2020春季学期

《人工智能引论》课程小组项目书面报告

利用机器学习方法构建
信用卡评分模型

SpiritedAwayCN　eesxy　学徒月亮

2020年5月

注：此文本移除了真实姓名、机构等涉及隐私的内容，非报告原文

摘 要

利用 kaggle 比赛 GiveMeSomeCredit 提供的数据，使用机器学习等算法训练两年内是否严重违约的二分类器，最终得到 kaggle 上评分的 AUC 值 0.86780 (private board)，对应排名 65/943。并基于此模型，利用统计学的分析方法确定了信用卡评分公式$score = 600 - 81.8 \ln \frac{p}{1-p}$。

目 录

1. 项目背景

人工智能的两大关键因素就是“算力”和“数据”。其中，如何利用数据，是一个至关重要的问题，它决定了你能否挖掘出足够多的信息，进而决定了所得的人工智能的最终质量。因此，在人工智能的学习过程中，很有必要训练同学们的数据处理能力。在“人工智能引论”的“数据智能”小班中，老师就通过几个课题来让同学们自由发挥，自己去寻找一种最佳的数据挖掘、模型训练方法，得到自己的第一个“人工智能”。

我们选的课题是“信用卡评分模型”。其大概内容就是根据一个用户的一些基本信息、财产状况和信用记录来判断，如果给用户贷款，它会违约的概率有多大。这样的一个模型在银行的财政系统中将会起到很大的作用。

2. 正文

我们的工作主要分为以下三个阶段：预处理、模型训练和评分器的生成。

预处理就是将我们的得到的原始数据进行“清洗”、“去异常”、“增加或修改特征”等等操作，从而使之能更好地被模型利用。在此之后就是至关重要的模型训练过程。我们尝试了多种模型，最终主要确定了两个方案：第一，xgb 加线性分类器；第二，一个简易的神经网络。这两个模型兼顾了效率和质量，综合看来的表现相对较好。而最后一个评分器生成过程就是将[0,1]概率空间较好地映射到[300,900]分数空间的过程。我们分析了数据中统计量 lnOR 的分布情况，并决定以此为依据设计评分函数。

下面我们就对这三部分依次介绍。

a) 数据预处理

i. 数据总览

除待预测的属性“是否有严重违约”(SeriousDlqin2yrs)外，对其余十个属性分别作箱线图（图 1）和直方图（图 2），以便对数据有一个整体直观的认识。

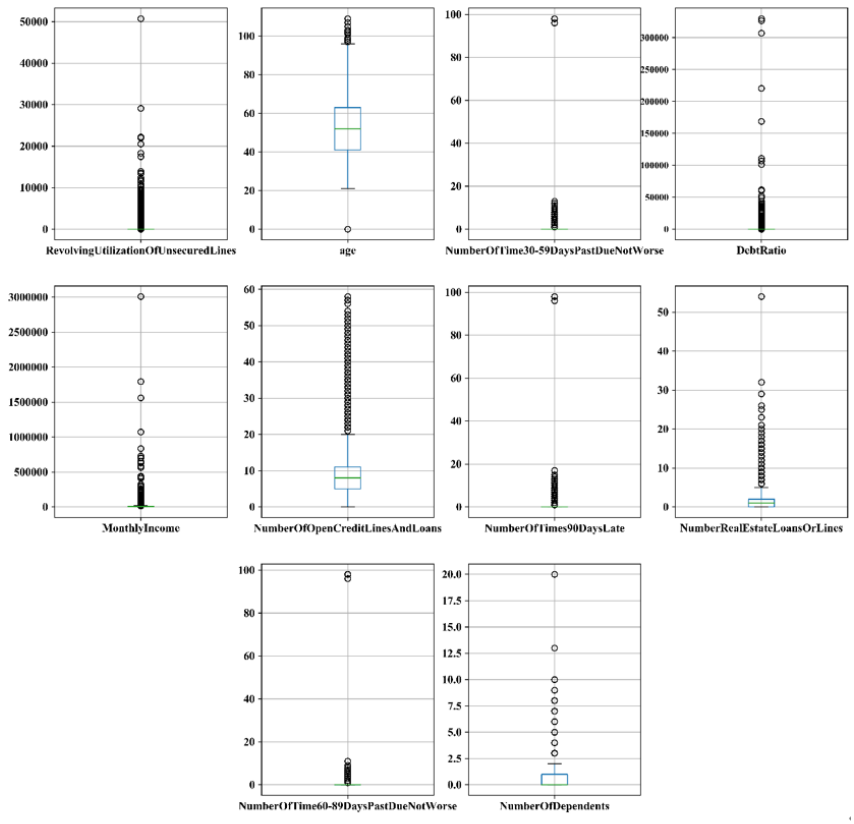

RevolvingUtilizationOfUnsecuredLines
age
NumberOfTime30-59DaysPastDueNotWorse
DebtRatio
MonthlyIncome
NumberOfOpenCreditLinesAndLoans
NumberOfTimes90DaysLate
NumberRealEstateLoansOrLines
NumberOfTime60-89DaysPastDueNotWorse
NumberOfDependents

