银行信贷违约检测

对不平衡数据集分类问题的研究

摘要

面对复杂的金融形势，如何更好地降低信用风险是一个值得关注的问题。本文讨论的问题是关于银行客户信用类型预测，面对数据的是高维稀疏特征以及样本不平衡，采用 AUC 作为模型评估指标，并用此建立合适的分类模型进行实际生产预测。

首先本文采用数据预处理和特征工程将数据进行清洗和加工，提高导入模型的数据质量，然后采用了 8 ：2 的比例划分训练集和测试集，同时使用 Xgboost、EasyEnsemble、BalanceCascade 三种算法进行建模和模型测试，用 AUC 作为衡量模型好坏的指标，最后对比不同模型的表现效果和时间成本，推荐最优算法模型运用于实际业务开展和模型构建过程。

本文的优势在于，在原数据特征基础上，进行了独热编码等，提高了特征在算法中的解释性，利于算法更好的拟合真实情况；同时使用了专门针对高维稀疏特征以及样本不平衡的三类算法，进一步缓解了数据本身存在的劣势。论文所得算法模型适用于我国银行信用风险监测领域，对金融机构更好降低信用风险具有一定的实际指导意义和价值。

关键词：不平衡数据、高维稀疏、AUC、Xgboost、EasyEnsemble、BalanceCascade

1　问题背景描述

信用风险是金融监管机构重点关注的风险，关乎金融系统运行的稳定。在实际业务开展和模型构建过程中，面临着高维稀疏特征以及样本不平衡等各种问题，如何应用机器学习等数据挖掘方法提高信用风险的评估和预测能力，是各家金融机构积极探索的方向

论文从 2019 厦门国际银行“数创金融杯”数据建模大赛中提供实际业务场景中的信贷数据作为建模的对象，所采用的数据集内容主要分为三个方面：用户基本属性信息、用户借贷相关信息以及用户征信相关信息，希望可以能借此运用所学的机器学习算法和数据挖掘的知识来锻炼和检验实战能力，同时也希望我们

的分类预测结果对于金融监管机构更好的保证金融系统运行的稳定性提供一定的实际指导意义。

2　数据预处理与特征工程

2.1　数据预处理过程

数据预处理有多种方法：数据清理，数据集成，数据变换，数据归约等。论文采用的数据集来源于真实数据，在建立模型之前对数据进行预处理是非常必要的步骤，数据预处理技术在数据挖掘之前使用，会大大提高了数据的质量，对于提升模型的效率有很重要的作用。

下面展示数据预处理的思路与过程：

2.1.1　数据集介绍

数据共分为两个数据集，训练集（train_x.csv、train_target.csv）和测试集（test_x.csv），其中 train_x.csv 为训练集的特征，train_target.csv 为训练集的目标变量，其中，为了增强模型的泛化能力，训练集由两个阶段的样本组成，由字段 isNew 标记。test_x.csv 为测试集的特征，特征变量与训练集一致。建模的目标即根据训练集对模型进行训练，并对测试集进行预测。三个数据表内容：

（1）训练集 train_x.csv 有 132029 行用户记录，其中有 104 个特征属性；

（2）训练集 train_target.csv 有 132029 行用户记录，其中有 1 个目标标签；

（3）测试集 test_x.csv 有 23561 行用户记录，其中有 104 个特征属性；

由此可见数据集是属于高维特征属性数据，其中数据集包含的属性和特征详细分为以下三种类型：

（Ⅰ）用户基本属性信息；

字段名称	中文解释	字段名称	中文解释
id	用户唯一标识	edu	学历
target	违约标识，1 违约、0 正常	job	单位类型
certId	证件号	ethnic	民族
gender	性别	highestEdu	最高学历
age	年龄	certValidBegin	证件号起始日
dist	地区	certValidStop	证件号失效日

（Ⅱ）用户借贷相关信息；

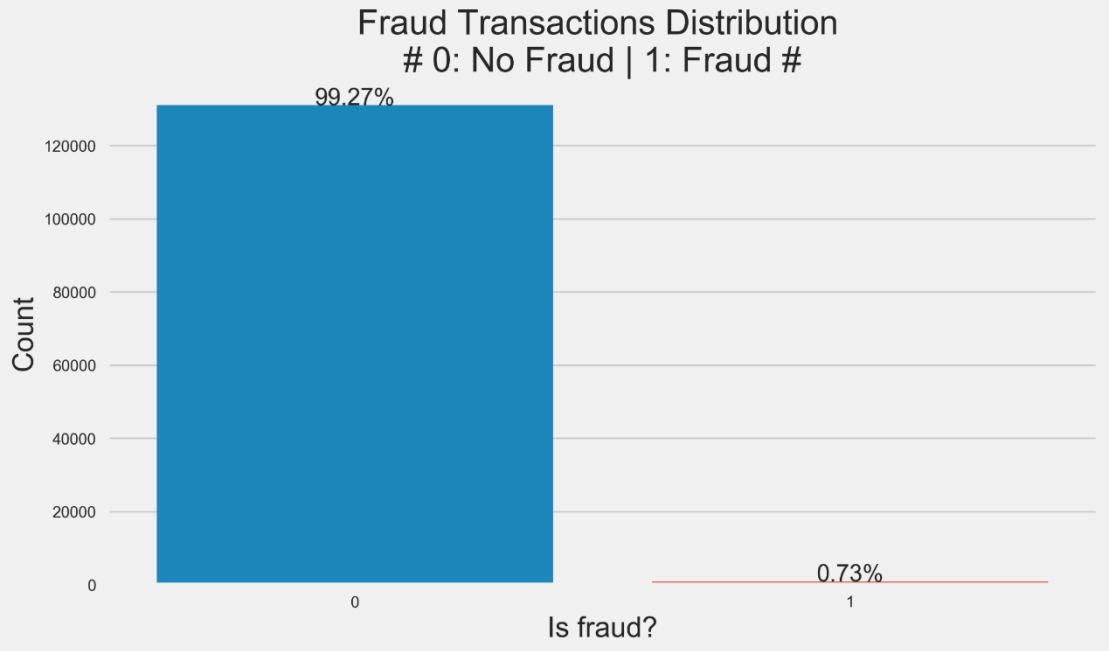
字段名称	中文解释	字段名称	中文解释
loanProduct	产品类型	residentAddr	居住地
lmt	预授信金额	linkRela	联系人关系
basicLevel	基础评级	setupHour	申请时段
bankCard	放款卡号	weekday,	申请日（周几）
isNew	是否新增数据		

（Ⅲ）用户征信相关信息；

X_0 至 x_78 以及 ncloseCreditCard, unpayIndvLoan, unpayOtherLoan, unpayNormalLoan, 5yearBadloan 该部分数据涉及较为第三方敏感数据，未做进一步说明。

对数据进行简单探索，可视化观察正负目标标签的数量，如下图所示：

图 1 正负标签可视化



计算得到正负标签之比：

(Y=0): (Y =1)= 136.67361835245046:1

由此可知，该二分类问题面对的是极端不平衡数据对象，同时存在高维特征属性，这两点是影响论文数据处理和算法选择的关键因素。

2.1.2 缺失值与重复值处理

上述简单的数据探索，对采用的数据集有一定初步认识，现在论文针对每个特征属性，进行详细的探索分析。

（1）重复值查找

由于数据量大和高维特征，考虑查找数据集中是否存在大量的重复值，常见大量重复值出现的原因有：重复记录。论文采用数据集中无重复数据。

（2）缺失值查找

数据集中存在的缺失值分为两种情况： NA 与-999。

a.仅“bankCard”字段含有 NA 型缺失值，而此特征对于分类并无太大作用，故去除。

b.对于表现为-999 的缺失值，在后续编码时分别进行不同的处理，具体将在后面部分介绍。

至此，仅仅处理了含有"缺失值"的列进行了处理,且数据没有重复数据。还存在变量内部还有另一种以-999 填补好的缺失值。

2.1.3 分类型特征处理

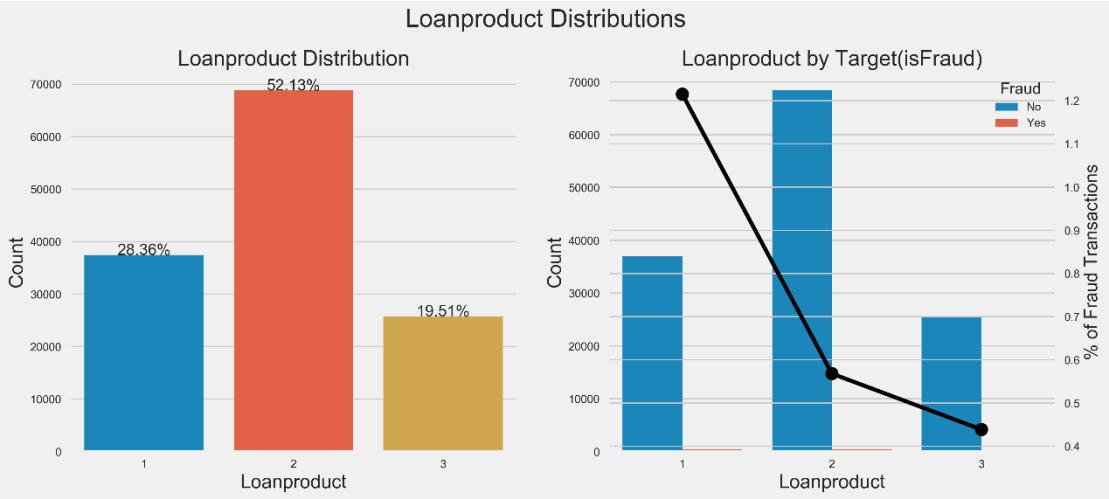
分类数据直白来说就是取值为有限的，或者说是固定数量的可能值，例如：gender、ethnic 等。这些特征内部往往不存在显著的数量大小关系，为了让数据适应后续建模的算法和库，必须将数据进行编码。这样的变化，让算法能够彻底领悟，原来特征取值是没有可计算性质的，是“有你就没有我”的不等概念。因此需要使用独热编码，将特征都转换为哑变量

