中心趋势③众数一般用于类别变量，无大小、先后顺序之分，所以对于数据近似符合正态分布，用该变量的**均值**填补缺失，对于数据存在偏态分布的情况，采用**中位数**进行填补。

本文中连续型和离散型特征都统一使用中位数填补，分类型特征使用众数进行填补，上述两次填补完成之后还存在着特征employmentLength有缺失值，因此使用采用决策树来填补就业年限（employmentLength）。

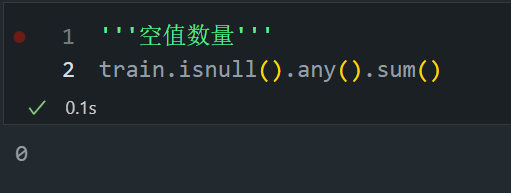


图3-13训练集中的空值数量

（c）异常值处理（没有进行异常值处理）

在数据描述中的特征可视化呈现中发现变量的分布都是相对集中的，不存在特别的大的偏差，因此本文没有进行异常值处理

3.2.2特征预处理

（a）时间数据的处理（处理后结果如图3-14）

①对于时间本身的特征，选择将时间变量作为类别变量

②对于时间变量之间的组合特征，选择将多个特征根据含义组合

本文中需要处理的时间数据为issueDate，选择将其转化为表示数据日期离数据集中日期最早的日期（2007-06-01）的天数得到新特征issueDateDT，测试集与训练集同理

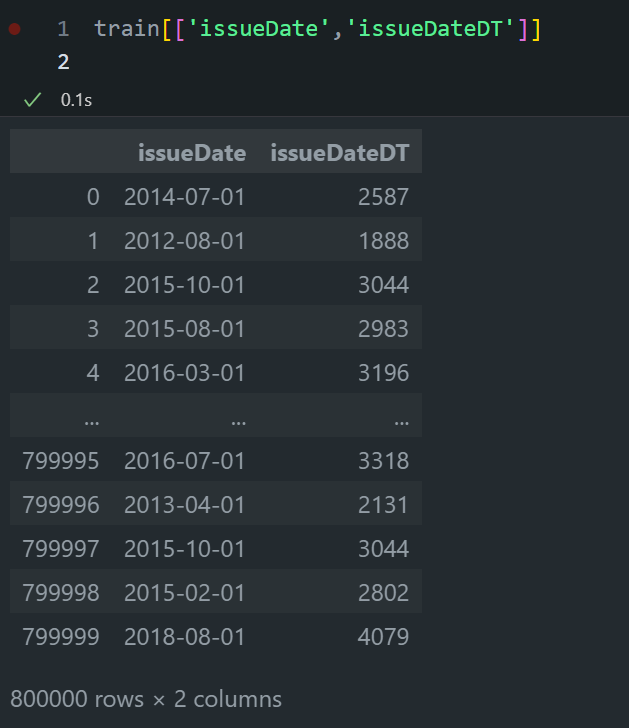


图3-14训练集时间数据处理效果图

（b）特征交叉

特征交叉是指数值特征之间的相互作用，在本文中利用到 issueDate（贷款发放的月份）处理earliesCreditLine（借款人最早报告的信用额度开立的月份），得到新的特征CreditLine（开卡年限），然后将issueDate和earliesCreditLine特征删除.

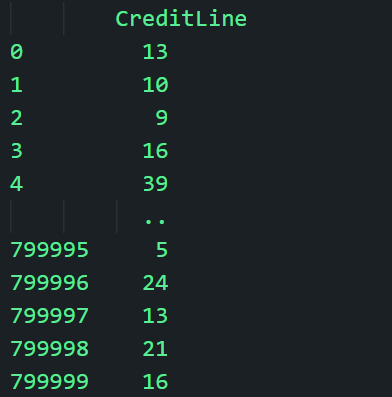


图3-16新加入的特征CreditLine

（c）特征编码

对于离散型特征，可以使用OneHotEncoder独热编码，本文中处理的特征有train[['homeOwnership','verificationStatus','purpose']]

3.2.3特征选择

（a）根据相关性删除一些特征，使模型更稳定

①人工判断与目标无关联特征为"id"，删除id列

②求出各个特征与目标的相关系数，综合考虑排除corr小于0.01的特征删除了initialListStatus,n5,n11,n12,n8,postCode,policyCode

③特征间高相关过滤，删除两两特征之间高于0.6的特征loanAmnt贷款金额，installment分期付款金额两个特征间相关系数为0.95，ficoRangeLow所属的下限范围，ficoRangeHigh所属的上限范围特征间相关系数为1，openAcc未结信用额度的数量，n10两个特征间相关系数为0.93，n3，n2两个特征间相关系数为1；n3，n9 两个特征间相关系数为0.98，故选择删除特征installment,ficoRangeHigh,openAcc,n3,n9

④低方差过滤结合相关性过滤方差小于0.1的特征applicationType

（b）样本不平衡处理

分类问题中各类别样本数量差距太大，会造成样本不均衡的问题。样本不均衡不利于建立与训练出正确的模型，且不能做出合理的评估，在数据描述中的图3-8显示了违约和未违约的样本数量大于为4：1，因此分别生成了上抽样和下抽样的的数据，上抽样数据如图3-17，下抽样数据如图3-18

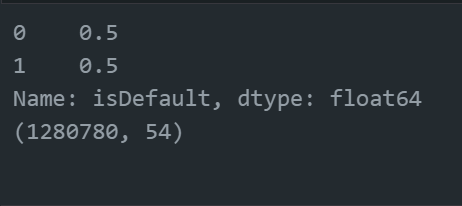
****

图3-17上抽样结果

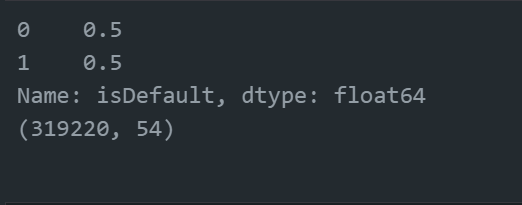


图3-18下抽样结果图

3.3建模分析和调参

3.3.1建模分析

完成数据预处理过程之后：

1. 首先在数据的训练集和测试集的划分上放弃了K折交叉验证法（尝试过5折交叉验证运行时间相对较长，不适合调参等后续步骤的进行，又因为数据量相对充足，因此选择留出法来进行训练/测试集的划分，size=0.25）
2. 因为存在样本数量不均衡的情况，在特征选择时，将样本进行了上采样样和下采样处理，因此模型中对原数据、上采样数据和下采样数据都进行了训练:

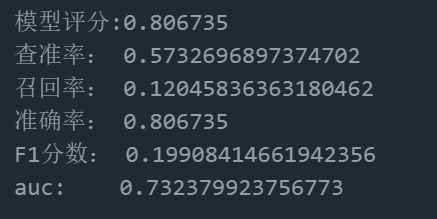


图3-19原数据模型的评价指标

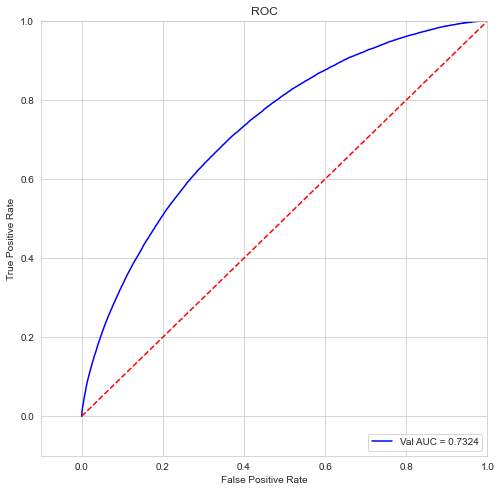


图3-20原数据模型的ROC曲线

由图3-19可见，模型的准确率较高，但查准率和召回率却不尽人意，分析原因可知，因为训练集样本存在不平衡的现象，违约和未违约的样本数量比时4：1，所以对模型产生分类的影响（原因分析XGB算法会误认数据量大的重要程度高），对此现象的改进是模型使用上采样的数据和下采样的数据再次进行训练，训练结果如下所示：