图 1

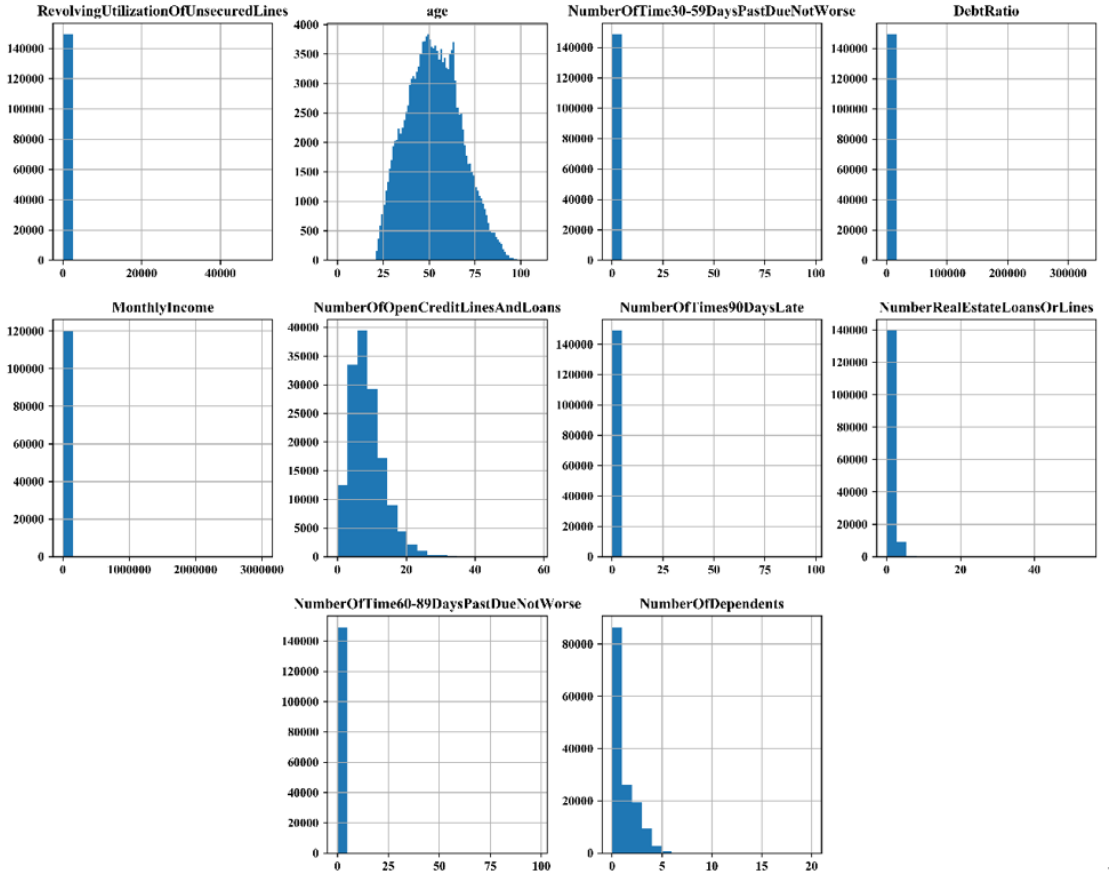

RevolvingUtilizationOfUnsecuredLines
age
NumberOfTime30-59DaysPastDueNotWorse
DebtRatio
MonthlyIncome
NumberOfOpenCreditLinesAndLoans
NumberOfTimes90DaysLate
NumberRealEstateLoansOrLines
NumberOfTime60-89DaysPastDueNotWorse
NumberOfDependents

图 2

ii. 异常值处理

从箱线图中可以直观看出，逾期 30-59 天笔数(NumberOfTime30-59DaysPastDueNotWorse)、逾期 60-89 天笔数(NumberOfTime60-89DaysPastDueNotWorse)、逾期 90 天以上笔数(NumberOfTimes90DaysLate)三个属性的异常值具有相同的特征，将三个属性画在同一张箱线图中如图 3 所示。

经检查，三个属性的异常值均为 98 和 96，且这三个属性的异常均发生在相同的行，因此直接将这些行删去。删去异常值后的箱线图如图 4，可以看出大部分数据取值为 0，即没有出现短期违约现象，只有少量数据出现了短期违约情况，这与失信者远少于未失信者这一分布规律相吻合。最终结果表明删去异常值后模型性能大幅提升。

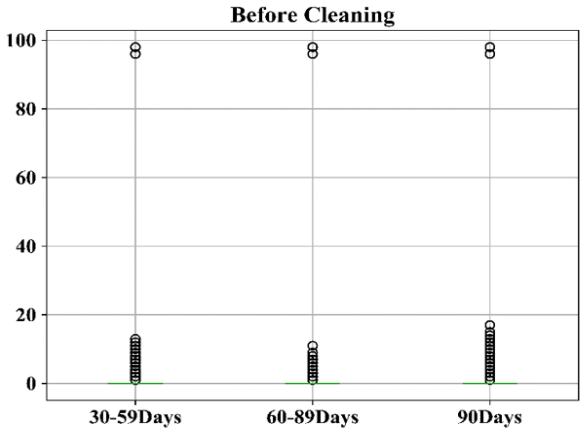


图 3

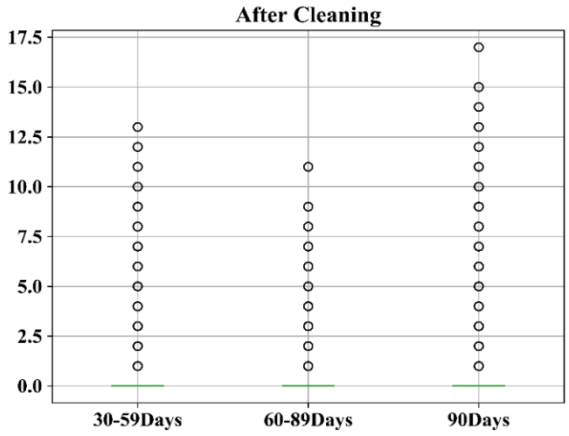


图 4

此外，从箱线图中还可以看出年龄(age)这一属性有明显的孤立点 0（图 5），显然贷款者的年龄不会为 0，因此认定这部分数据为异常值。将年龄为 0 的行删去后年龄最小项为 21，符合常识。删去异常值后的年龄分布直方图如图 6。