2.1.3.1 直接编码

通过正负标签下不同特征属性可视化的表现，针对如下的特征属性：

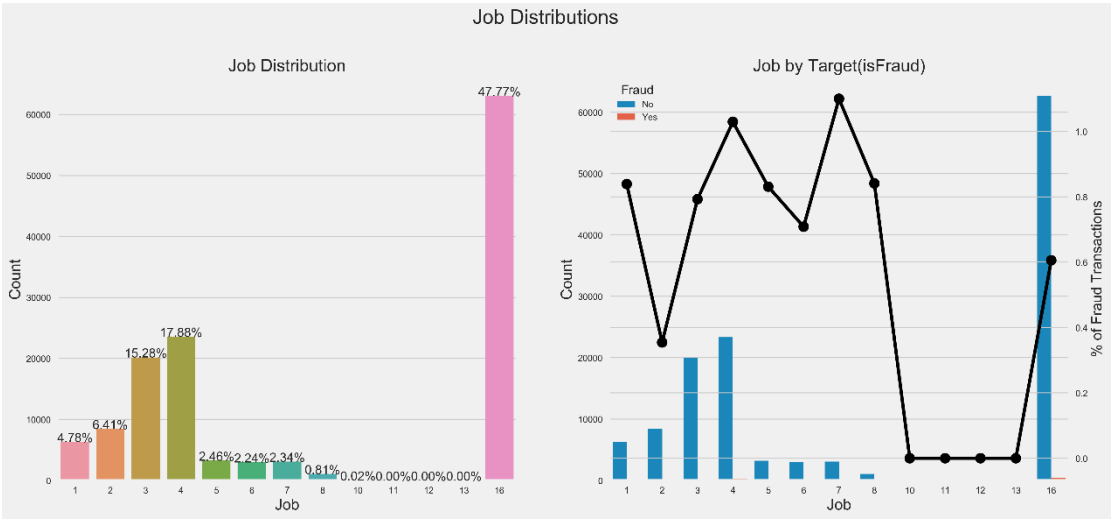
loanProduct,job,linkRela,ncloseCreditCard,unpayIndvLoan,unpayOtherLoan,unpayNormalLoan, 5yearBadloan，gender 等采用直接独热编码。

（1）loanProduct：表示贷款产品的类型，共有三个取值 1,2,3；



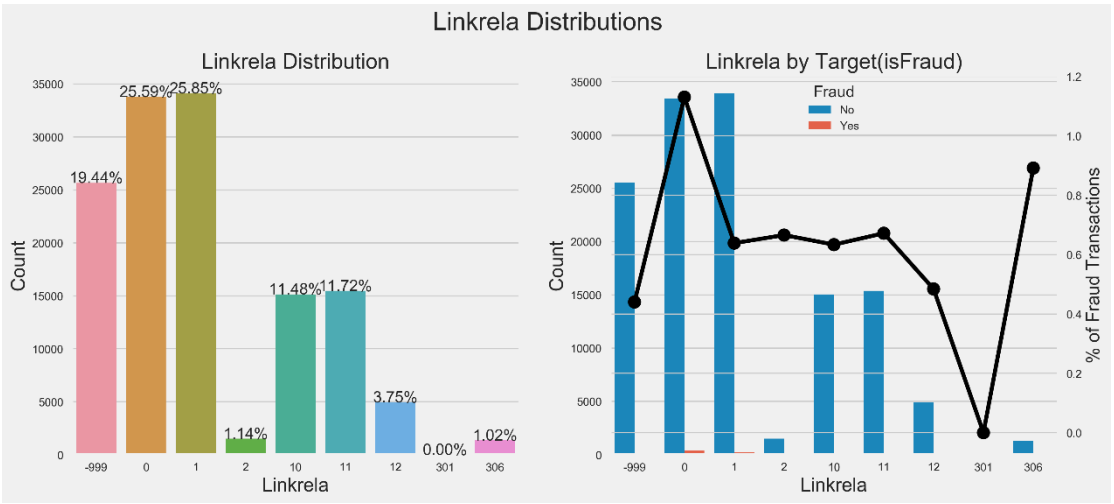
通过图像可以看出，不同产品在用户中贷款的比例存在差异，其中 2 号产品在用户中占比最大，高达 52.13%；同时通过贷款违约折线图发现，1 号产品的违约率最高，这就提示银行在向客户推荐和签订 1 号产品需要采取更加谨慎的态度和条款规定，尽可能降低用户的违约率；

（2）job：用户工作单位，共有 1,2，…，15 个取值；



由图像可以看出，用户是按照不同比例分布在各个工作单位，其中 job-15 占比最多，达 47.77%；其次是 job-3 和 job-4，分别为 15.25%和 17.88%；而贷款违约率最高的是 job-7，其次是 job-4 和 job-1，结合折线图和条形图可以认为，job-4 的用户是贷款违约的重要人群，值得银行在后续的工作中重点关注的工作人群，以此尽可能降低用户的违约率；

（3）linkRela：用户联系人关系，共有 9 个不同取值；

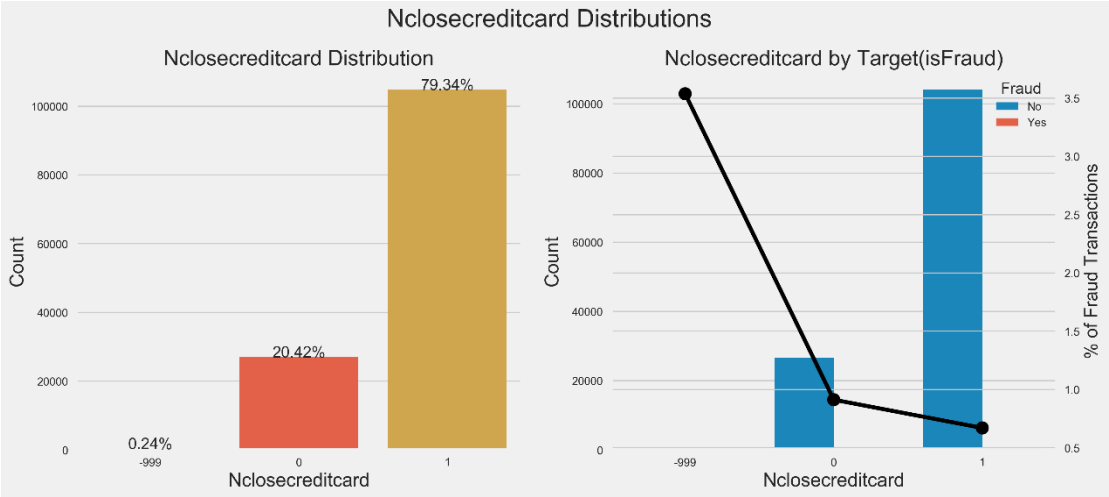


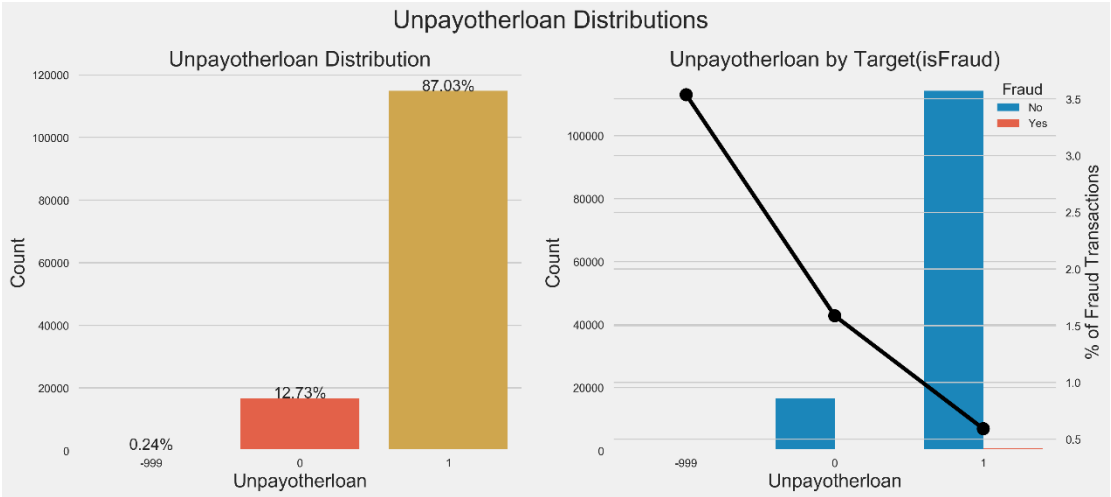
由图像坐标我们可以看出特征 linkRela 取值跨度很大，且其中包含-999 这样的数值，但是又因为真实的各个取值之间无明显的数量大小关系，此处依旧选择直接独热编码的方式处理。