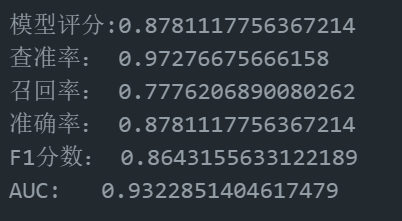


图3-21上采样数据模型的评价指标

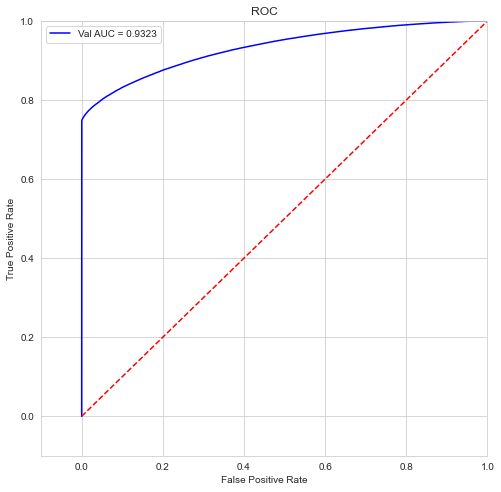


图3-24上采样数据模型的ROC曲线

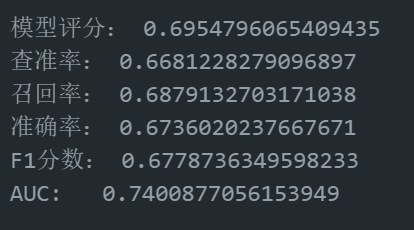


图3-25下采样数据模型的评价指标

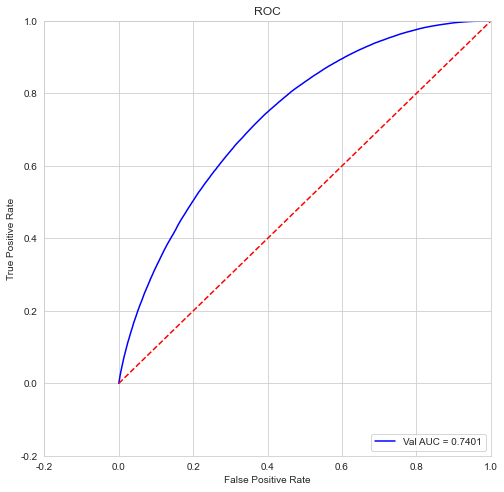


图3-26下抽样数据模型的ROC曲线

对上采样数据训练得结果分析可得：在平衡样本之后，模型的AUC，查准率和都得到提升，模型能够在平衡后的训练集上相对于原数据得到得到更好的预测效果，说明上采样对XGBoost模型的提升效果较为不错，smote过采样算法可以很好的处理数据集不平衡带来的差距，但上采样数据训练的模型的ROC曲线不平滑，在后续调参时需要对比三个数据集对模型参数调整的效果（网格搜索算法调参算法运行时间较长，且不一定有结果）

3.3.2调参

调参使用的是GridSearchCV算法，GridSearchCV的名字其实可以拆分为两部分，GridSearch和CV，即网格搜索和交叉验证。这两个名字都非常好理解。网格搜索，搜索的是参数，即在指定的参数范围内，通过循环遍历，尝试每一种可能性，利用调整的参数训练学习器，从所有的参数中找到在验证集上精度最高的参数，这其实是一个训练和比较的过程，表现最好的参数就是最终的结果，缺点是运算时间，在面对大数据集和多参数是非常耗时的，且可能会调到局部最优而不是全局最优，但优点是自动调参，只要把参数输进去，就能给出最优化的结果和参数。