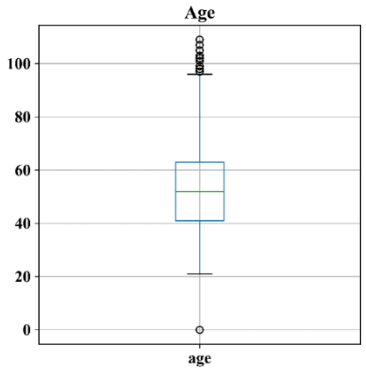


图 5

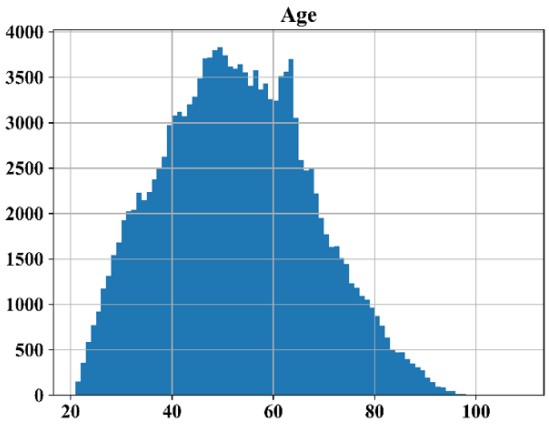


图 6

iii. 特征增添与修改

受到 Kaggle 论坛的启发，适当地修改、增添特征将有效地提升最终的 AUC 值。 因此，重新审视提供数据的特征。

1. 对数变换

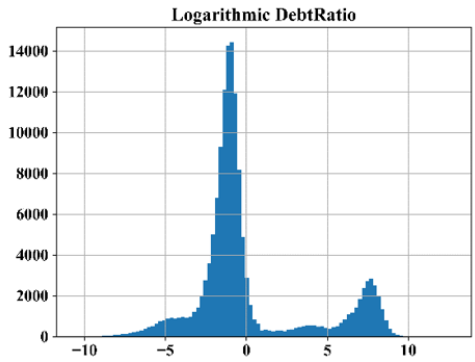
不难发现有这样的事实：月收入 1 与 10,000 差别巨大，而月收入 100,000 与 110,000 几乎无本质差别。因此对于此类特征，可以考虑作对数变换，用对数值替换原特征。表 1 是经可视化与本地验证集表现而最终确定的需要对数变换的特征列表。

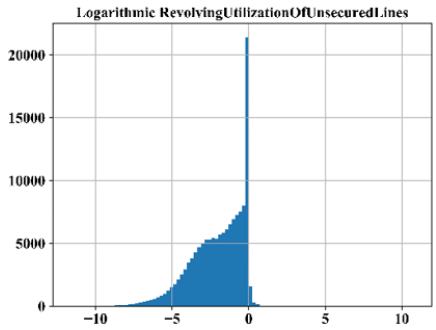
```
data[ 'Log' + fname ] = np.log( data[ fname ] )
```

图 7 显示了经对数变换后各特征的分布，观察发现取对数后，数据的分布更为自然。本地验证集的测试结果表明，这些维度作对数变换后，AUC 值均获得了 0.01~0.05 不等的提升。

MonthlyIncome	Real
RUOUL[1]	Percentage
DebtRatio	Percentage
Age	Integer

表 1





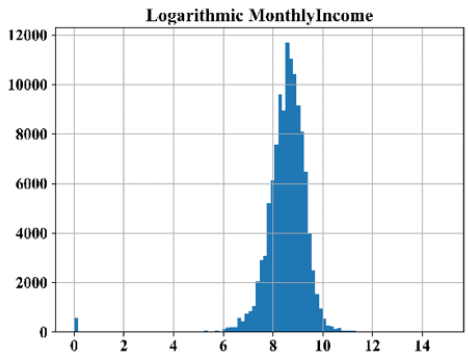


图 7

2. 0-1 特征

一方面，部分特征零与非零值有着本质的区别，比如 RUOUL、MonthlyIncome 等；另一

方面，部分特征的值在某一区间内具有某种全新的意义与属性，如低收入、未成年等。对于此类特征，可以增加对应的 0-1 特征将其强化，这是一个离散化的过程。表 2 是曾经尝试过作为 0-1 特征的特征列表，最终模型中并非所有特征均被采用。

NoIncome	月收入为 0	采用
LowIncome	月收入小于某值	
NoDependents	家属数目为 0	
YoungAge	年龄小于某值	
OldAge	年龄大于某值	
NoDebtRatio	负债率为 0	
NoRUOUL	RUOUL 值为 0	
No30-59DaysPastDueNotWorse	30-59 天内无违约	弃用↓ 无效果
No60-89DaysPastDueNotWorse	60-89 天内无违约	
No90DaysLate	90 天及之前无违约	
HighIncome	月收入大于某值	
NaNIncome	月收入数据缺失	
NaNDependents	家属数目数据缺失	

表 2

部分特征引入后，本地验证集的 AUC 值取得明显提升，如 NoRUOUL，引入后本地验证集 AUC 值提升约 0.1，kaggle 评分提升约 0.12，模型性能明显提升。而部分特征是否引入并无明显区别，如天数内无违约的特征，引入后 AUC 值几乎不变（甚至因 overfitting 导致降低），此类特征在 xgboost 的 feature importance 中得分极低，甚至为 0。

另外这些 0-1 特征中，有些特征的判定依据存在待给定参数，如高低收入、年龄。对于 YoungAge 与 OldAge，其判定边界由现实知识确定，分别为 24,65。而对于 LowIncome 的边界，训练集月收入后 10%的位置的值约为 200，对 200 附近的数进行尝试，通过分箱后的 WOE 值判断边界值的好坏，最终确定将 180 作为 LowIncome 的判断边界。

LowIncome	月收入小于 180
YoungAge	年龄小于 24
OldAge	年龄大于 65

3. 特征间的关系

部分特征间相互作用可以形成新的有意义特征。

如 DebtRatio 与 MonthlyIncome，它们间相乘可以得到类似“负债金额”意义的特征 Debt，取对数后将 LogDebt 作为新特征加入数据集。事实上，本地验证集表明使用 DebtRatio 与 LogMonthlyIncome 的乘积效果会更好。与不引入 LogDebt 特征相比，此方案在本地的验证集上得到了约 0.05 的 AUC 值提升。

另一方面，受金融业内人士的启发，可以将月收入与负债金额平均到每个家属上，即 MonthlyIncome 与 Debt 分别除以 NumberOfDependents，得到 MonthlyInomePerPerson 与 DebtPerPerson 两个新特征。本地验证集 AUC 值表明改用 LogMonthlyInome 与 LogDebt 的效果更好，采用此方案获得了约 0.03 的 AUC 值。