其中 linkRela-（-999）、linkRela-0 和 linkRela-1 所占比例整体较高，同时有这三种联系人关系的用户的贷款违约率也相对较高，所以银行以后要考虑比如采用合适的条款要求用户填写像 linkRela-301 或者 linkRela-306 这样的联系人关系，

起到更好的监督关系和还款保证，这样有利于降低用户的违约率；

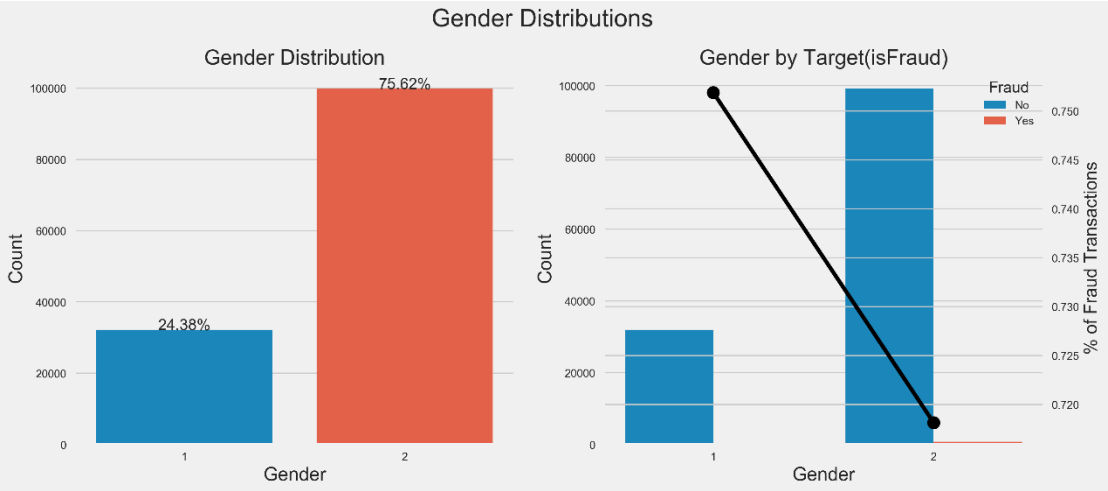
（4）ncloseCreditCard、unpayOtherLoan：此类属性涉及用户隐私，未作解释；





由图像可以看出 ncloseCreditCard-1 和 unpayOtherLoan -1 所占比例为多,但是其违约率最低，ncloseCreditCard-（-999）和 unpayOtherLoan -（-999）所占比例最少，但是违约率最高，因此相关人群是银行后续降低违约率需要关注的；

（5）gender：用户性别，共 2 个取值；



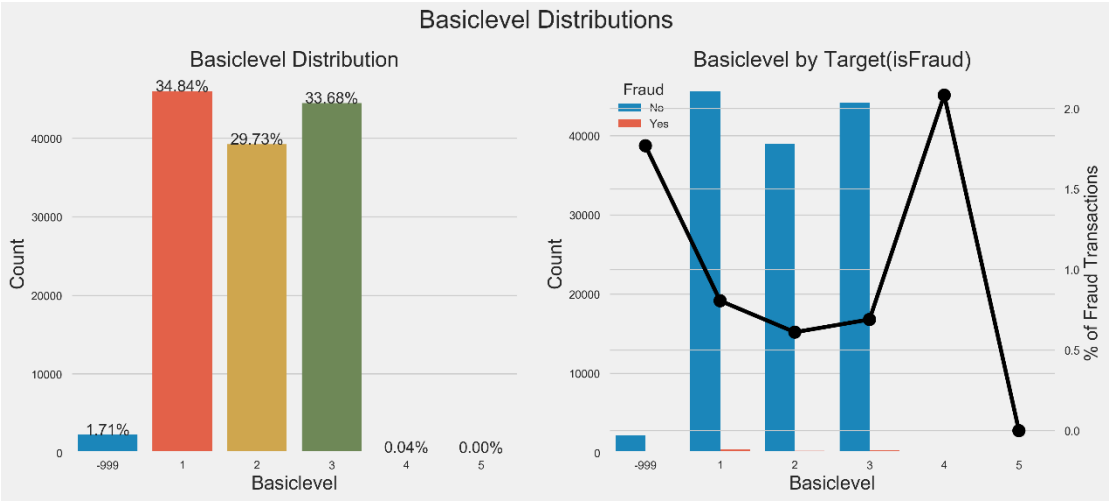
由图像可以清楚地看到，gender-1 用户数量占比低于 gender-2，但是两者违约率恰好相反，这说明 gender-1 类的用户有更强的违约趋势。同时两个属性区分则不明显，即重叠比较严重，这里采取保守策略，即保留此类区分度不高的特征。

由上述可视化展示的结果可以知道，列举的 10 种特征属性不同值对应的贷款违约率有较为明显的区别，同时属性内部之间不存在数量大小关系，只存在有你没我的关系，独热编码很明确的传达这样的关系。因此对它们采用独热编码是必要的的。