经过网格搜索和人工手动调参（由于有些调参过程时间过长故事用人工手动调参）后的参数表：

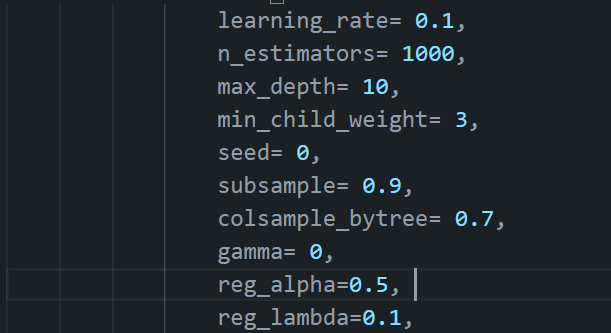


图3-27调参后的参数表

四、结果分析

对原数据、上采样数据和下采样数据分别用调参后的模型重新训练，得到如下结果：

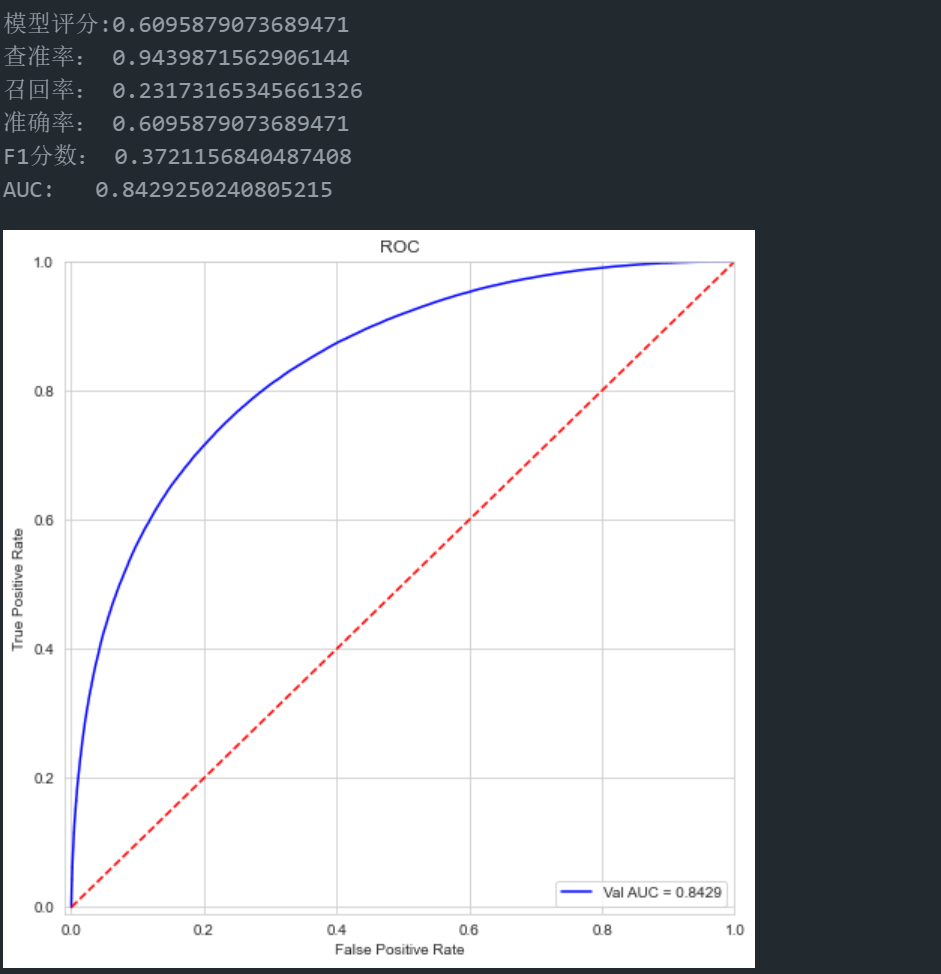


图3-28原数据调参后模型训练结果

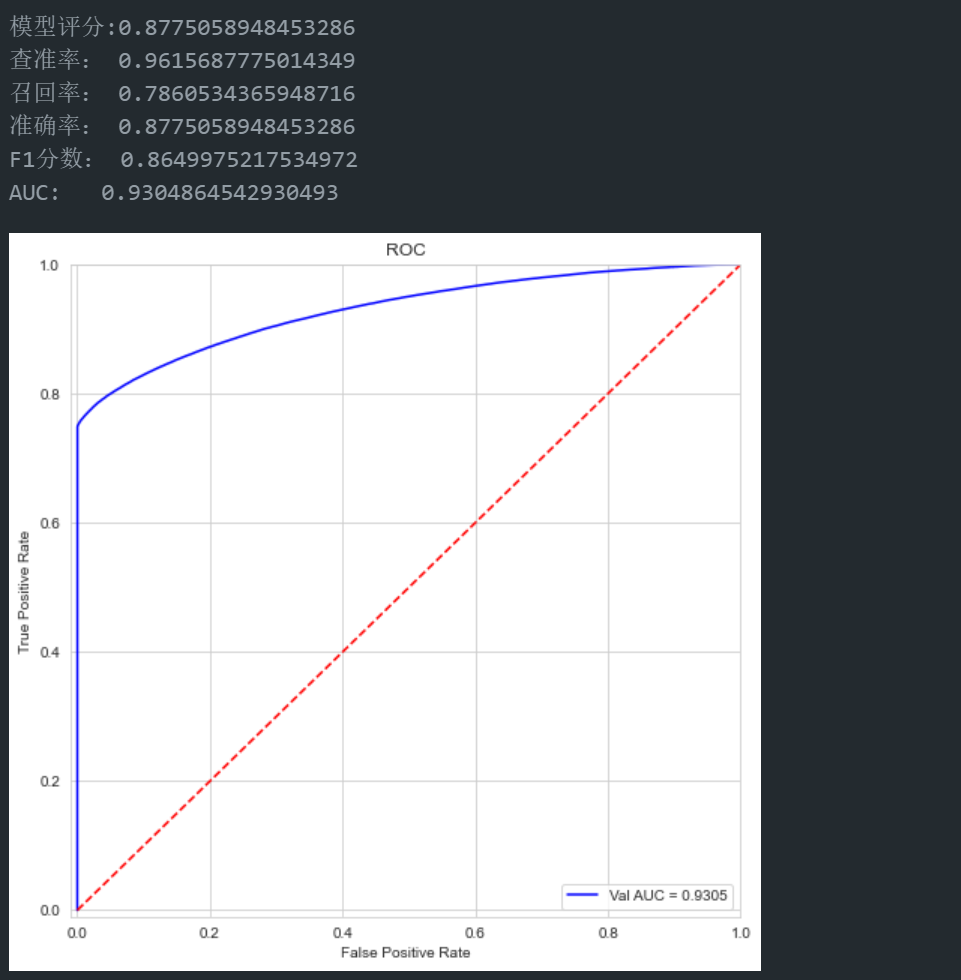


图3-29上采样数据调参后模型训练结果

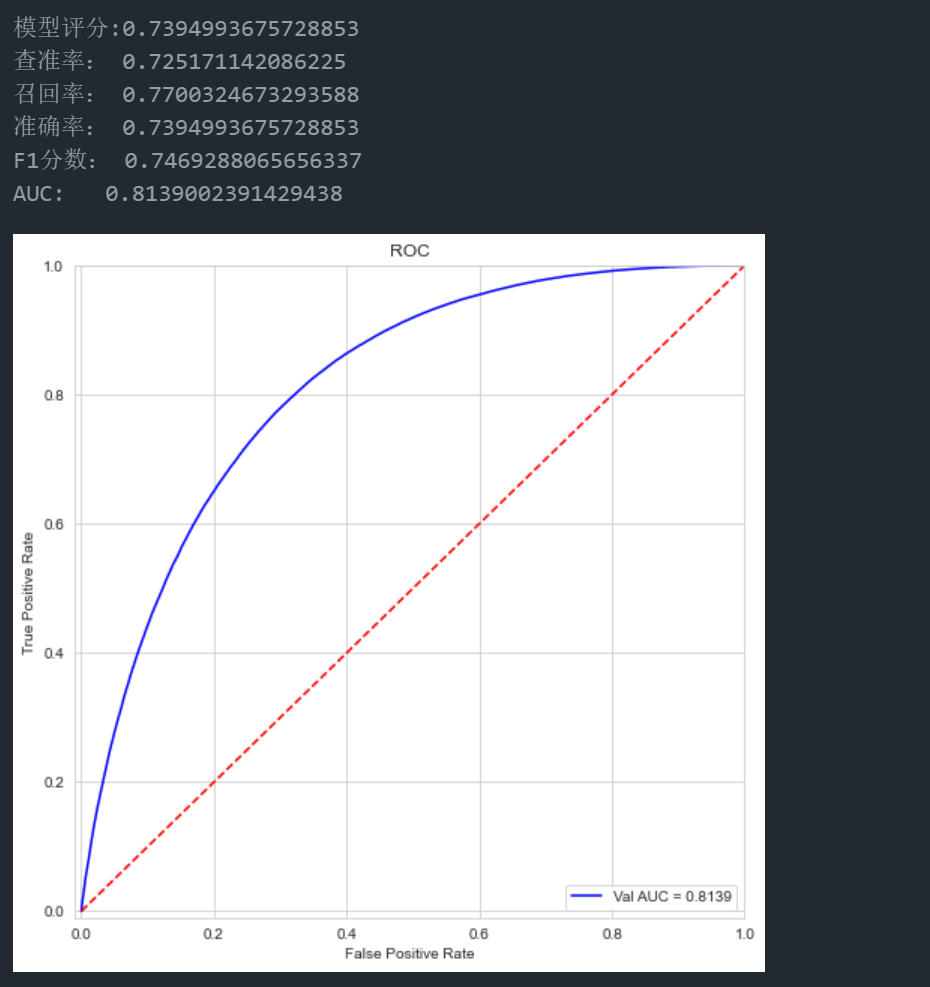


图3-30下采样数据调参后模型训练结果

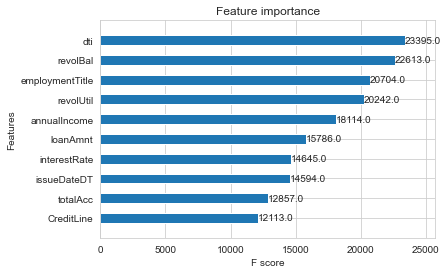


图3-31下采样数据的特征重要性

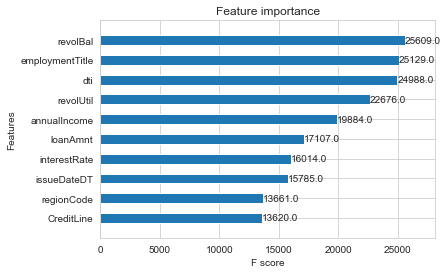


图3-32原数据的特征重要性

对于图3-31和图3-32得两个模型得特征重要性总结：

原数据训练模型特征重要性前十：

1. 信贷周转余额合计
2. 就业职称
3. 债务收入比
4. 循环额度利用率，或借款人使用的相对于所有可用循环信贷的信贷金额
5. 年收入
6. 贷款金额
7. 分期付款金额
8. 数据日期离数据集中日期最早的日期（2007-06-01）的天数
9. 地区编码
10. 开卡年限（交叉特征）

下采样数据训练模型特征重要性前十:

1. 债务收入比
2. 信贷周转余额合计
3. 就业职称
4. 循环额度利用率，或借款人使用的相对于所有可用循环信贷的信贷金额
5. 年收入
6. 贷款金额
7. 贷款利率
8. 数据日期离数据集中日期最早的日期（2007-06-01）的天数
9. 借款人信用档案中当前的信用额度总数
10. 开卡年限（交叉特征）

对于特征重要性的结果对比可以看出：原数据模型更看重可以体现客户偿债能力的指标而下采样数据模型则比较平衡，既考虑了风险，也考虑了收益率，对两个模型整体的评价就是原数据模型优先考虑风险，其次再是收益，而下采样数据则平等对待这两类指标。

对上述结果图（28-30）的分析;首先考虑上采样数据（图3-29）训练结果，正常来说ROC曲线是不受样本平衡状况影响的，但原数据训练模型对比出现了不平滑和评价指标集体提升的现象，为此本文给出的解释是可能是因为出现了过拟合，因为smote算法新增的数据是与临近样本相似的，即使效果好于直接复制小样本，但当需要新增样本数量较大时仍会出现过拟合的情况，该数据集中，本文中的上采样增加的违约的样本数量为480780，原数据是800000，是原数据的60%左右，因此即使上采样的数据训练出来的结果各项指标都较高，本文仍不选择上采样数据作为调参后模型的训练。

对比原数据和下采样数据，用原数据训练（图3-28）的模型AUC和查准率较高，说明对于违约样本的预测水平较好，但召回率和准确率较低，说明对未违约的预测效果不如违约的预测效果，而使用下采样数据训练（图3-30）的模型各项指标较为均衡，AUC>0.8处于较好水平，对于违约和未违约的预测效果几乎一致.

因此对于本文得出两种违约预测模型，原数据训练模型作为A模型，下采样数据训练模型作为B模型，对于资金流水较小，抗风险能力较差得小企业如城镇银行等可以考虑对违约预测效果较好的A模型（对违约敏感提高企业的资金安全性），而对于资金流水大，抗风险能力较强的企业如工商银行等可以使用B模型（利于放贷，提高资金的利用率）

五、课程设计总结

在这次课程设计当中，首先寻找有关违约预测的数据源花费了一定的时间，其次是算法的选择，初衷是使用多个算法训练然后再通过投票的方式将模型融合达到更好的预测效果，但是在实践中遇到了较多的小bug，如单个模型预测时效果较好，将多个模型混合后在AUC的指标上预测效果反而比单个模型差，经过很长时间的调试（本身数据量大，电脑算力有限，一次调试长则几个小时，短则几十分钟）也没有将模型效果修正，因此只选择展示拟合效果较好的XGBoost算法，在实际XGBoost算法运行中，模型的拟合花费的时间不会很长，但在超参数的调整里花费时间很长，以图3-33的max\_depth和min\_child\_weight超参数的调参过程为例，在处理调参中max\_depth的取值范围设置的是[4,5,6,7,8,9,10]，在代码运行到第631分钟时电脑蓝屏了因此后续将取值范围设置在[8,9,10]，后续在接近3个半小时的算法中得出了max\_depth的最优参数：10，其他的超参数的调整花费时间也相差无几，因此在论文的编写过程中大部分时间都是花费在调参过程中的，而且网格搜索算法调参过程是连续的，在得到这一份调参代码的输出之后才可以为下一部分参数调参输入初始值，也试过写while循环判断输出当这一份调参代码输出结果后继续运行下一份参数调整的代码，但该方法存在很多漏洞，首先时运算量大大增加耗时更长，其次是这一次的调参代码不一定有输出，当没有输出时就需要人工手动调整参数范围重新迭代，while循环只会输出不太正确的结果，因此在整个调参过程中花费了大量时间，在这个过程中我熟悉了XGBoost算法的各个超参数，对XGBoost算法也有了更深刻的理解,也发现了自己的不足，对于很多算法仍不熟悉，缺乏实践经验，对于大数据分析机器学习方向还需要更进一步的学习。