图 8 展示了 LogIncomePerPerson 与 LogDebt 两种特征的分布图，其分布较为自然，可以用于训练。

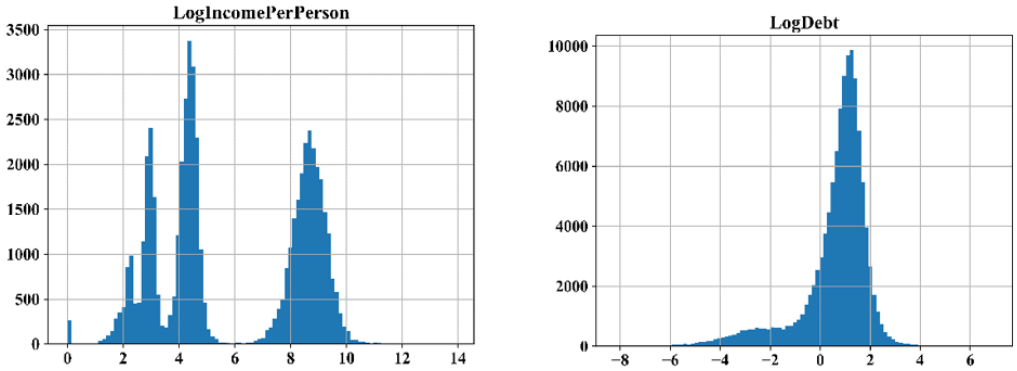


图 8

4. 特殊数值处理

通过以上方案对数据集的特征作了增添与变换，但其变换过程中会出现一些需要处理特殊值，它们主要来源于：训练集本身部分条目 MonthlyIncome 与 NumberOfDependents 的缺失；对 0 取对数造成的无穷值；除以 0 造成的无穷值或 NaN 值。

对于数据集中原本的缺失值，曾经尝试使用 CART 进行补全，代码请参考【待定】，即 NumberOfDependents 用中位数补全，后抽取 MonthlyIncome 未缺失的条目作为训练集，用随机森林回归器学习月收入并用于预测 MonthlyIncome 缺失的条目，将预测值作为 MonthlyIncome 的值进行补全。与直接用 0 补全相比，本地验证集的 AUC 值提升了约 0.1，但 kaggle 评分大幅度下降，应当是补全过程中对训练集数据过拟合，导致训练过程模型性能下降。最终弃用此方法。

最后直接采用的以 0 补全，由于 NoDependents 与 NoIncome 特征的存在，数据本身是 0（下称为“真零”）与补全出的 0（下称为“假零”）得以区分，由此可以得到模型另一个优点：训练出的模型可以预测家属数目或月收入缺失的数据。

对于 0 取对数以及除以 0 造成的问题。经检验，所有出现此问题的特征，均有一个对应的 0-1 特征作为标记。因此，可以直接用 0 替换这些无穷值与 NaN 值，这些“假零”与“真零”可以通过 0-1 特征得以区分。对于 0 取对数造成的无穷值，我们尝试过将其设置为比非无穷最小值还小的常数，但性能不如直接用 0 修正。

```
data[ fname ].fillna( 0 , inplace = True )
data.loc[ ~np.isfinite( data[ 'Log' + fname ] ), 'Log' + fname] = 0
```

5. 特征修改小结

如表 3，最终经特征修改数据集特征列表共 20 个，是原特征数目两倍，图 9 是修改特征后，任意两特征间相关系数的热度图，相关系数越接近 0 即协方差越接近 0，越能代表线性意义上的独立。观察发现除由某特征派生出的 0-1 特征外，其余任意两个特征间无明显的线性相关。

NumberOfTime30-59DaysPastDueNotWorse	原数据集
NumberOfTime60-89DaysPastDueNotWorse	
NumberOfTimes90DaysLate	
NumberOfOpenCreditLinesAndLoans	
NumberRealEstateLoansOrLines	
NumberOfDependents	
LogIncome	对数变换
LogAge	
LogDebtRatio	
LogRevolvingUtilizationOfUnsecuredLines	
LogDebt	不同特征间运算
LogIncomePerPerson	
LogDebtPerPerson	
NoIncome	0-1 特征（体现有无）
NoRevolvingUtilizationOfUnsecuredLines	
NoDebtRatio	
NoDependents	
YoungAge	0-1 特征（体现范围）
OldAge	
LowIncome	