2.1.3.2 转换编码

对于 1.1.2 中未解决的-999 填补缺失值情形，此处作如下处理：

（1）basicLevel：基础评级，共有 6 个取值，作图结果如下：

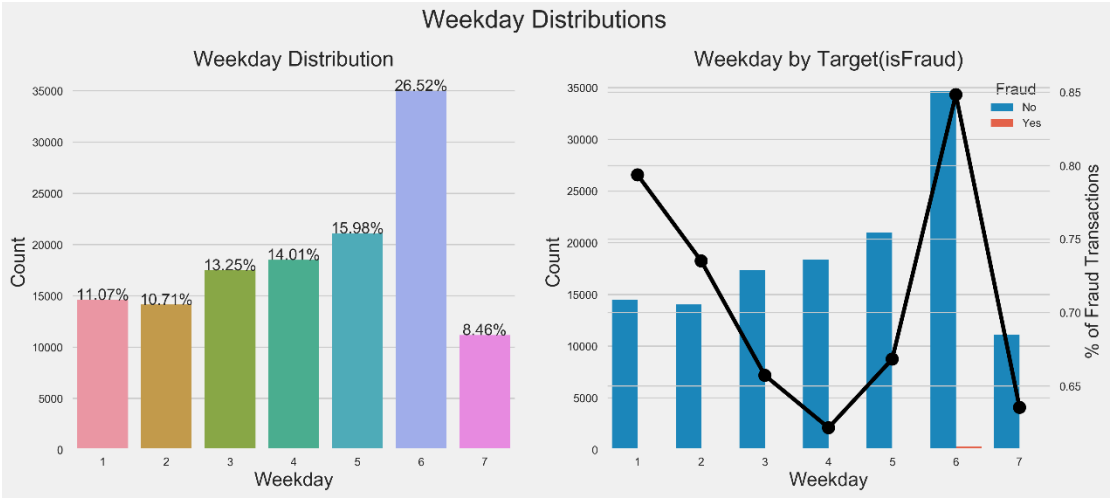


由图可知，basicLevel-（-999）和 basicLevel-4 数量占比都很少，但是拥有此

类特征的用户违约率占比却非常高，因此它们编码为 1，其他编码为 0，用以尽可能降低维度，减少特征矩阵的稀疏性；

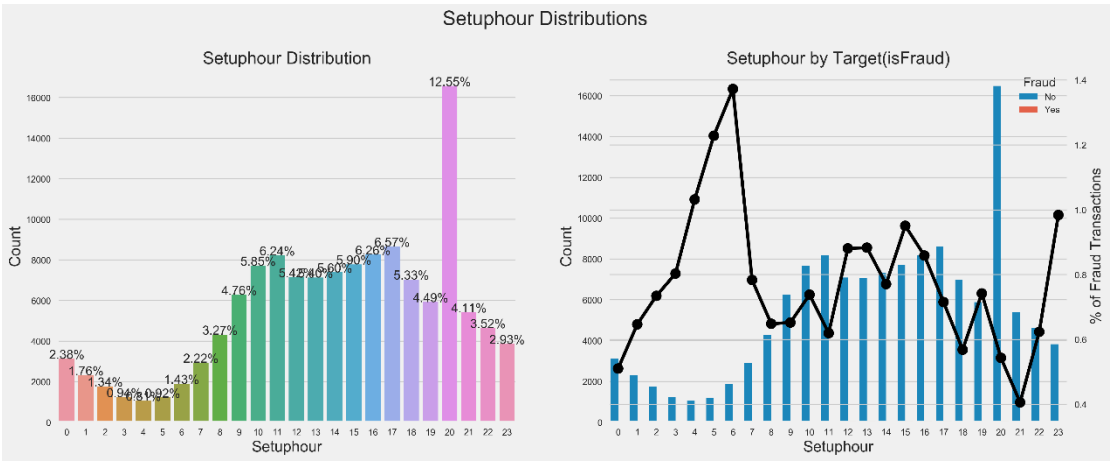
（2）时间分段编码

a. weekday：申请日（周几），共有七个属性；



时间分段编码，周一周二交易量较少，但违约率较高，因此它们编码为 1，其他编码为 0，用以尽可能降低维度，减少特征矩阵的稀疏性；

b. setupHour:申请时段，共有 24 个取值；



由图可大致认为折线图中点高于柱型图表示违约率较高，如 5 点；反之越低，如 20 点。据此，将早晨 9 点到晚上 9 点视作“平稳期”，晚上 9 点到早晨 9 点视作“危险期”。因此将时间分段编码，晚九点到第二天早九点交易量较少，违约率较高，因此它们编码为 1，其他编码为 0，用以尽可能降低维度，减少特征矩阵的稀疏性；

至此数据预处理过程全部完成，主要工作总结和结论如下：

（1）删除无关缺失值；

（2）采用独热编码，消除分类型特征内部的不存在的数量大小关系，更好地表达了传入模型的属性数据的意义；

（3）转换编码：聚合相似表现的属性取值，同时起到降维的作用。

（4）可视化不同特征属性对于贷款违约与否的不同影响，可以直观地提示银行相关工作人员重点关注例如采用了 1 号产品的用户等等；

2.2 特征工程

特征工程是将原始数据转换为更能代表预测模型的潜在问题的特征的过程，可以通过挑选最相关的特征，提取特征以及创造特征来实现。在论文采用的数据集中可能面对的问题有：特征之间有相关性，特征和标签无关，特征太多或太小，或者干脆就无法表现出应有的数据现象或无法展示数据的真实面貌。特征工程的目的：

1) 降低计算成本，
2) 提升模型上限

2.2.1 数值型特征处理

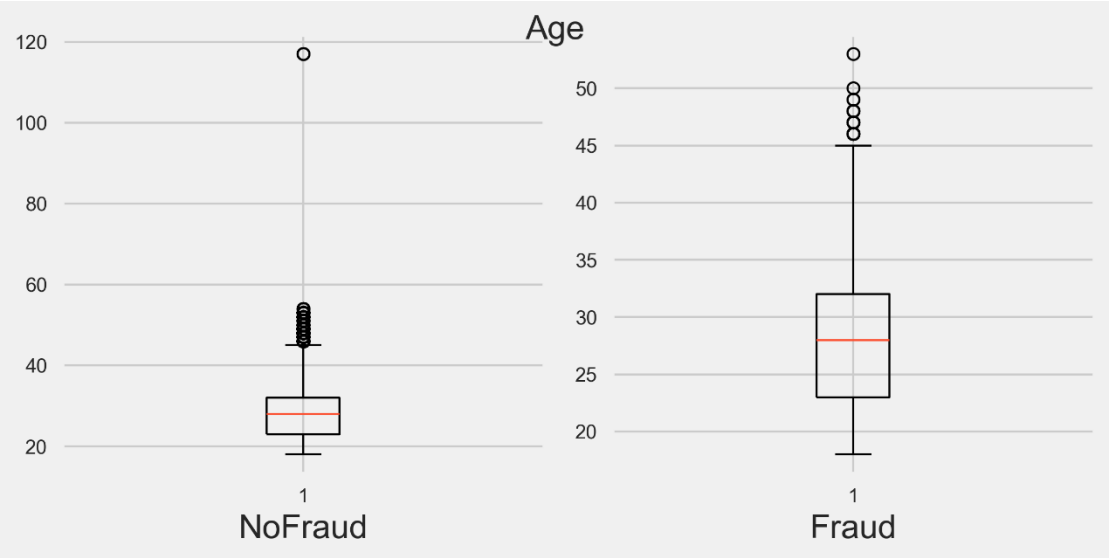
2.2.1.1 无关属性处理

数据集中共包含 104 个特征属性，但并非所有的属性都对我们的目标标签有影响，例如：id,certId, dist,residentAddr, isNew 就是对目标无关的 5 个属性，采取删除处理。