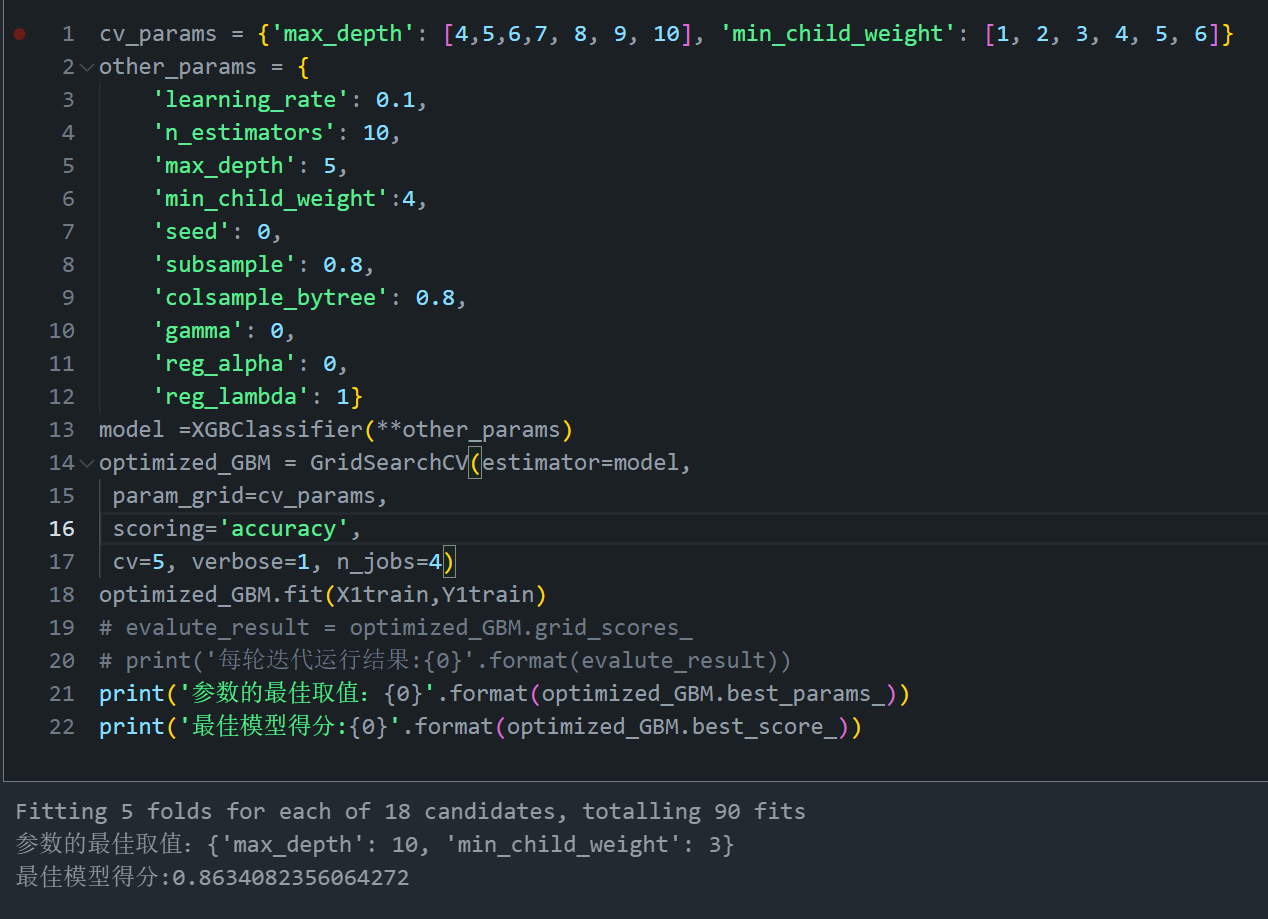


图3-33max\_depth和min\_child\_weight超参数的调参过程

**参考文献**

[1]佘朝兵.Logistic回归在银行个人信贷风险评估上的应用研究[J].科技与创新,2018(19):113-114+118-119.

[2]沙靖岚. 基于LightGBM与XGBoost算法的P2P网络借贷违约预测模型的比较研究[D].东北财经大学,2017.

[3]凌杰. 基于机器学习的小额借贷违约预测[D].西南大学,2021.

[4]逯瑶瑶. 基于机器学习分类算法的贷款违约预测研究[D].兰州大学,2021.

[5]许可. 基于机器学习混合模型的网贷违约风险识别研究[D].华南理工大学,2020.

[6]欧阳梦倩.基于机器学习算法的网络小额借款项目评估与筛选[J].管理现化,2022.

[7]张佳倩,李伟,阮素梅.基于机器学习的贷款违约风险预测[J].长春理工大学学报(社会版),2021,34(04):105-111.

**代码和数据**

见附件；

|  |  |
| --- | --- |
| 模型.ipynb | 模型的可执行代码 |
| 模型.pdf | 模型输出的pdf格式 |
| 数据来源 | <https://tianchi.aliyun.com/competition/entrance/531830/information> |