表 3

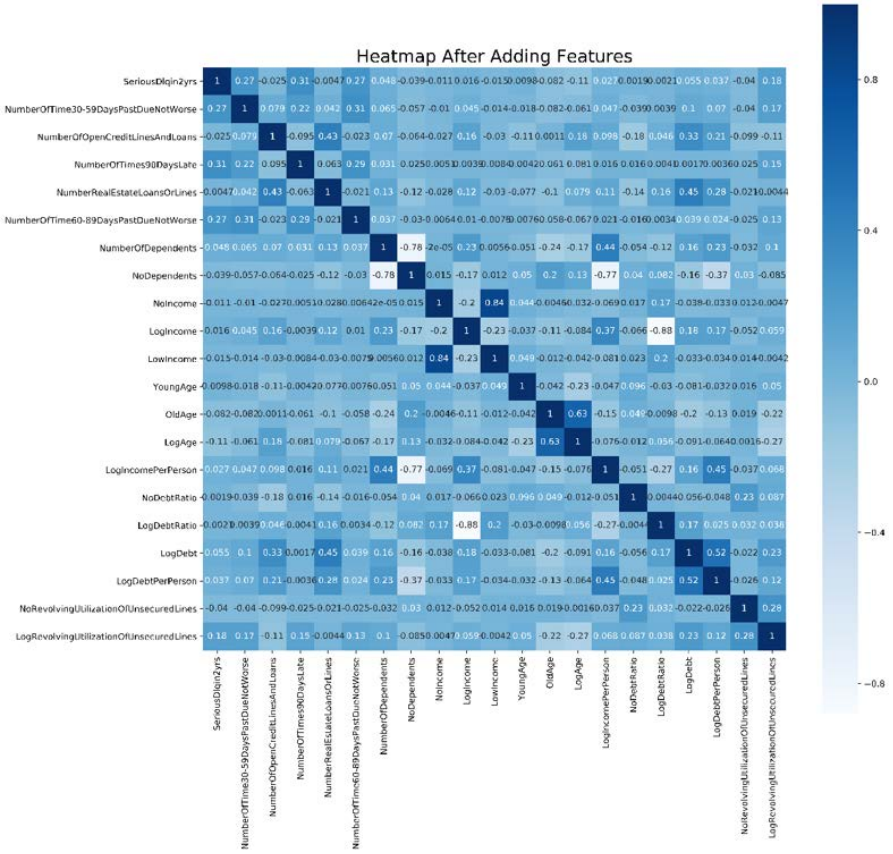


图 9

iv. 重采样

目前为止，我们已经得到了可以用于训练的数据集，但是观察发现：SeriousDlqin2yrs 值为 1 的只占训练集不到 6.7%，也即两年内严重违约的比例只占小部分，这是符合现实的。从算法的角度讲，CART、GDBT 等算法对不平衡的数据集训练，通过调整权重可以解决此问题，但是神经网络等模型，不平衡的训练数据将导致最终结果出现明显的偏差。为了有更多的模型作为训练的选择，应考虑对数据集进行平衡处理。

降采样方法使得数据集数量级缩小，本地验证集 AUC 值大幅下降，因此考虑过采样。

1. SMOTE 及变种

此方法可以根据分析少数类分布或少数类与多数类边界，补全新的少数类数据。实验表明，本地验证集的 AUC 值升高，但 kaggle 评分明显降低，应当是 SMOTE 补全导致原数据集出现不合理的数据，使得模型过拟合，最终弃用此方法。

2. ROS 过采样

即最原始的过采样方法，随机重复少量数据直到平衡。采用此方法使得 xgboost 分类器的 AUC 值无明显变化，而神经网络的 AUC 值升高约 1.0，性能明显提升。最终使用 ROS 过采样使得数据集平衡。

v. 标准化

由于神经网络隐藏层与输出层均将采用 sigmoid 激活函数，未经标准化的输入将导致梯度消失，同时严重影响收敛速度与最终性能，因此对于神经网络，输入数据须经过标准化。

而本地验证集的测试结果表明：对于 CART、GDBT 及使用 xgboost 训练的模型，标准化后的 AUC 值小幅度降低。可能由于部分特征如 LogRUOUL 分布不对称，导致归一化后因部分离群值使得大量值分布较为集中，影响树的分裂增益。

综合以上两点，最终使用非神经网络模型的输入数据未经标准化，而由神经网络模型训练的输入数据经过了标准化。

b) 模型训练

使得分类器在测试集的 AUC 值尽可能高，同时使得打分结果的分布宏观上有一定的区分度。通常，评分公式为如下形式：

$$score = b_0 - b\ln OR, \qquad OR = \frac{p}{1-p}$$

i. WOE 评分模型

WOE 评分卡是金融领域使用最广泛的评分模型，通过对特征进行 WOE 变换并使用线性分类器分类。在使用 sigmoid 激活函数的情况下，lnOR 恰好是分类器系数与对应特征 WOE 指数的内积，从而分类器系数与得分有直接的线性关系，因此 WOE 评分卡的最大优势在于其评分理据相当直观。

但是，由于线性分类器分类能力有限且 WOE 变换的分箱策略繁多，此类模型的性能并非一定更优。为了尝试与线性分类器相比更为复杂的模型，评分器未采用传统的 WOE 评分卡。

ii. 模型设计与验证方法

用训练数据独立地训练不同的模型，这里的不同可以指学习算法的不同，也可以指模型超参数的不同。后将这些模型独立的预测结果输入同一个线性分类器，将线性分类器的结果作为最终输出。这些独立的模型包括但不限于 CART、GDBT、XBG、神经网络等，将依照本地训练集、验证集的表现决定是否选取对应模型，以及模型超参数的选取。

学习过程中，首先随机选取所提供训练集的 30%作为验证集，其余 70%作为训练集，整个超参数调试过程均不参与训练。后训练除神经网络外的所有模型（不包括最后的线性分类器）时，均对训练集使用 StratifiedKFold 交叉验证(k=5)，每交叉验证一次的同时用当前训练出参数去测试验证集。独立训练完成所有模型后，将训练集在所有模型的预测值存入数组，用以训练线性分类器，完成后将验证集于所有模型上预测结果的平均值输入线性分类器预测，得到验证集的预测结果。超参数调试过程中，验证集是先前已经划分的验证集；而超参数调试完成最终要预测测试集时，训练集将是所提供的完整训练集，而所提供的测试集将代替原验证集进行预测。

注意到验证集大小有限（约 8.4 万），在超参数调试过程中，ROS 及其余随机因素对验

证集甚至训练集的 AUC 值均有影响，约有±0.5 范围的波动。因此，对 5 次相同训练过程得到的 AUC 值取平均，作为评定超参数性能的指标。前文与后文所述的验证集 AUC 值均为 5 次结果的平均值。

iii. 最终模型

经超参数调优后，最终确定的模型结构为：独立训练两个参数不同的 XGB 分类器，后将预测结果输入，如图 10 所示。同时，代码中有 apply_nn 参数，如果设置为真将独立训练 XGB 与一个全相连神经网络，并将此三个模型的预测结果作为线性分类器的输入。

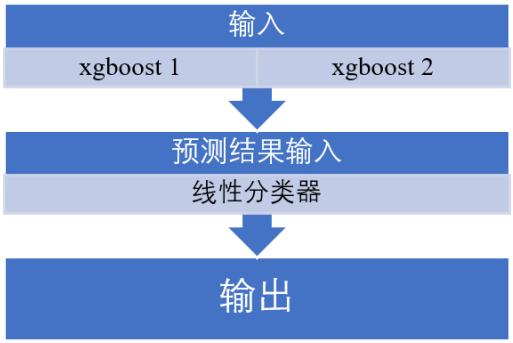


图 10

参数调优过程中，XGB 由于支持并行训练因此速度较快，而 CART 与 GBDT 的训练速度很慢使得调试周期较长，且进行的超参数调试均未使得它们的模型性能接近或超越 XGB，因此未采用，代码中 CART 与 GBDT 的超参数经过一定程度的调优。