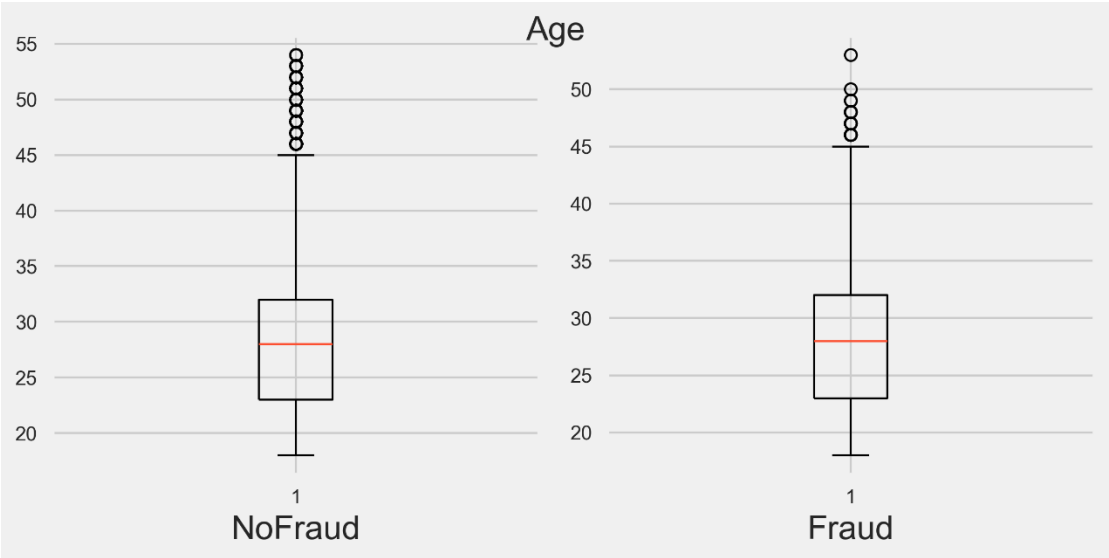
2.2.1.2 异常值处理

在数据预处理过程中，未真正考虑过数据中异常值的存在，放在特征工程中进行处理：

（1）age:用户年龄



由箱线图显示存在年龄大于 100 岁的人偏离五数之外且数量极少，对于模型拟合没有正向作用，作为异常值去除。去除之后的图像如下：

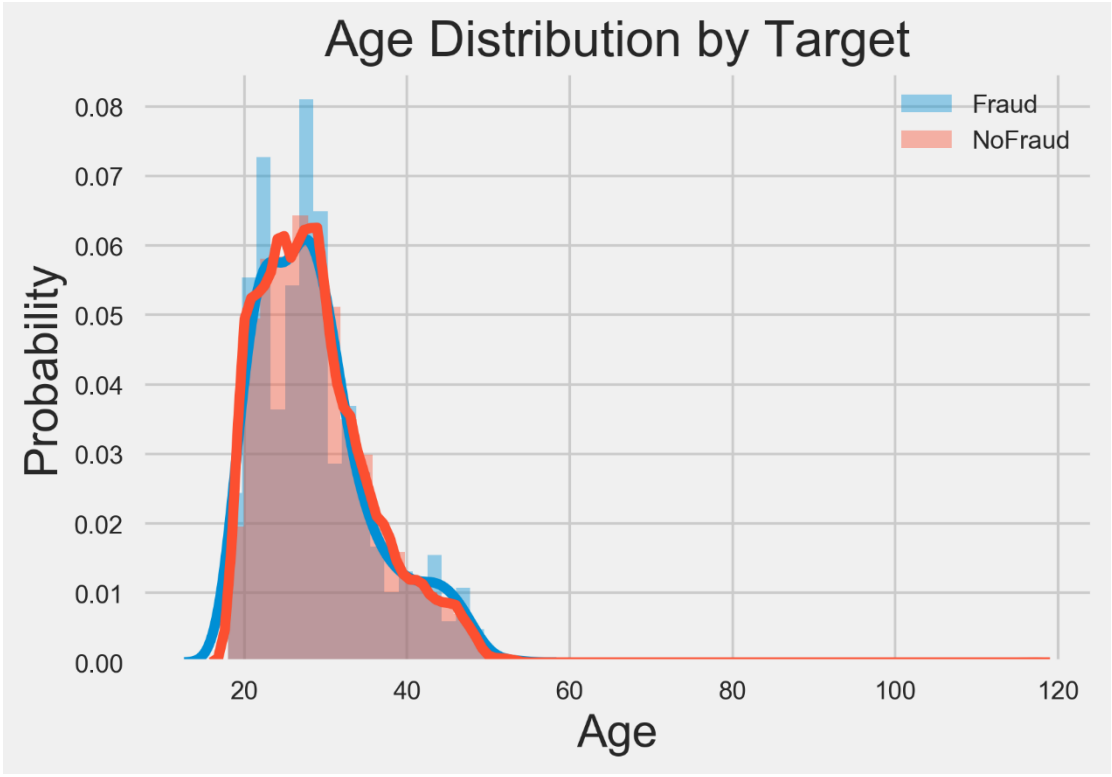


由图像可以看出，箱线图 age 属于五数之外的数量少了许多，说明将年龄大于 100 的用户删掉是合理的。

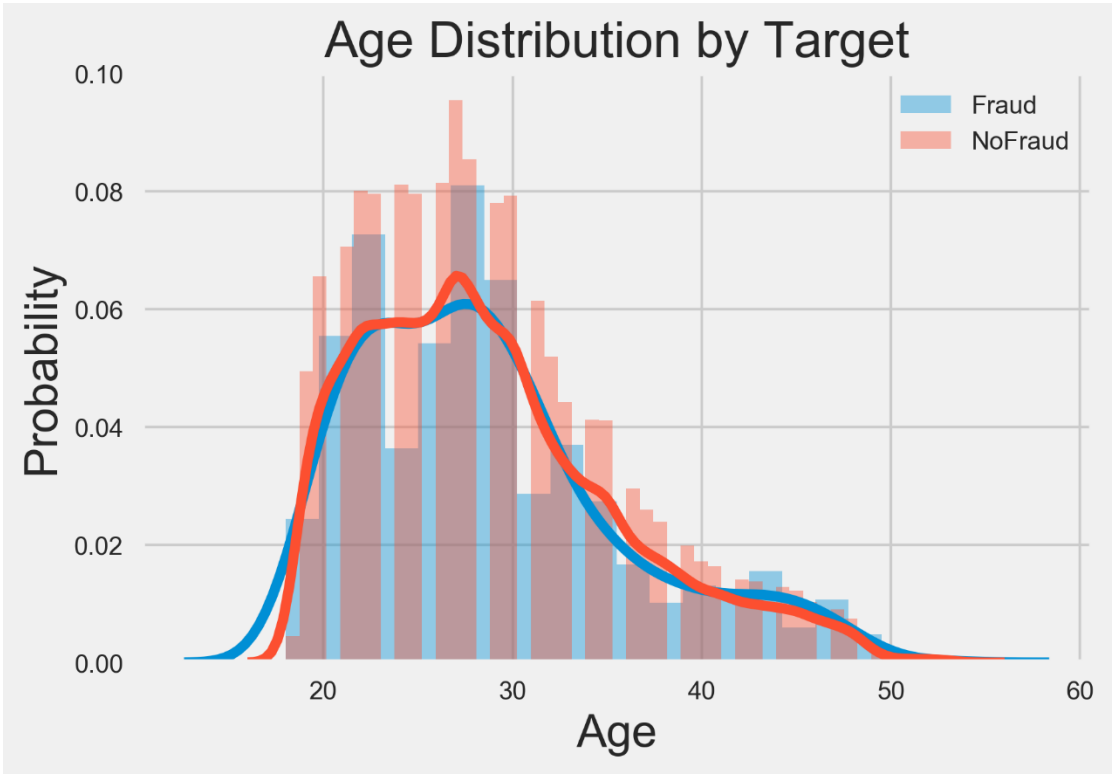
2.2.1.3 标准化处理

下面考虑对 age, highestEdu 等进行标准化处理：（以 age 为例说明）

删除异常值之前标准化图像：



删除异常值之后标准化图像：



由图像可以看出，异常值处理之后的标准化 age 对应的直方图呈现的右偏趋势较之前有明显的减弱，同时在直方图上拟合出违约和不违约用户的密度曲线，对称性有点增强；说明将年龄大于 100 的用户删掉是合理的。

最后，对变量 x_0 到 x_78，我们没有任何已知的信息，保持不变。

至此，数据预处理过程提升了数据的质量和表达，特征工程深入挖掘了数据内部的信息，两者结合起来对数据进行操作，可以让所得数据适应模型，匹配模型的需求，更好地服务于算法模型。我们的数据集为：

（1）训练集 train_x.csv 有 132008 行用户记录，其中有 128 个特征属性；

（2）训练集 train_target.csv 有 132008 行用户记录，其中有 1 个目标标签；

（3）测试集 test_x.csv 有 23561 行用户记录，其中有 128 个特征属性；

将用于下面不同模型的训练和测试。

3 模型建立与测试

3.1 模型数据集划分

在数据探索性分析中，提到过运用于模型的数据呈现出极端不平衡的表现，即计算得到正负标签之比：

(Y=0): (Y =1)= 136.67361835245046:1

因此论文中划分的测试集和训练集的时候会以预测标签 y 为策略进行分层抽样，否则会仅有的极少数的正样本大部分出现在训练集或者测试集，其中最后得到的划分比例为：

训练集数据量 ： 测试集数据量 = 8 ： 2

3.2 模型效率评估指标：AUC

论文采用 2019 厦门国际银行“数创金融杯”数据建模大赛提供的数据集，比赛要求提交的模型评价指标为：AUC。AUC 最直观的理解为：随机挑选一个正样本以及一个负样本，算法根据计算得到的 score 值将这个正样本排在负样本前面的概率，即是 AUC 值。score 值就是预测为正的概率的值，排在前面表示的是正样本的预测为正的概率值大于负样本的预测为正的概率值。本文中采用了直观地绘制 ROC 曲线，计算 ROC 曲线下面的面积估计 AUC 的值，由此展示论文中模型在训练和测试上的表现好坏。

ROC 曲线：接收者操作特征(receiveroperating characteristic),roc 曲线上每个

点反映着对同一信号刺激的感受性。横轴：假正类率(false postive rate FPR)特异度，划分实例中所有负例占所有负例的比例；(1-Specificity)；纵轴：真正类率(true postive rate TPR)灵敏度，Sensitivity(正类覆盖率)。

针对一个二分类问题，将实例分成正类(postive)或者负类(negative)。但是实际中分类时，会出现四种情况.

(1)若一个实例是正类并且被预测为正类，即为真正类(True Postive TP)

(2)若一个实例是正类，但是被预测成为负类，即为假负类(False Negative FN)

(3)若一个实例是负类，但是被预测成为正类，即为假正类(False Postive FP)

(4)若一个实例是负类，但是被预测成为负类，即为真负类(True Negative TN)

由此可以得到一个二分类问题的混淆矩阵：

样本		预测值	
		1	0
实际值	1	TP（真正）	FN（假负）
	0	FP（假正）	TN（真负）

可以知道二分类 ROC 曲线横纵轴值，计算方式为：

横轴：（FPR）= FP/(FP+TN)

纵轴：（TPR）= TP/(TP+FN)

我们使用了 jupyter notebook 中的 plotROC 绘制 ROC 曲线，roc_auc_score 估计 AUC 值，同时计算出混淆矩阵，下面开始对数据进行建模。

3.3 Xgboost 模型原理与应用

3.3.1 Xgboost 模型简介和原理

XGBoost 全称是 Extreme Gradient Boosting，译为极限梯度提升算法。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。它能够比其他使用梯度提升的集成算法更加快速，并且已经被认为是在分类和回归上都拥有超高性能的先进评估器。

XGBoost 算法一般采用正则化+泰勒展开学习得到目标函数最优化，目标函数是自身的损失函数和正则化惩罚项相加而成的，由此算法能够防止过拟合，高维稀疏等影响模型效果的消极因素；