Jupyted导出的python脚本文件（具体执行文件可见附件模型.ipynb或模型.pdf）：

|  |
| --- |
| # %%  '''导入一些必要的包'''  import warnings  warnings.filterwarnings("ignore")  import numpy as np  import pandas as pd  import seaborn as sns  import matplotlib.pyplot as plt  import statsmodels.formula.api as smf  import xgboost as xgb  from sklearn.model\_selection import train\_test\_split  from sklearn.model\_selection import KFold  plt.rcParams["font.sans-serif"]=["SimHei"]  plt.rcParams["axes.unicode\_minus"]=False  # %% [markdown]  # ##数据描述及探索性分析  # %%  '''导入数据'''  train = pd.read\_csv('train.csv')  test = pd.read\_csv('testA.csv')  # %%  '''样本个数和特征维度'''  train.shape  # %%  '''样本个数和特征维度'''  test.shape  # %%  '''样本的特征'''  train.columns  # %%  '''数据集的一些基本信息（缺失情况、类型等）'''  train.info()  # %%  '''数据的描述性分析'''  # train.describe()  train.describe().T.head()  # %%  '''数值类型（连续变量、离散型变量和单值变量）'''  numerical\_feature = list(train.select\_dtypes(exclude=['object']).columns)  print(len(numerical\_feature)) ## 42  numerical\_feature  # %%  # 连续型变量  serial\_feature = []  # 离散型变量  discrete\_feature = []  # 单值变量  unique\_feature = []  for fea in numerical\_feature:  temp = train[fea].nunique()# 返回的是唯一值的个数  if temp == 1:  unique\_feature.append(fea)  # 自定义变量的值的取值个数小于10就为离散型变量  elif temp <= 10:  discrete\_feature.append(fea)  else:  serial\_feature.append(fea)  # %%  '''连续型变量'''  serial\_feature  # %%  '''每个数字特征得分布可视化'''  f = pd.melt(train, value\_vars=serial\_feature)  g = sns.FacetGrid(f, col="variable", col\_wrap=4, sharex=False, sharey=False,palette="Set1")  g = g.map(sns.distplot, "value")  # %%  '''贷款金额 loanAmnt 的分布情况'''  plt.figure(1 , figsize = (8 , 5))  sns.distplot(train.loanAmnt,bins=40)  plt.xlabel('loanAmnt')  # %%  '''违约与不违约两类样本的贷款金额分布情况'''  label=train['isDefault']  sns.kdeplot(train.loanAmnt[label[label==1].index], label='1', shade=True)#违约  sns.kdeplot(train.loanAmnt[label[label==0].index], label='0', shade=True)#没有违约  plt.xlabel('loanAmnt')  plt.ylabel('Density')  # %%  '''年收入的分布情况'''  plt.figure(1 , figsize = (8 , 5))  sns.distplot(train['annualIncome'])  plt.xlabel('annualIncome')  # %%  '''离散型变量'''  discrete\_feature  # %%  for f in discrete\_feature:  print(f, '类型数：', train[f].nunique())  # %%  '''离散型特征可视化呈现'''  df\_ = train[discrete\_feature]  sns.set\_style("whitegrid") # 使用whitegrid主题  fig,axes=plt.subplots(nrows=4,ncols=2,figsize=(8,10))  for i, item in enumerate(df\_):  plt.subplot(4,2,(i+1))  #ax=df[item].value\_counts().plot(kind = 'bar')  ax=sns.countplot(item,data = df\_,palette="Pastel1")  plt.xlabel(str(item),fontsize=14)  plt.ylabel('Count',fontsize=14)  plt.xticks(fontsize=13)  plt.yticks(fontsize=13)  #plt.title("Churn by "+ str(item))  i=i+1  plt.tight\_layout()  plt.show()  # %%  '''单值变量'''  unique\_feature  # %%  '''分类型特征'''  category\_feature = list(filter(lambda x: x not in numerical\_feature,list(train.columns)))  category\_feature  # %%  '''分类型特征的结构'''  train[category\_feature]  # %%  '''分类型特征可视化呈现'''  df\_category = train[['grade', 'subGrade']]  sns.set\_style("whitegrid") # 使用whitegrid主题  color = sns.color\_palette()  fig,axes=plt.subplots(nrows=2,ncols=1,figsize=(10,10))  for i, item in enumerate(df\_category):  plt.subplot(2,1,(i+1))  #ax=df[item].value\_counts().plot(kind = 'bar')  ax=sns.countplot(item,data = df\_category)  plt.xlabel(str(item),fontsize=14)  plt.ylabel('Count',fontsize=14)  plt.xticks(fontsize=13)  plt.yticks(fontsize=13)  #plt.title("Churn by "+ str(item))  i=i+1  plt.tight\_layout()  plt.show()  # %%  '''分类型特征可视化呈现'''  plt.figure(1 , figsize = (10 , 8))  sns.barplot(train["employmentLength"].value\_counts(dropna=False),  train["employmentLength"].value\_counts(dropna=False).keys())  plt.xticks(fontsize=13)  plt.yticks(fontsize=13)  plt.xlabel('employmentLength',fontsize=14)  plt.show()  # %%  '''统计一下issueDate与earliesCreditLine类别的数量'''  for i in train[['issueDate', 'earliesCreditLine']]:  print(train[i].value\_counts())  print()  # %%  '''查看目标变量（标签）是否平衡'''  abel=train.isDefault  label.value\_counts()/len(label)  # %%  '''目标变量可视化'''  sns.countplot(label)  # %%  '''目标变量和分类类别之间的分布关系'''  train\_loan\_fr = train.loc[train['isDefault'] == 1]  train\_loan\_nofr = train.loc[train['isDefault'] == 0]  fig, ((ax1, ax2), (ax3, ax4)) = plt.subplots(2, 2, figsize=(15, 8))  # 目标变量为1时候grade的分布  train\_loan\_fr.groupby("grade").size().plot.bar(ax=ax1)  # 目标变量为0时候grade的分布  train\_loan\_nofr.groupby("grade")["grade"].count().plot.bar(ax=ax2)  # 目标变量为1时候employmentLength的分布  train\_loan\_fr.groupby("employmentLength").size().plot.bar(ax=ax3)  # 目标变量为0时候employmentLength的分布  train\_loan\_nofr.groupby("employmentLength")["employmentLength"].count().plot.bar(ax=ax4)  plt.xticks(rotation=90)  # %%  '''正负样本的数据差异'''  train\_positve = train[train['isDefault'] == 1]  train\_negative = train[train['isDefault'] != 1]  f, ax = plt.subplots(len(numerical\_feature),2,figsize = (10,80))  for i,col in enumerate(numerical\_feature):  sns.distplot(train\_positve[col],ax = ax[i,0],color = "blue")  ax[i,0].set\_title("positive")  sns.distplot(train\_negative[col],ax = ax[i,1],color = 'red')  ax[i,1].set\_title("negative")  plt.subplots\_adjust(hspace = 1)  # %%  '''缺省值查看'''  # 去掉标签  X\_missing = train.drop(['isDefault'],axis=1)  # 查看缺失情况  missing = X\_missing.isna().sum()  missing = pd.DataFrame(data={'特征': missing.index,'缺失值个数':missing.values})  #通过~取反，选取不包含数字0的行  missing = missing[~missing['缺失值个数'].isin([0])]  # 缺失比例  missing['缺失比例'] = missing['缺失值个数']/X\_missing.shape[0]  missing  # %%  '''缺失值可视化'''  (train.isnull().sum()/len(train)).plot.bar(figsize = (20,6),color=['#d6ecf0','#a3d900','#88ada6','#ffb3a7','#cca4e3','#a1afc9'])  # %%  '''数据相关关系'''  f, ax = plt.subplots(1,1, figsize = (20,20))  cor = train[numerical\_feature].corr()  sns.heatmap(cor, annot = True, linewidth = 0.2, linecolor = "white", ax = ax, fmt =".1g" )  # %% [markdown]  # ##特征工程  # %%  '''重复值'''  train.duplicated().sum()  # %%  '''缺失值填补'''  label = 'isDefault'  Y\_label = train['isDefault']  numerical\_feature.remove(label)  # 训练集  train[numerical\_feature] = train[numerical\_feature].fillna(train[numerical\_feature].median())  # 测试集  test[numerical\_feature] = test[numerical\_feature].fillna(train[numerical\_feature].median())  # %%  '''分类型特征查看'''  train[category\_feature]  # %%  '''分类型特征用众数填补'''  # 训练集  train[category\_feature] = train[category\_feature].fillna(train[category\_feature].mode())  # 测试集  test[category\_feature] = test[category\_feature].fillna(train[category\_feature].mode())  # %%  '''查看缺失值情况'''  '''employmentLength 列还存在缺失值'''  train.isnull().sum()  # %%  '''employmentLength的分布'''  train.employmentLength  # %%  '''采用决策树来填补就业年限（employmentLength）'''  from sklearn.tree import DecisionTreeClassifier  empLenNotNullInd = train.employmentLength.notnull() # 不是空的行，返回True  columns = ['postCode','regionCode','employmentTitle','annualIncome'] # 用四个特征来预测employmentLength  train\_empLen\_X = train.loc[empLenNotNullInd,columns]  train\_empLen\_y = train.employmentLength[empLenNotNullInd]  DTC = DecisionTreeClassifier() # 实例化  DTC.fit(train\_empLen\_X ,train\_empLen\_y) # 训练  print(DTC.score(train\_empLen\_X ,train\_empLen\_y))# 0.9809320486828881  # 预测  for data in [train,test]:  empLenIsNullInd = data.employmentLength.isnull()  test\_empLen\_X = data.loc[empLenIsNullInd,columns]  empLen\_pred = DTC.predict(test\_empLen\_X)  data.employmentLength[empLenIsNullInd] = empLen\_pred  # %%  '''空值数量'''  train.isnull().any().sum()  # %%  train['employmentLength'][:20]  '''  0 2 years  1 5 years  2 8 years  3 10+ years  4 5 years  5 7 years  6 9 years  7 1 year  8 5 years  9 6 years  10 10+ years  11 3 years  12 2 years  13 10+ years  14 2 years  15 2 years  16 9 years  17 < 1 year  18 10+ years  19 9 years  Name: employmentLength, dtype: object  '''  # %%  train['employmentLength'].value\_counts(dropna=False).sort\_index()  '''  1 year 55034  