对于神经网络，验证集 AUC 值表明神经网络对最终结果有轻微改善，而 kaggle 上的评分小幅度下降。故而将是否引入神经网络作为可选参数。

神经网络的结构如图 11 所示输入层与特征个数一致，有一个 25 单元的隐藏层，后全相联至 1 个神经元。两个全相连层均使用 sigmoid 激活函数与 l2 正则化，正则系数 0.0001；隐藏层使用 dropout 正则化，失活概率为 0.1；采用二分类交叉熵损失函数；使用随机梯度下降，动量 0.99，批大小 512，训练 10 个 epoch，起始学习率 0.1，每 5 个 epoch 除以 5。注意到使用 dropout 十分重要，否则将导致显著的过拟合。

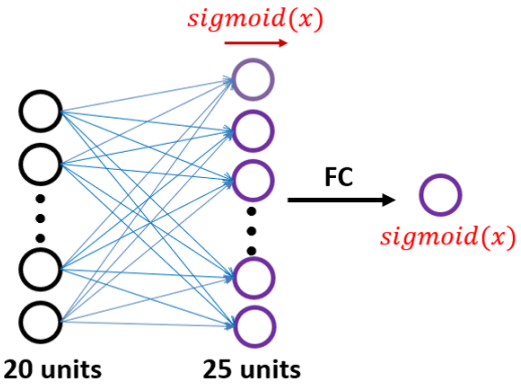


图 11

最终仅使用两个 XGB 分类器结合线性分类器，最高得到 kaggle 评分 0.86780(private board)与 0.86048(public board)，在 private board 上排名 65/943。

```
model = models.Sequential([
    Dense(25, activation='sigmoid', kernel_regularizer=regularizers.l2(_wd)),
    Dropout(0.1),
    Dense(1, activation='sigmoid', kernel_regularizer=regularizers.l2(_wd))
])
model.compile(optimizer=SGD(lr=0.05, momentum=0.99), loss='binary_crossentropy', metrics=['accuracy'])
model.fit(X, y, epochs=e, batch_size=bs, callbacks=[LearningRateScheduler(lambda e: 0.05 if e <= 5 else 0.01)])

classifiers['xgb1'] = xgb.XGBClassifier(max_depth=3, n_estimators=320, learning_rate=0.05)
classifiers['xgb2'] = xgb.XGBClassifier(max_depth=3, n_estimators=300, learning_rate=0.05)
```

iv. 特征重要程度

XGB 学习出的模型中，可以统计某特征在每次分裂节点时带来的总增益，对该特征被用来分裂节点的次数取平均，得到的值可以认为与特征的重要程度正相关，可用该指标评估对应特征的重要程度，即 gain 值。由于两个 XGB 输出后又输入线性分类器，故用线性分类器中对应系数进行加权，最后结果如图 15 所示。

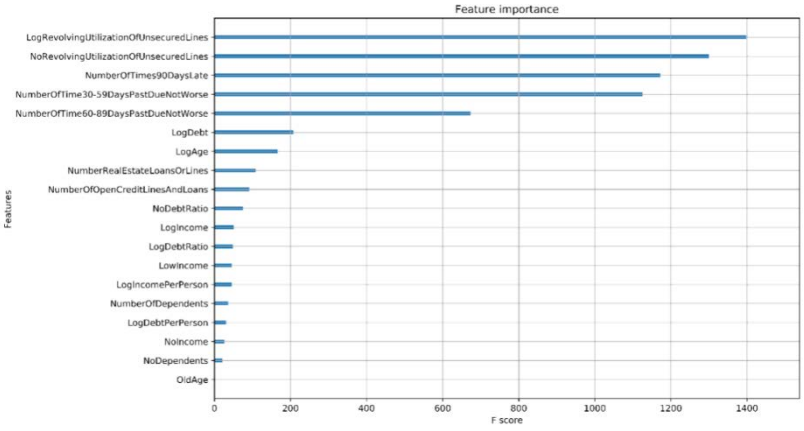


图 15

观察发现：特征 LogRUOUL 与 NoRUOUL 具有极高的重要程度，可以解释引入 NoRUOUL 后模型性能明显提升的原因。而近期的违约次数也是重要的因素。注意到 YoungAge 特征未出现在此图而 OldAge 重要程度很低，因此还可考虑舍弃上述两个特征。

c）评分器生成

i. 评分公式

$$score = b_0 - b\ln OR, \qquad OR = \frac{p}{1-p}$$

最终得分的范围为[300,900]。式中，p为模型的输出，可将其解释为该客户两年内严重违约的概率。假定基准：违约概率为 0.5 时得分为 600，因此$b_0 = 600$，还需确定b。

与 WOE 评分器相比，本模型的预测过程较为黑盒化，难以直接把控评分区间。注意到有$\sigma_{score} = b \cdot \sigma_{\ln OR}$，考虑采用统计学方法确定$b$，下研究所给测试集上的预测值分布。