其中，目标函数一般形式为：

$$\text{Object}=\sum_{i=1}^{n} l(y_i,\hat{y}_i)+\sum_k \Omega(f_k), f_k \in F$$

第一项为损失函数，第二项为正则惩罚项；

求解方法采用加性训练，大致如下：

$$\hat{y}_i^{(t)} = \hat{y}_i^{(t-1)} + f_t(x_i)$$

$$\text{Object} = \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)) + \sum_k \Omega(f_k) + constant, f_k \in F$$

再结合泰勒展开，最后我们的目标函数可以转化为：

$$\text{Object} = \sum_{i=1}^{n} [g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)] + \Omega(f_k), f_k \in F$$

其中：$g_i = \partial_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)}), h_i = \partial^2_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)})$

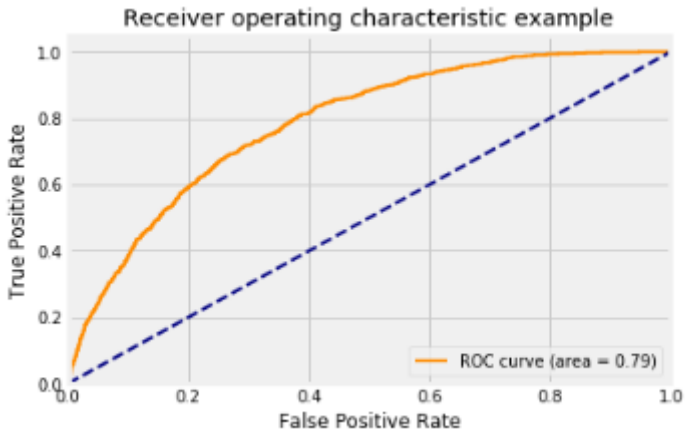
XGBoost 背后也是 CART 树，这意味着 XGBoost 中所有的树都是二叉的。其思想是不断地添加树，不断地进行特征分裂来生长一棵树，每次添加一个树，其实是学习一个新函数，去拟合上次预测的残差。当我们训练完成得到 k 棵树，我们要预测一个样本的分数，其实就是根据这个样本的特征，在每棵树中会落到对应的一个叶子节点，每个叶子节点就对应一个分数，最后只需要将每棵树对应的分数加起来就是该样本的预测值。

下面开始利用 XGBoost 对数据进行建模。

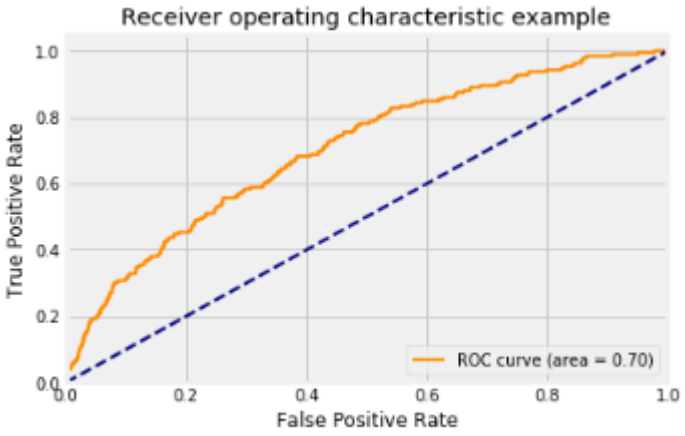
3.3.2 Xgboost 模型应用

向 jupyter notebook 中导入 xgboost 库，我们在初步试探模型参数后，考虑采用 5 折交叉验证，调参，选取最优参数，对全部的训练集进行训练（所有算法调参方法大体一致），结果如下：（此处通过 ROC 曲线展示）

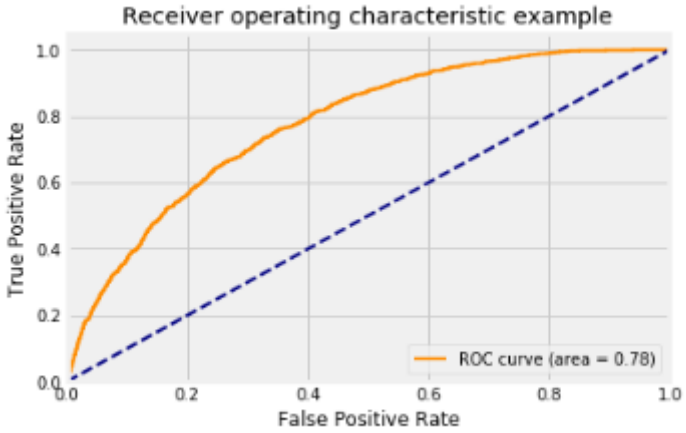
训练集上得到 ROC 曲线：



测试集上得到 ROC 曲线：



全数据集上得到 ROC 曲线：



由 ROC 曲线可以得出，测试集上模型的表现低于训练集上模型的表现，同时此时模型在测试集上的 **AUC= 0.7760489431907484**，表现效果较好。

3.4 EasyEnsemble 算法原理与应用

3.4.1 EasyEnsemble 算法简介和原理

算法步骤：

Algorithm 1 The `EasyEnsemble` algorithm.

1: {Input: A set of minority class examples $\mathcal{P}$, a set of majority class examples $\mathcal{N}$, $|\mathcal{P}| < |\mathcal{N}|$, the number of subsets T to sample from $\mathcal{N}$, and s_i, the number of iterations to train an AdaBoost ensemble H_i}

2: $i \Leftarrow 0$

3: **repeat**

4: $i \Leftarrow i + 1$

5: Randomly sample a subset $\mathcal{N}_i$ from $\mathcal{N}$, $|\mathcal{N}_i| = |\mathcal{P}|$.

6: Learn H_i using $\mathcal{P}$ and $\mathcal{N}_i$. H_i is an AdaBoost ensemble with s_i weak classifiers $h_{i,j}$ and corresponding weights $\alpha_{i,j}$. The ensemble's threshold is θ_i, i.e.,

$$H_i(x) = \text{sgn}\left(\sum_{j=1}^{s_i} \alpha_{i,j} h_{i,j}(x) - \theta_i\right).$$

7: **until** $i = T$

8: Output: An ensemble

$$H(x) = \text{sgn}\left(\sum_{i=1}^{T}\sum_{j=1}^{s_i} \alpha_{i,j} h_{i,j}(x) - \sum_{i=1}^{T} \theta_i\right).$$

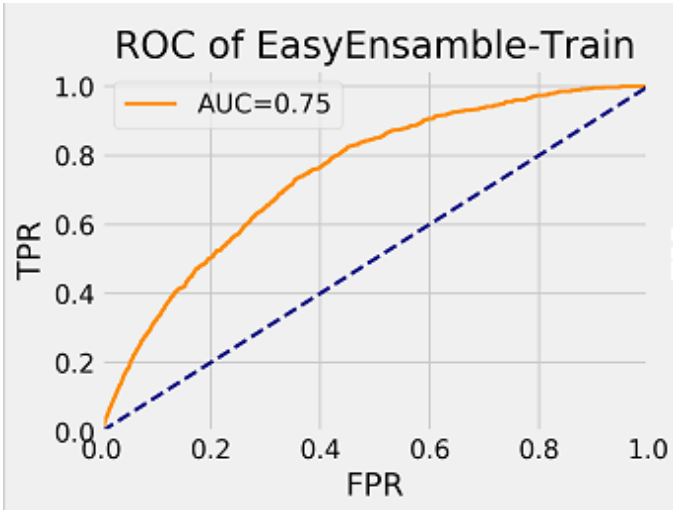
easy ensemble 算法每次从多数类中抽样出和少数类数目差不多的样本，然后和少数类样本组合作为训练集。在这个训练集上学习一个 adaboost 分类器。

最后预测的时候，是使用之前学习到的所有 adaboost 中的弱分类器（就是每颗决策树）的预测结果向量（每个树给的结果组成一个向量）和对应的权重向量做内积，然后减去阈值，根据差的符号确定样本的类别。