10+ years 276853  2 years 76435  3 years 68888  4 years 50893  5 years 54038  6 years 39517  7 years 37175  8 years 37903  9 years 31463  < 1 year 71801  Name: employmentLength, dtype: int64  '''  # %%  '''时间数据处理'''  import datetime  # 转化成时间格式 issueDateDT特征表示数据日期离数据集中日期最早的日期（2007-06-01）的天数  train['issueDate'] = pd.to\_datetime(train['issueDate'],format='%Y-%m-%d')  startdate = datetime.datetime.strptime('2007-06-01', '%Y-%m-%d')  train['issueDateDT'] = train['issueDate'].apply(lambda x: x-startdate).dt.days  # %%  '''处理效果'''  train[['issueDate','issueDateDT']]  # %%  '''测试集时间数据处理'''  #转化成时间格式  test['issueDate'] = pd.to\_datetime(train['issueDate'],format='%Y-%m-%d')  startdate = datetime.datetime.strptime('2007-06-01', '%Y-%m-%d')  test['issueDateDT'] = test['issueDate'].apply(lambda x: x-startdate).dt.days  # %%  '''处理效果可视化'''  plt.figure(1 , figsize = (10 , 8))  plt.hist(train['issueDateDT'], label='train');  plt.hist(test['issueDateDT'], label='test');  plt.legend();  plt.title('Distribution of issueDateDT dates');  #train 和 test issueDateDT 日期有重叠 所以使用基于时间的分割进行验证是不明智的  # %%  train[['issueDate','earliesCreditLine']]  '''  issueDate earliesCreditLine  0 2014-07-01 Aug-2001  1 2012-08-01 May-2002  2 2015-10-01 May-2006  3 2015-08-01 May-1999  4 2016-03-01 Aug-1977  ... ...  799995 2016-07-01 Aug-2011  799996 2013-04-01 May-1989  799997 2015-10-01 Jul-2002  799998 2015-02-01 Jan-1994  799999 2018-08-01 Feb-2002  [800000 rows x 2 columns]  '''  # %%  '''特征交叉'''  train\_earliesCreditLine\_year = train['earliesCreditLine'].apply(lambda x:x[-4:]).astype('int64')  test\_earliesCreditLine\_year = test['earliesCreditLine'].apply(lambda x:x[-4:]).astype('int64')  train\_issueDate\_year = train['issueDate'].astype('str').apply(lambda x:x[:4]).astype('int64')  test\_issueDate\_year = test['issueDate'].astype('str').apply(lambda x:x[:4]).astype('int64')  train['CreditLine'] = train\_issueDate\_year - train\_earliesCreditLine\_year  test['CreditLine'] = test\_issueDate\_year - test\_earliesCreditLine\_year  train = train.drop(['earliesCreditLine','issueDate'],axis=1)  test = test.drop(['earliesCreditLine','issueDate'],axis=1)  # %%  '''特征交叉结果'''  train['CreditLine']  '''  CreditLine  0 13  1 10  2 9  3 16  4 39  ..  799995 5  799996 24  799997 13  799998 21  799999 16  Name: CreditLine, Length: 800000, dtype: int64  '''  # %%  # %%  '''特征编码'''  def employmentLength\_to\_int(s):  if pd.isnull(s):  return s  else:  return np.int8(s.split()[0])  for data in [train, test]:  data['employmentLength'].replace(to\_replace='10+ years', value='10 years', inplace=True)  data['employmentLength'].replace('< 1 year', '0 years', inplace=True)  data['employmentLength'] = data['employmentLength'].apply(employmentLength\_to\_int)  # %%  train['employmentLength'][:20]  '''  0 2 years  1 5 years  2 8 years  3 10+ years  4 5 years  5 7 years  6 9 years  7 1 year  8 5 years  9 6 years  10 10+ years  11 3 years  12 2 years  13 10+ years  14 2 years  15 2 years  16 9 years  17 < 1 year  18 10+ years  19 9 years  Name: employmentLength, dtype: object  '''  # %%  train['employmentLength']  '''  0 2  1 5  2 8  3 10  4 5  ..  799995 7  799996 10  799997 10  799998 10  799999 5  Name: employmentLength, Length: 800000, dtype: int64  '''  # %%  a2z = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'  a2z\_code = np.arange(1,27)  a2z\_mapping = dict(zip(a2z, a2z\_code))  for data in [train,test]:  data.loc[:,['grade','subGrade']] = data.loc[:,['grade','subGrade']].applymap(lambda g:g.replace(g[0], str(a2z.index(g[0])+1))).astype('int')  # %%  train[['grade','subGrade']]  # %%  train[['grade','subGrade']]  # %%  train[['homeOwnership','verificationStatus','purpose']]  '''  homeOwnership verificationStatus purpose  0 2 2 1  1 0 2 0  2 0 2 0  3 1 1 4  4 1 2 10  ... ... ...  799995 1 0 0  799996 0 2 4  799997 1 2 0  799998 0 2 4  799999 0 0 4  [800000 rows x 3 columns]  '''  # %%  train.shape# (800000, 47)  # %%  '''独热编码'''  from sklearn.preprocessing import OneHotEncoder  oh = OneHotEncoder(sparse=False)  oh.fit(train[['homeOwnership','verificationStatus','purpose']])  OneHot1 = oh.transform(train[['homeOwnership','verificationStatus','purpose']])  OneHot2 = oh.transform(test[['homeOwnership','verificationStatus','purpose']])  OneHot1.shape# (800000, 23)  '''  array([[0., 0., 1., ..., 0., 0., 0.],  [1., 0., 0., ..., 0., 0., 0.],  [1., 0., 0., ..., 0., 0., 0.],  ...,  [0., 1., 0., ..., 0., 0., 0.],  [1., 0., 0., ..., 0., 0., 0.],  [1., 0., 0., ..., 0., 0., 0.]])  '''  train = pd.concat([train, pd.DataFrame(OneHot1)], axis=1)  test = pd.concat([test, pd.DataFrame(OneHot2)], axis=1)  train = train.drop(['homeOwnership','verificationStatus','purpose'],axis=1)  test = test.drop(['homeOwnership','verificationStatus','purpose'],axis=1)  train.shape# (800000, 67)  # %%  train['loanAmnt\_bin3'] = pd.qcut(train['loanAmnt'], 10, labels=False)  # %%  train=train.drop(["id"],axis=1)  train.shape # (800000, 66)  # %%  test=test.drop(["id"],axis=1)  test.shape # (200000, 65)  # %%  train.corr()["isDefault"].sort\_values()  # %%  train=train.drop(["initialListStatus","n5","n11","n12","n8","postCode","policyCode"],axis=1)  test=test.drop(["initialListStatus","n5","n11","n12","n8","postCode","policyCode"],axis=1)  train.shape# (800000, 59)  # %%  # 显示相关性高于0.6的变量  def getHighRelatedFeatureDf(corr\_matrix, corr\_threshold):  highRelatedFeatureDf = pd.DataFrame(corr\_matrix[corr\_matrix>corr\_threshold].stack().reset\_index())  highRelatedFeatureDf.rename({'level\_0':'feature\_x', 'level\_1':'feature\_y', 0:'corr'}, axis=1, inplace=True)  highRelatedFeatureDf = highRelatedFeatureDf[highRelatedFeatureDf.feature\_x != highRelatedFeatureDf.feature\_y]  highRelatedFeatureDf['feature\_pair\_key'] = highRelatedFeatureDf.loc[:,['feature\_x', 'feature\_y']].apply(lambda r:'#'.join(np.sort(r.values)), axis=1)  highRelatedFeatureDf.drop\_duplicates(subset=['feature\_pair\_key'],inplace=True)  highRelatedFeatureDf.drop(['feature\_pair\_key'], axis=1, inplace=True)  return highRelatedFeatureDf  getHighRelatedFeatureDf(train.corr(),0.6)  '''  feature\_x feature\_y corr  2 loanAmnt installment 0.953369  5 interestRate grade 0.953269  6 interestRate subGrade 0.970847  11 grade subGrade 0.993907  22 delinquency\_2years n13 0.658946  24 ficoRangeLow ficoRangeHigh 1.000000  28 openAcc totalAcc 0.700796  29 openAcc n2 0.658807  30 openAcc n3 0.658807  31 openAcc n4 0.618207  32 openAcc n7 0.830624  33 openAcc n8 0.646342  34 openAcc n9 0.660917  35 openAcc n10 0.998717  37 pubRec pubRecBankruptcies 0.644402  44 totalAcc n5 0.623639  45 totalAcc n6 0.678482  46 totalAcc n8 0.761854  47 totalAcc n10 0.697192  53 n1 n2 0.807789  54 n1 n3 0.807789  55 n1 n4 0.829016  56 n1 n7 0.651852  57 n1 n9 0.800925  61 n2 n3 1.000000  62 n2 n4 0.663186  63 n2 n7 0.790337  64 n2 n9 0.982015  65 n2 n10 0.655296  70 n3 n4 0.663186  71 n3 n7 0.790337  72 n3 n9 0.982015  73 n3 n10 0.655296  79 n4 n5 0.717936  80 n4 n7 0.742157  81 n4 n9 0.639867  82 n4 n10 0.614658  86 n5 n7 0.618970  87 n5 n8 0.838066  97 n7 n8 0.774955  98 n7 n9 0.794465  99 n7 n10 0.829799  105 n8 n10 0.640729  113 n9 n10 0.660395  '''  # %%  col = ['installment','ficoRangeHigh','openAcc','n3','n9']  for data in [train,test]:  data.drop(col,axis=1,inplace=True)  # %%  train.shape # (800000, 54)  # %%  train.var().sort\_values()  # %%  col = ['applicationType']  for data in [train,test]:  data.drop(col,axis=1,inplace=True)  # %%  train.shape # (800000, 53)    # %%  label=train['isDefault']  label.value\_counts()/len(label)  '''  0 0.800488  1 0.199513  Name: isDefault, dtype: float64  '''  # %%  import imblearn  from imblearn.over\_sampling import SMOTE  over\_samples = SMOTE(random\_state=2022,k\_neighbors=5)  train\_over,label\_over = over\_samples.fit\_resample(train, label)  train\_over.to\_csv('train\_over.csv',index=False)  