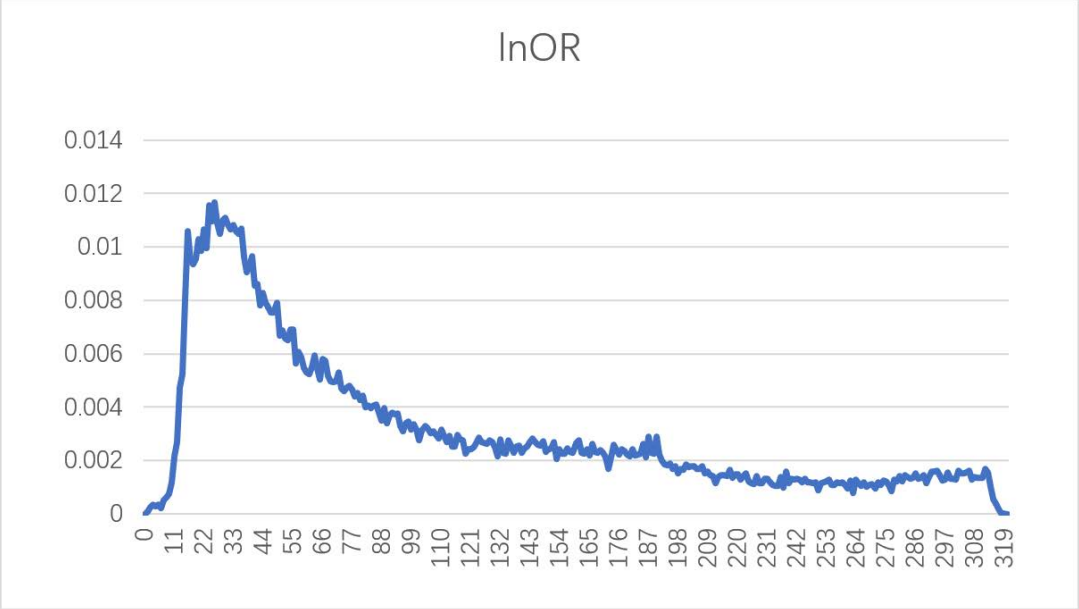


图 12

$\ln OR$值的在各区间的频率如图 12 所示，观察到曲线可能近似于泊松分布、麦克斯韦速率分布、卡方分布（伽马分布）。为验证猜想，可采用拟合优度检验中的卡方检验。

ii. 分布参数估计

1. 泊松分布与卡方分布

泊松分布：$P(X=k) = \frac{\lambda^k}{k!}e^{-\lambda}, (k \geq 0).$

卡方分布：$p(x) = \frac{1}{2^{\frac{n}{2}}\Gamma\left(\frac{n}{2}\right)} x^{\frac{n}{2}-1} e^{-\frac{x}{2}}, (x > 0).$

由于卡方检验首先需要通过最大似然估计确定分布参数，而这两种分布的似然函数最值相对较难计算。

而若采用矩估计，参数估计：λ=n=E(X)≈103.6。考虑其峰值位置，泊松分布峰值于 λ 处取得，而卡方分布峰值于 n-2 处取得，与图示分布曲线严重不符。

因此退一步，根据其峰值位置估计参数，粗略得到λ=27 与 n=38。

2. 麦克斯韦速率分布

麦克斯韦速率分布：$p(x)=\frac{4}{\sqrt{\pi}}a^3e^{-a^2x^2}x^2, x>0$，用最大似然估计来估计$a$.

似然函数：$L=\left(\frac{4}{\sqrt{\pi}}a^3\right)^n e^{-a^2\sum x^2}\prod x^2$

最大值时满足：$0=\frac{\partial \ln L}{\partial a}=\left(\frac{4}{\sqrt{\pi}}\right)^n\left(\frac{3n}{a}-2a\sum x^2\right)$，即$a=\sqrt{\frac{3}{2E(X^2)}}\approx 0.0187062$

iii. 卡方检验

对于所给测试集，其有$n=101503$个条目，$\sqrt{n}\approx 318.6$，将$\ln OR$的分布区间 319 等分，在首尾各延伸一个等宽的区间，则共有$m=319+2=321$个等距区间。

设V_i表示$\ln OR$中落入第i个区间$(i=1,2,\cdots,321)$的样本个数，p_i表示对应分布落入对应区间的概率，于是：

$$V=\sum_{i=1}^{m}\left(\frac{V_i}{n}-p_i\right)^2\frac{n}{p_i}\sim\chi^2(m-1-1)\sim\chi^2(319)$$

否定域：$W(\vec{x}:V>c), P(\chi^2_{319}>c)=\alpha$.

于是，可以得到如表 4 所示的数据。

分布	V	p值
泊松分布	1290.7836574324556	~0.0
麦克斯韦速率分布	**280.18929425914297**	**0.9426026**
卡方分布	805.167803087271	~0.0

表 4

因此，只有在显著性水平$\alpha>0.9426026$时，才可否定零假设“$\ln OR$不服从麦克斯韦速

率分布”。由于此处仅是近似估计，故在某种意义下，可以认为$\ln OR$服从麦克斯韦速率分布。图 13 给出了几种分布曲线与$\ln OR$分布曲线的对比，其中 Maxwell2 对应矩估计得到的分布参数。