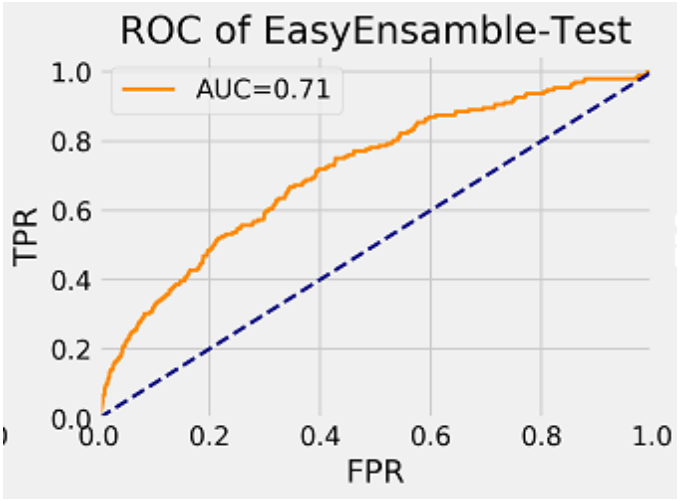
3.4.2 EasyEnsemble 算法应用

向 jupyter notebook 中导入 EasyEnsemble 库，在经验下调节参数情况下的运行结果为：（此处通过 ROC 曲线展示）

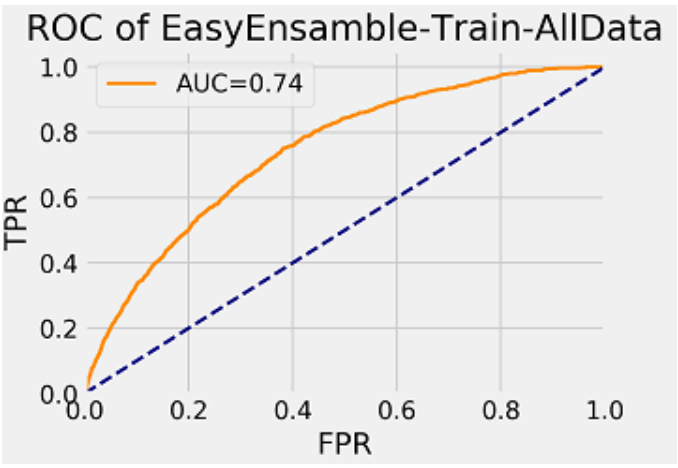
训练集上得到 ROC 曲线：



测试集上得到 ROC 曲线：



全数据集上得到 ROC 曲线：



3.5 Balance Cascade 算法原理与应用

3.5.1 Balance Cascade 算法简介和原理

算法步骤：

T to sample from $\mathcal{N}$, and s_i, the number of iterations to train an AdaBoost ensemble H_i}

2: $i \Leftarrow 0$, $f \Leftarrow \sqrt[T-1]{|\mathcal{P}|/|\mathcal{N}|}$, f is the false positive rate (the error rate of misclassifying a majority class example to the minority class) that H_i should achieve.

3: **repeat**

4: $i \Leftarrow i + 1$

5: Randomly sample a subset $\mathcal{N}_i$ from $\mathcal{N}$, $|\mathcal{N}_i| = |\mathcal{P}|$.

6: Learn H_i using $\mathcal{P}$ and $\mathcal{N}_i$. H_i is an AdaBoost ensemble with s_i weak classifiers $h_{i,j}$ and corresponding weights $\alpha_{i,j}$. The ensemble's threshold is θ_i i.e.,

$$H_i(x) = \text{sgn}\left(\sum_{j=1}^{s_i} \alpha_{i,j} h_{i,j}(x) - \theta_i\right),$$

7: Adjust θ_i such that H_i's false positive rate is f.

8: Remove from $\mathcal{N}$ all examples that are correctly classified by H_i.

9: **until** $i = T$

10: Output: A single ensemble

$$H(x) = \text{sgn}\left(\sum_{i=1}^{T}\sum_{j=1}^{s_i} \alpha_{i,j} h_{i,j}(x) - \sum_{i=1}^{T} \theta_i\right).$$

可以看出 balance cascade 算法和 easy ensemble 还是很相似的，差别就在于第 7 步和第 8 步。

第 6 步中由算法训练出一个分类器，在第 7 步调整分类器H_i的阈值θ_i以保证分类器H_i的正负率w为 f。而 f 的定义为:

$$f=\frac{|P|}{|N|}^{\frac{1}{(T-1)}}$$

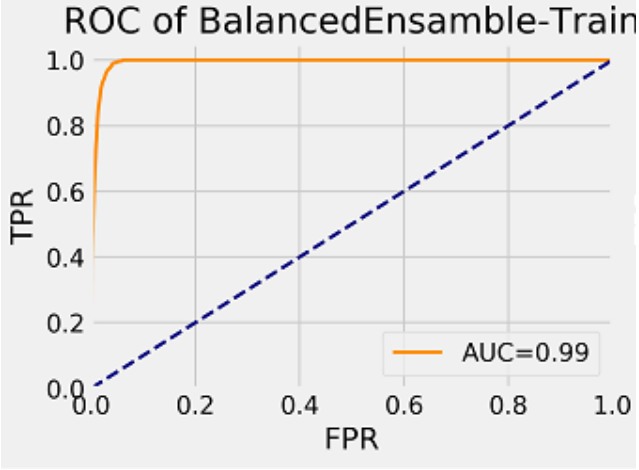
这样的话，根据H_i的阈值θ_i对所有的多数类样本进行预测，如果样本的估计值大于该阈值则判定为正例（即少数类），这些都是预测错误的多数类样本。如果样本的估计值小于阈值则判定为负例（即多数类），这些就是预测正确的多数类样本，第 8 步中去掉的也就是这一部分样本，而这些样本占当前所

有多数类的样本的比例是 $1-f$，即每次迭代多数类样本保留下来的比例为 f。那么 T-1 次后，多数类样本还剩 $|N| \times f^{T-1} = |P|$ 个，那么在第 T 次迭代中，多数类样本就会少于少数类样本，此时可以停止迭代。