label\_over.to\_csv('label\_over.csv',index=False)  print(label\_over.value\_counts()/len(label\_over))  print(train\_over.shape)  # %%  from imblearn.under\_sampling import RandomUnderSampler  under\_samples = RandomUnderSampler(random\_state=1234)  train\_under, label\_under = under\_samples.fit\_resample(train,label)  train\_under.to\_csv('train\_under.csv',index=False)  label\_under.to\_csv('label\_under.csv',index=False)  print(label\_under.value\_counts()/len(label\_under))  print(train\_under.shape)  # %% [markdown]  # ##建模分析  # %%  from sklearn import metrics  from sklearn.metrics import roc\_auc\_score  """原数据的模型训练"""  X = train.drop(['isDefault'], axis=1)  y = train.loc[:,'isDefault']  Xtrain, Xtest, Ytrain, Ytest = train\_test\_split(X,y,test\_size=0.25,random\_state=2022)  from xgboost.sklearn import XGBClassifier  clf1 = XGBClassifier(n\_jobs=-1)  clf1.fit(Xtrain,Ytrain)  print("模型评分:{}".format(clf1.score(Xtest,Ytest)))  print("查准率：",metrics.precision\_score(Ytest, np.round(clf1.predict(Xtest))))  print("召回率：",metrics.recall\_score(Ytest, np.round(clf1.predict(Xtest))))  print("准确率：",metrics.accuracy\_score(Ytest, np.round(clf1.predict(Xtest))))  print("F1分数：",metrics.f1\_score(Ytest, np.round(clf1.predict(Xtest))))  print("AUC: ",roc\_auc\_score(Ytest, clf1.predict\_proba(Xtest)[:,1]))  """画出roc曲线图"""  plt.figure(figsize=(8, 8))  fpr, tpr, threshold = metrics.roc\_curve(Ytest, clf1.predict\_proba(Xtest)[:,1])  roc\_auc = metrics.auc(fpr, tpr)  plt.plot(fpr, tpr, 'b', label = 'Val AUC = %0.4f' % roc\_auc)  plt.ylim(-0.01,1)  plt.xlim(-0.01,1)  plt.legend(loc = 'lower right')  plt.title('ROC')  plt.ylabel('True Positive Rate')  plt.xlabel('False Positive Rate')  # 画出对角线  plt.plot([0,1],[0,1],'r--')  plt.show()  # from sklearn.metrics import roc\_curve, auc  # predict\_proba = clf1.predict\_proba(Xtest)  # false\_positive\_rate, true\_positive\_rate, thresholds = roc\_curve(Ytest, predict\_proba[:,1])  # print("AUC:{}".format(auc(false\_positive\_rate, true\_positive\_rate)))  # %%  # plot feature importance using built-in function  from xgboost import plot\_importance  # plot feature importance  plot\_importance(clf1, max\_num\_features=10, height=0.5, title='Feature importance' )  plt.show()  # %%  """上抽样数据的模型训练"""  X1 = train\_over.drop(['isDefault'], axis=1)  y1 = train\_over.loc[:,'isDefault']  X1train, X1test, Y1train, Y1test = train\_test\_split(X1,y1,test\_size=0.25,random\_state=2022)  from xgboost.sklearn import XGBClassifier  clf2 = XGBClassifier(n\_jobs=-1)  clf2.fit(X1train,Y1train)  print("模型评分:{}".format(clf2.score(X1test,Y1test)))  print("查准率：",metrics.precision\_score(Y1test, np.round(clf2.predict\_proba(X1test)[:,1])))  print("召回率：",metrics.recall\_score(Y1test, np.round(clf2.predict\_proba(X1test)[:,1])))  print("准确率：",metrics.accuracy\_score(Y1test, np.round(clf2.predict\_proba(X1test)[:,1])))  print("F1分数：",metrics.f1\_score(Y1test, np.round(clf2.predict\_proba(X1test)[:,1])))  print("AUC: {}".format(roc\_auc\_score(Y1test, clf2.predict\_proba(X1test)[:,1])))  # from sklearn.metrics import roc\_curve, auc  # predict\_proba = clf2.predict\_proba(X1test)  # false\_positive\_rate, true\_positive\_rate, thresholds = roc\_curve(Y1test, predict\_proba[:,1])  # print("AUC:{}".format(auc(false\_positive\_rate, true\_positive\_rate)))  """画出roc曲线图"""  fpr, tpr, threshold = metrics.roc\_curve(Y1test, clf2.predict\_proba(X1test)[:,1])  roc\_auc = metrics.auc(fpr, tpr)  plt.figure(figsize=(8, 8))  plt.plot(fpr, tpr, 'b', label = 'Val AUC = %0.4f' % roc\_auc)  plt.ylim(-0.01,1)  plt.xlim(-0.01,1)  plt.legend(loc = 'best')  plt.title('ROC')  plt.ylabel('True Positive Rate')  plt.xlabel('False Positive Rate')  # 画出对角线  plt.plot([0,1],[0,1],'r--')  plt.show()  # %%  """下抽样数据的模型训练"""  from xgboost.sklearn import XGBClassifier  clf3 = XGBClassifier(n\_jobs=-1)  clf3.fit(train\_under.drop(['isDefault'], axis=1),label\_under)  clf3.score(train\_under.drop(['isDefault'], axis=1),label\_under)  from sklearn.metrics import roc\_curve, auc  predict\_proba = clf3.predict\_proba(X1test)  false\_positive\_rate, true\_positive\_rate, thresholds = roc\_curve(Y1test, predict\_proba[:,1])  print("模型评分：",clf3.score(train\_under.drop(['isDefault'], axis=1),label\_under))  print("查准率：",metrics.precision\_score(Y1test, np.round(clf3.predict\_proba(X1test)[:,1])))  print("召回率：",metrics.recall\_score(Y1test, np.round(clf3.predict\_proba(X1test)[:,1])))  print("准确率：",metrics.accuracy\_score(Y1test, np.round(clf3.predict\_proba(X1test)[:,1])))  print("F1分数：",metrics.f1\_score(Y1test, np.round(clf3.predict\_proba(X1test)[:,1])))  print("AUC: {}".format(roc\_auc\_score(Y1test, clf3.predict\_proba(X1test)[:,1])))  # from sklearn.metrics import roc\_curve, auc  # predict\_proba = clf2.predict\_proba(X1test)  # false\_positive\_rate, true\_positive\_rate, thresholds = roc\_curve(Y1test, predict\_proba[:,1])  # print("AUC:{}".format(auc(false\_positive\_rate, true\_positive\_rate)))  """画出roc曲线图"""  fpr, tpr, threshold = metrics.roc\_curve(Y1test, clf3.predict\_proba(X1test)[:,1])  roc\_auc = metrics.auc(fpr, tpr)  plt.figure(figsize=(8, 8))  plt.plot(fpr, tpr, 'b', label = 'Val AUC = %0.4f' % roc\_auc)  plt.ylim(-0.01,1)  plt.xlim(-0.01,1)  plt.legend(loc = 'lower right')  plt.title('ROC')  plt.ylabel('True Positive Rate')  plt.xlabel('False Positive Rate')  # 画出对角线  plt.plot([0,1],[0,1],'r--')  plt.show()  # %% [markdown]  # ##调参  # %%  cv\_params = {'max\_depth': [ 8, 9, 10], 'min\_child\_weight': [1, 2, 3, 4, 5, 6]}  other\_params = {'learning\_rate': 0.1, 'n\_estimators': 10, 'max\_depth': 5, 'min\_child\_weight':4, 'seed': 0,  'subsample': 0.8, 'colsample\_bytree': 0.8, 'gamma': 0, 'reg\_alpha': 0, 'reg\_lambda': 1}  model =XGBClassifier(\*\*other\_params)  optimized\_GBM = GridSearchCV(estimator=model, param\_grid=cv\_params, scoring='accuracy', cv=5, verbose=1, n\_jobs=4)  optimized\_GBM.fit(X1train,Y1train)  # evalute\_result = optimized\_GBM.grid\_scores\_  # print('每轮迭代运行结果:{0}'.format(evalute\_result))  print('参数的最佳取值：{0}'.format(optimized\_GBM.best\_params\_))  print('最佳模型得分:{0}'.format(optimized\_GBM.best\_score\_))  # %%  cv\_params = {'gaama':[0,0.05,0.1,0.2,0.3]}  other\_params = {'learning\_rate': 0.1, 'n\_estimators': 10, 'max\_depth': 5, 'min\_child\_weight':4, 'seed': 0,  'subsample': 0.8, 'colsample\_bytree': 0.8, 'gamma': 0, 'reg\_alpha': 0, 'reg\_lambda': 1}  model =XGBClassifier(\*\*other\_params)  optimized\_GBM = GridSearchCV(estimator=model, param\_grid=cv\_params, scoring='accuracy', cv=5, verbose=1, n\_jobs=4)  optimized\_GBM.fit(X1train,Y1train)  # evalute\_result = optimized\_GBM.grid\_scores\_  # print('每轮迭代运行结果:{0}'.format(evalute\_result))  print('参数的最佳取值：{0}'.format(optimized\_GBM.best\_params\_))  print('最佳模型得分:{0}'.format(optimized\_GBM.best\_score\_))  # %%  print('参数的最佳取值：{0}'.format(optimized\_GBM.best\_params\_))  print('最佳模型得分:{0}'.format(optimized\_GBM.best\_score\_))  # %%  cv\_params = {'subsample': [0.6, 0.7, 0.8, 0.9], 'colsample\_bytree': [0.6, 0.7, 0.8, 0.9]}  other\_params = {'learning\_rate': 0.1, 'n\_estimators':10, 'max\_depth': 10, 'min\_child\_weight':3, 'seed': 0,  'subsample': 0.8, 'colsample\_bytree': 0.8, 'gamma': 0, 'reg\_alpha': 0, 'reg\_lambda': 1}  model = XGBClassifier(\*\*other\_params)  optimized\_GBM = GridSearchCV(estimator=model, param\_grid=cv\_params, scoring='accuracy', cv=5, verbose=1, n\_jobs=4)  optimized\_GBM.fit(X1train,Y1train)  # evalute\_result = optimized\_GBM.grid\_scores\_  #print('每轮迭代运行结果:{0}'.format(evalute\_result))  print('参数的最佳取值：{0}'.format(optimized\_GBM.best\_params\_))  print('最佳模型得分:{0}'.format(optimized\_GBM.best\_score\_))  # %%  from sklearn.model\_selection import GridSearchCV  from xgboost.sklearn import XGBClassifier  cv\_params = {'reg\_alpha': [0.05, 0.1, 0.5,1, 