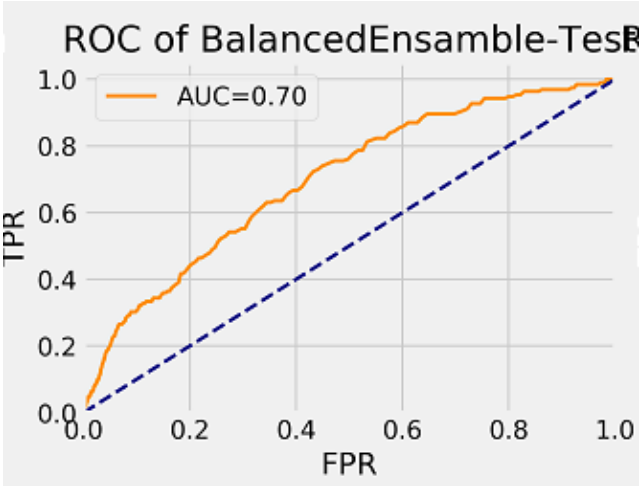
3.5.2 Balance Cascade 算法应用

向 jupyter notebook 中导入 EasyEnsemble 库，在经验下调节参数情况下的运行结果为：（此处通过 ROC 曲线展示）

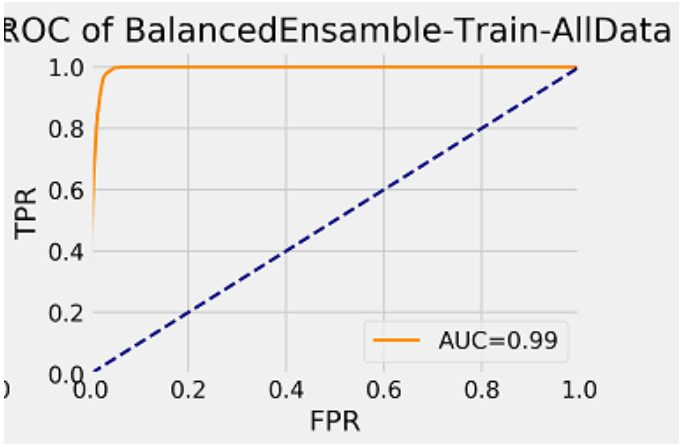
训练集上得到 ROC 曲线：



测试集上得到 ROC 曲线：



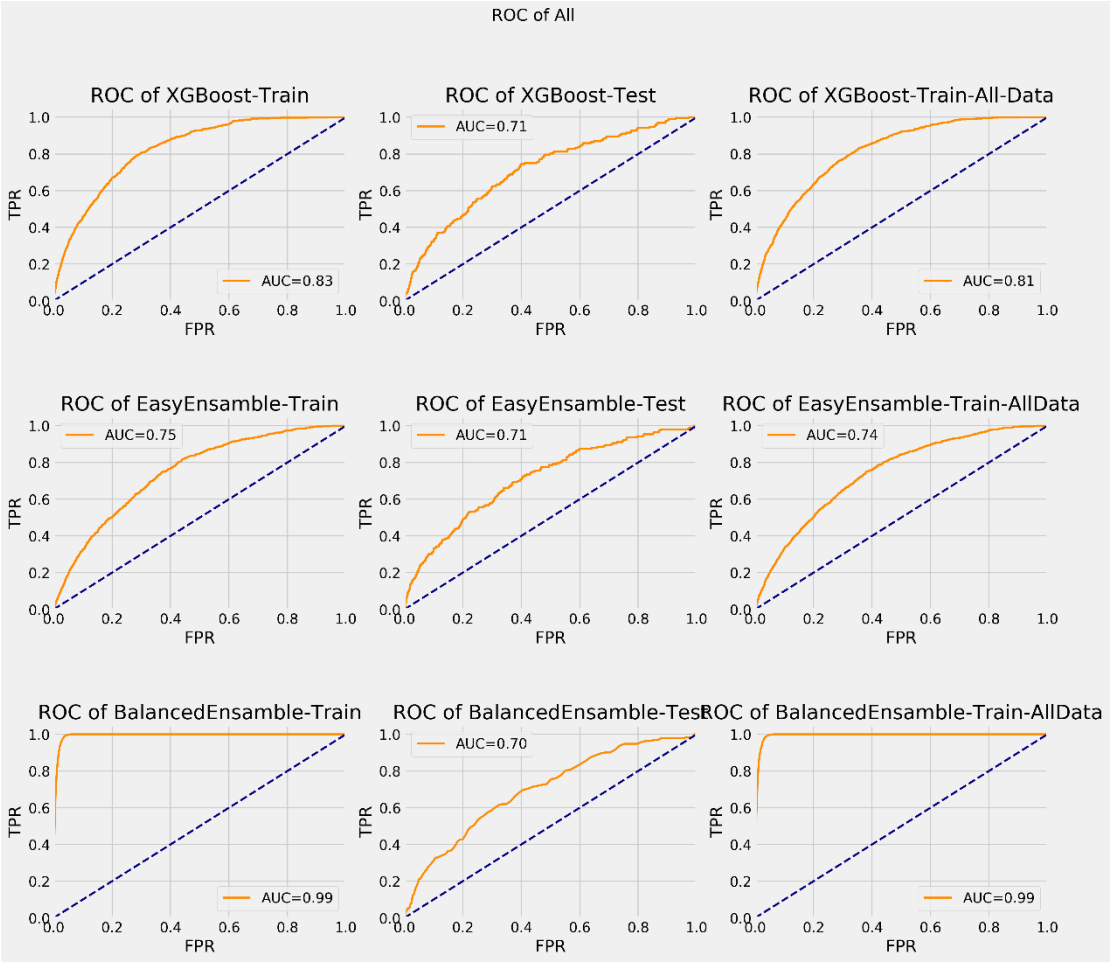
全数据集上得到 ROC 曲线：



3.6 模型对比与总结

上面针对高维稀疏和不平衡数据问题，我们采用了 Xgboost 算法，easy ensemble 算法以及 Balance Cascade 算法进行建模和测试，同时我们也测试了不同基分类器个数下的训练测试效果，展示结果如下：

（1）150-Estimator-ROC：



（2）150-Estimator-Time ：

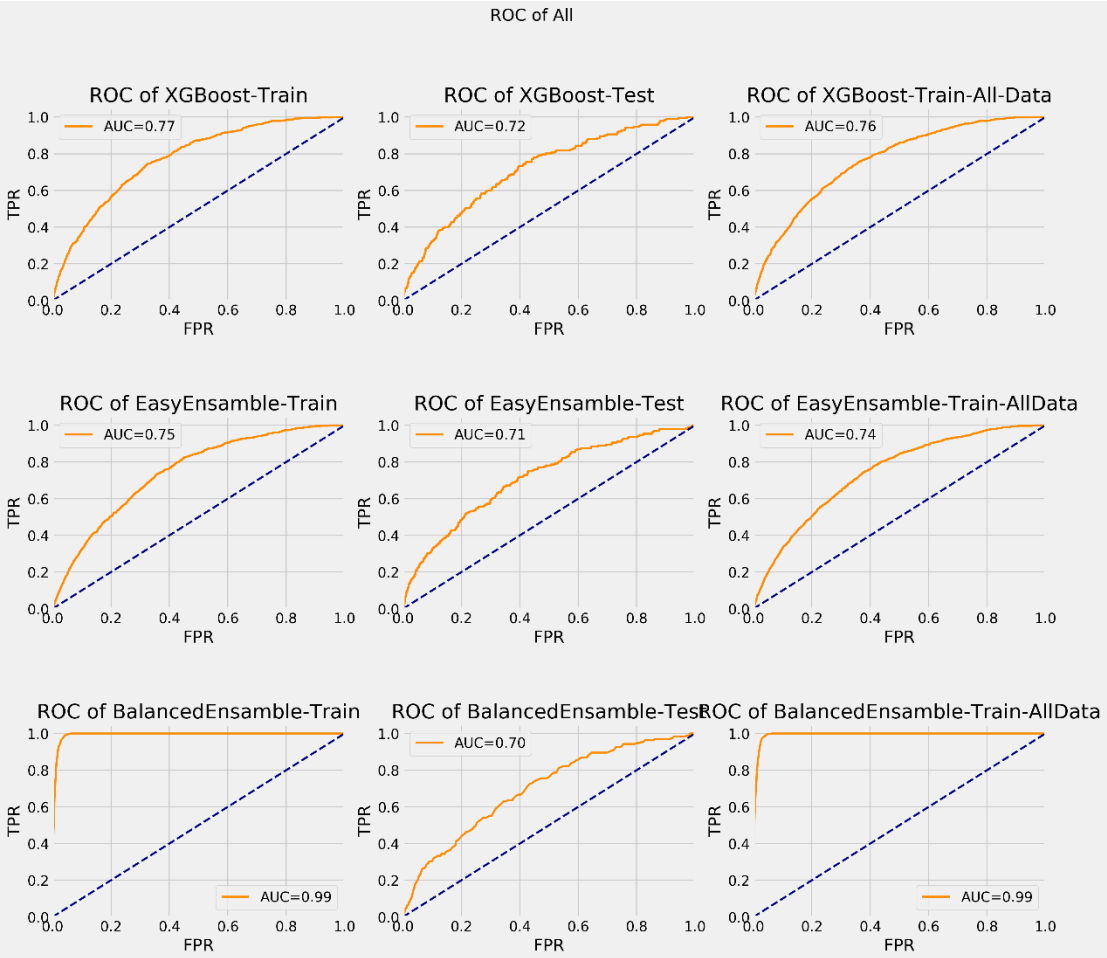
Time(seconds)	Search	Train	Prediction
XGBoost	306.152	20.553	0.16
EasyEnsamble	1633.848	62.603	29.182
BalancedEnsamble	1166.192	81.818	1.145

考虑增加基分类器的数量来提高算法的效果，最后经过多次尝试，我们将基分类器的数量增加为 300，运行结果如下：

（3）300-Estimator-Time：

Time(seconds)	Search	Train	Prediction
XGBoost	1512.681	22.744	0.158
EasyEnsamble	3635.758	142.838	64.232
BalancedEnsamble	2764.941	194.82	2.833

（4）300-Estimator-ROC：



从 ROC 曲线表现：

一、XGBoost，在 150 个基分类器的时候表现出较强的过拟合现象，主要是由于其 Boosting 的机制存在 target leakage (Liudmila, NIPS 2018)。在增加基分类器的数量后，由于 XGBoost 的行列采样机制引入更多随机性，降低了过拟合。此外，由于代价敏感的引入，使得其对于不平衡数据集也有良好的表现。

二、 EasyEnsamble 采用无监督的降采样，很好地平衡了数据集， 且由于其对负样本的采样是随机的，所以基本不存在过拟合 问题。

三、BalancedCascade 由于其采用有监督的降采样，容易引起过 拟合。因为中途被剔除的数据可能也是有用的（F.-Z. Marcos, 2004）。这里在极度不平衡数据集的表现并非如论文 （Xu-Ying Liu, 2008）所说那么好。

从时间成本：

一、XGBoost 运行效率的提升来自两方面，分别是算法理论和工 程实现。算法理论上主要是 Column Subsampling， Histogram-based Algorithm(特征分割点-> 分割区间) 以及 Sparsity-aware Split Finding，减少了计算量。更主要的是工程上的实现带来了效率提升，底层由 C 和 C++ 实现， Cache-aware block structure + Out-of-core tree learning 且支 持并行 (如寻找最优分割时)。

二、 EasyEnsamble 较慢的原因主要是工程实现。因为所用库 imblearn 由 Python 实现，虽然对并行支持很好，也是无法 弥补与 C++ 效率上的差距。

三、 BalancedCascade，也是用库 imblearn，所以总体也是偏慢。 但是由于其剔除样本的性质，所以其对特大数据集的效率会 比 EasyEnsamble 高。

最后结合模型效率以及时间成本，本文最后推荐采用 Xgboost 算法训练得到的模型用于实际预测。

4 结论

从金融市场的发展现状来看，更好的预测信用风险是一个极其重要的问题，关乎金融系统运行的稳定。论文针对客户信用类型的预测，首先对于收集到的实际数据，采用了数据预处理和特征工程，进一步提升了训练模型的数据质量，对于提高模型效率的上限有一定帮助;然后分别采用了 Xgboost 算法、easy ensemble 算法和 Balance Cascade 算法对数据进行建模和训练，最后结合模型效率 AUC=0.72 以及时间成本，最后推荐采用 Xgboost 算法训练得到的模型用于实际预测。