2], 'reg\_lambda': [0.1, 0.5, 1, 2, ]}  other\_params = {'learning\_rate': 0.1, 'n\_estimators':10, 'max\_depth': 10, 'min\_child\_weight':3, 'seed': 0,  'subsample': 0.7, 'colsample\_bytree': 0.6, 'gamma': 0, 'reg\_alpha': 0, 'reg\_lambda': 1}  model = XGBClassifier(\*\*other\_params)  optimized\_GBM = GridSearchCV(estimator=model, param\_grid=cv\_params, scoring='accuracy', cv=5, verbose=1, n\_jobs=4)  optimized\_GBM.fit(X1train,Y1train)  # evalute\_result = optimized\_GBM.grid\_scores\_  #print('每轮迭代运行结果:{0}'.format(evalute\_result))  print('参数的最佳取值：{0}'.format(optimized\_GBM.best\_params\_))  print('最佳模型得分:{0}'.format(optimized\_GBM.best\_score\_))  # %%  cv\_params = {'learning\_rate': [0.01, 0.05, 0.07, 0.1, 0.2]}  other\_params = {'learning\_rate': 0.1, 'n\_estimators':50, 'max\_depth': 10, 'min\_child\_weight':3, 'seed': 0,  'subsample': 0.6, 'colsample\_bytree': 0.9, 'gamma': 0.1, 'reg\_alpha': 0.5, 'reg\_lambda': 0.1}  model = XGBClassifier(\*\*other\_params)  optimized\_GBM = GridSearchCV(estimator=model, param\_grid=cv\_params, scoring='accuracy', cv=5, verbose=1, n\_jobs=4)  optimized\_GBM.fit(Xtrain,Ytrain)  # evalute\_result = optimized\_GBM.grid\_scores\_  #print('每轮迭代运行结果:{0}'.format(evalute\_result))  print('参数的最佳取值：{0}'.format(optimized\_GBM.best\_params\_))  print('最佳模型得分:{0}'.format(optimized\_GBM.best\_score\_))  # %%  from sklearn.model\_selection import GridSearchCV  cv\_params = {'n\_estimators': [100,300,600,1000]}  other\_params = {'learning\_rate': 0.1, 'n\_estimators':50, 'max\_depth': 10, 'min\_child\_weight':3, 'seed': 0,  'subsample': 0.6, 'colsample\_bytree': 0.9, 'gamma': 0.1, 'reg\_alpha': 0.5, 'reg\_lambda': 0.1}  model = XGBClassifier(\*\*other\_params)  optimized\_GBM = GridSearchCV(estimator=model, param\_grid=cv\_params, scoring='accuracy', cv=5, verbose=1, n\_jobs=4)  best\_model1=optimized\_GBM .fit(X1train,Y1train)  # evalute\_result = optimized\_GBM.grid\_scores  # print('每轮迭代运行结果:{0}'.format(evalute\_result))  print('参数的最佳取值：{}'.format(optimized\_GBM.best\_params\_))  print('最佳模型得分:{}'.format(optimized\_GBM.best\_score\_))  # %%  # other\_params = {'learning\_rate': 0.1, 'n\_estimators':50, 'max\_depth': 10, 'min\_child\_weight':3, 'seed': 0,  # 'subsample': 0.6, 'colsample\_bytree': 0.9, 'gamma': 0.1, 'reg\_alpha': 0.5, 'reg\_lambda': 0.1}  from xgboost.sklearn import XGBClassifier  clf4 = XGBClassifier(  learning\_rate= 0.1,  n\_estimators= 1000,  max\_depth= 10,  min\_child\_weight= 3,  seed= 0,  subsample= 0.9,  colsample\_bytree= 0.7,  gamma= 0,  reg\_alpha=0.5,  reg\_lambda=0.1,  objective= 'binary:logistic',  )  # X2 = train\_under.drop(['isDefault'], axis=1)  # y2 = train\_under.loc[:,'isDefault']  Xtrain, Xtest, Ytrain, Ytest = train\_test\_split(X,y,test\_size=0.25,random\_state=2022)  clf4.fit(Xtrain,Ytrain)  y\_pred = clf4.predict(Xtest)  #auc  from sklearn.metrics import roc\_auc\_score  print("模型评分:{}".format(clf4.score(X1test,Y1test)))  print("查准率：",metrics.precision\_score(Y1test, np.round(clf4.predict\_proba(X1test)[:,1])))  print("召回率：",metrics.recall\_score(Y1test, np.round(clf4.predict\_proba(X1test)[:,1])))  print("准确率：",metrics.accuracy\_score(Y1test, np.round(clf4.predict\_proba(X1test)[:,1])))  print("F1分数：",metrics.f1\_score(Y1test, np.round(clf4.predict\_proba(X1test)[:,1])))  print("AUC: {}".format(roc\_auc\_score(Y1test, clf4.predict\_proba(X1test)[:,1])))  """画出roc曲线图"""  fpr, tpr, threshold = metrics.roc\_curve(Y1test, clf4.predict\_proba(X1test)[:,1])  roc\_auc = metrics.auc(fpr, tpr)  plt.figure(figsize=(8, 8))  plt.plot(fpr, tpr, 'b', label = 'Val AUC = %0.4f' % roc\_auc)  plt.ylim(-0.01,1)  plt.xlim(-0.01,1)  plt.legend(loc = 'lower right')  plt.title('ROC')  plt.ylabel('True Positive Rate')  plt.xlabel('False Positive Rate')  # 画出对角线  plt.plot([0,1],[0,1],'r--')  plt.show()  # %%  from xgboost.sklearn import XGBClassifier  clf4 = XGBClassifier(  learning\_rate= 0.1,  n\_estimators= 1000,  max\_depth= 10,  min\_child\_weight= 3,  seed= 0,  subsample= 0.9,  colsample\_bytree= 0.7,  gamma= 0,  reg\_alpha=0.5,  reg\_lambda=0.1,  objective= 'binary:logistic',  )  # X2 = train\_under.drop(['isDefault'], axis=1)  # y2 = train\_under.loc[:,'isDefault']  X1train, X1test, Y1train, Y1test = train\_test\_split(X1,y1,test\_size=0.25,random\_state=2022)  clf4.fit(X1train,Y1train)  y\_pred = clf4.predict(X1test)  #auc  from sklearn.metrics import roc\_auc\_score  print("模型评分:{}".format(clf4.score(X1test,Y1test)))  print("查准率：",metrics.precision\_score(Y1test, np.round(clf4.predict\_proba(X1test)[:,1])))  print("召回率：",metrics.recall\_score(Y1test, np.round(clf4.predict\_proba(X1test)[:,1])))  print("准确率：",metrics.accuracy\_score(Y1test, np.round(clf4.predict\_proba(X1test)[:,1])))  print("F1分数：",metrics.f1\_score(Y1test, np.round(clf4.predict\_proba(X1test)[:,1])))  print("AUC: {}".format(roc\_auc\_score(Y1test, clf4.predict\_proba(X1test)[:,1])))  """画出roc曲线图"""  fpr, tpr, threshold = metrics.roc\_curve(Y1test, clf4.predict\_proba(X1test)[:,1])  roc\_auc = metrics.auc(fpr, tpr)  plt.figure(figsize=(8, 8))  plt.plot(fpr, tpr, 'b', label = 'Val AUC = %0.4f' % roc\_auc)  plt.ylim(-0.01,1)  plt.xlim(-0.01,1)  plt.legend(loc = 'lower right')  plt.title('ROC')  plt.ylabel('True Positive Rate')  plt.xlabel('False Positive Rate')  # 画出对角线  plt.plot([0,1],[0,1],'r--')  plt.show()  # %%  from xgboost.sklearn import XGBClassifier  clf4 = XGBClassifier(  learning\_rate= 0.1,  n\_estimators= 1000,  max\_depth= 10,  min\_child\_weight= 3,  seed= 0,  subsample= 0.9,  colsample\_bytree= 0.7,  gamma= 0,  reg\_alpha=0.5,  reg\_lambda=0.1,  objective= 'binary:logistic',  )  X2 = train\_under.drop(['isDefault'], axis=1)  y2 = train\_under.loc[:,'isDefault']  X2train, X2test, Y2train, Y2test = train\_test\_split(X2,y2,test\_size=0.25,random\_state=2022)  clf4.fit(X2train,Y2train)  y\_pred = clf4.predict(X2test)  #auc  from sklearn.metrics import roc\_auc\_score  print("模型评分:{}".format(clf4.score(X1test,Y1test)))  print("查准率：",metrics.precision\_score(Y1test, np.round(clf4.predict\_proba(X1test)[:,1])))  print("召回率：",metrics.recall\_score(Y1test, np.round(clf4.predict\_proba(X1test)[:,1])))  print("准确率：",metrics.accuracy\_score(Y1test, np.round(clf4.predict\_proba(X1test)[:,1])))  print("F1分数：",metrics.f1\_score(Y1test, np.round(clf4.predict\_proba(X1test)[:,1])))  print("AUC: {}".format(roc\_auc\_score(Y1test, clf4.predict\_proba(X1test)[:,1])))  """画出roc曲线图"""  fpr, tpr, threshold = metrics.roc\_curve(Y1test, clf4.predict\_proba(X1test)[:,1])  roc\_auc = metrics.auc(fpr, tpr)  plt.figure(figsize=(8, 8))  plt.plot(fpr, tpr, 'b', label = 'Val AUC = %0.4f' % roc\_auc)  plt.ylim(-0.01,1)  plt.xlim(-0.01,1)  plt.legend(loc = 'lower right')  plt.title('ROC')  plt.ylabel('True Positive Rate')  plt.xlabel('False Positive Rate')  # 画出对角线  plt.plot([0,1],[0,1],'r--')  plt.show()  # %%  '''下采样数据的特征重要性'''  # plot feature importance using built-in function  from xgboost import plot\_importance  # plot feature importance  plot\_importance(clf4, max\_num\_features=10, height=0.5, title='Feature importance' )  plt.show()  # %%  '''原数据的特征重要性'''  from xgboost.sklearn import XGBClassifier  clf5 = XGBClassifier(  learning\_rate= 0.1,  n\_estimators= 1000,  max\_depth= 10,  min\_child\_weight= 3,  seed= 0,  subsample= 0.9,  colsample\_bytree= 0.7,  gamma= 0,  reg\_alpha=0.5,  reg\_lambda=0.1,  objective= 'binary:logistic',  )  X = train.drop(['isDefault'], axis=1)  y= train.loc[:,'isDefault']  Xtrain, Xtest, Ytrain, Ytest = train\_test\_split(X,y,test\_size=0.25,random\_state=2022)  clf5.fit(Xtrain,Ytrain)  y\_pred = clf5.predict(Xtest)  # plot feature importance using built-in function  from xgboost import plot\_importance  # plot feature importance  plot\_importance(clf5, max\_num\_features=10, height=0.5, title='Feature importance' )  plt.show() |