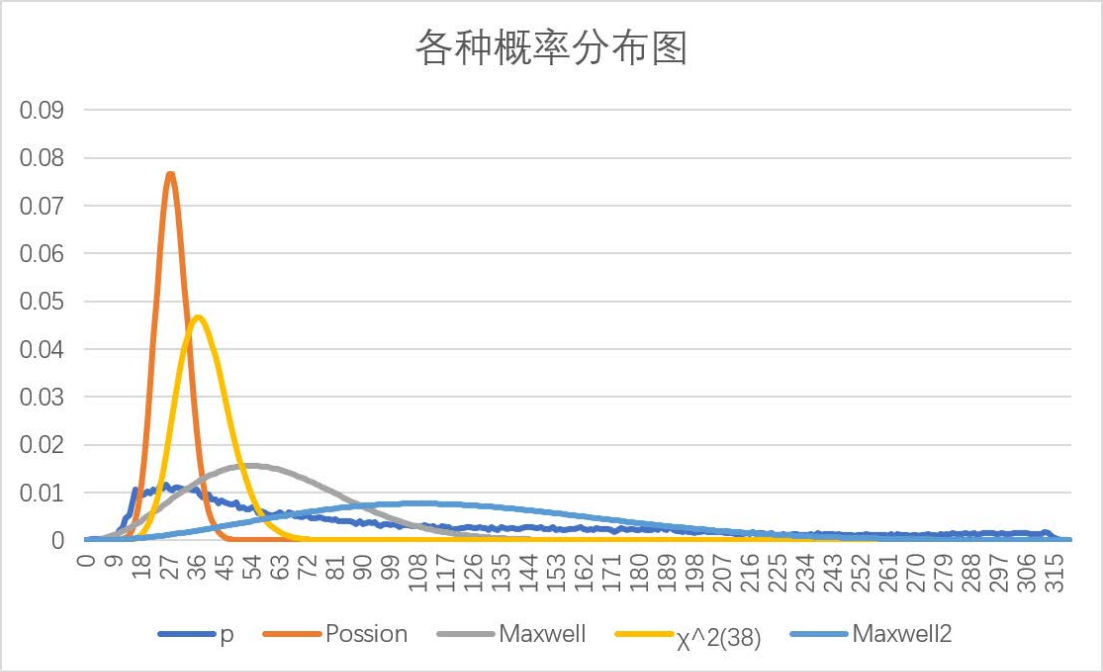


图 13

iv. 模型参数估计

假定$\ln OR$服从麦克斯韦速率分布，期望得到如下事实：“评分不属于[300,900]的概率不超过 0.1%”。即：某个麦克斯韦速率分布在一个宽为 600 的区间上，对概率密度函数积分不小于 0.999。由于麦克斯韦速率分布的峰值点为a^{-1}，不妨假定积分起点$\sqrt{a^{-1}}$，即：

$$\int_{\sqrt{a^{-1}}}^{\sqrt{a^{-1}}+600} PDF_{Maxwell}(x)dx \geq 0.999$$

最终得到：

$$a \approx 0.0106$$

于是，此麦克斯韦速率分布的标准差$\sigma = \sqrt{1.5}a^{-1}$，若要求$\sigma_{score} = \sigma$，则：

$$b = \frac{\sigma_{score}}{\sigma_{\ln OR}} = \frac{\sigma}{\sigma_{\ln OR}} \approx 81.836$$

v. 最终评分模型

综上所述，取$b_0 = 600, b = 81.8$，得到评分器：

$$score = 600 - 81.8 \ln OR, \qquad OR = \frac{p}{1-p}$$

于是最终得分：

$$SCORE = \min\{\max\{score, 300\}, 900\}$$

根据此评分卡对所提供的测试集所以条目评分，得到直方图如图 14。此得分的分布具有一定的区分度，违约概率越小得分越高，可以实际应用。

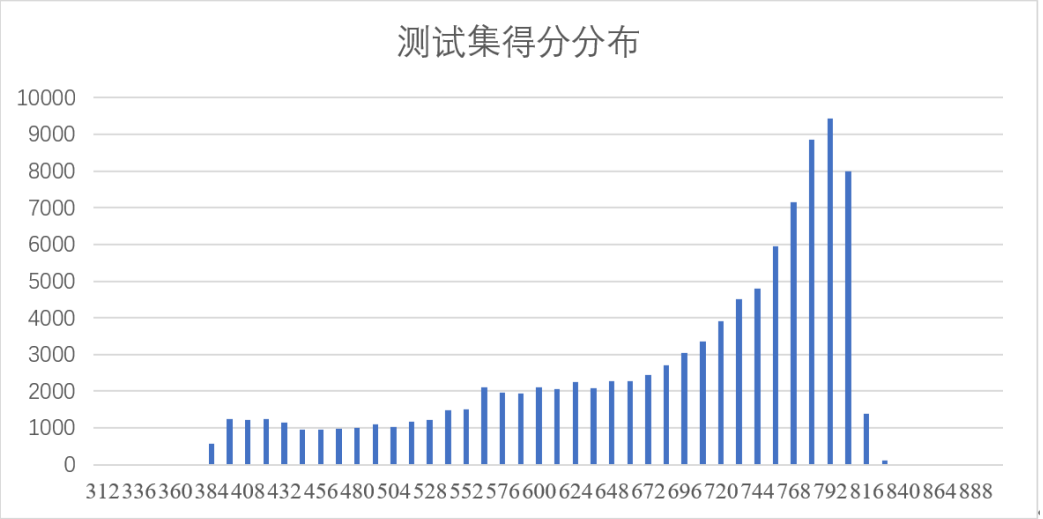


图 14

3. 总结

本文实现了一个给定客户数据，用以估计该客户两年内是否会严重违约概率的分类器。并利用此模型，结合其在实际数据上的 OR 值分布，利用拟合优度检验得出其近似于麦克斯韦速率分布，并分析得到评分器公式$score = 600 - 81.8 \ln \frac{p}{1-p}$。分类器的 AUC 值达到了 0.86780，评分卡分数与违约概率负相关且具有一定的区分度，有实际应用意义，缺点是评分过程不透明。

另一方面，神经网络在此模型中并未完全发挥出其功效，可能需要为原增添更多有意义的特征，同时需要更多种类的模型尝试与更广范围的超参数调优。另外数据清洗的并未删除过多的异常数据，利用机器学习的数据补全过程也可以设立验证集并进行超参数调优。

总体来讲，本次实验过程中尝试了多种数据处理的方式，并进行了多种分类模型的对比和反复的参数优化，最终取得了较好的实验结果。并从统计分析的角度出发，得出了一组符合当前分类模型的评分器参数。最终给出了一套从数据预处理、模型训练到评分器设计的完整设计思路，具有一定的参考价值。

4. 分工

由于涉及隐私信息，本部分内容被隐藏。

5. 鸣谢

由于涉及隐私信息，本部分内容被隐藏。

6. 附录

提交文件列表

目录	文件名	备注
./	Visualization.ipynb	用于数据可视化的 Jupyter notebook
	TrainingAndTesting.ipynb	用于训练、测试模型的 Jupyter notebook
	submit-final.csv	Kaggle 上 private board 评分 0.86780 的最好一次测试集预测结果
./abandoned	RandomForest.py	随机森林法进行缺失数据补全的代码，已弃用
./utils	draw.py	一些可视化代码
	其余文件	拟合优度检验过程中各辅助程序，具体作用请见每个文件开头的注释
./data	score.csv	测试集的评分结果
	score_bin.xlsx	OR 值分箱后，各箱内的分布频率及其曲线
	bin.csv	泊松分布、麦克斯韦速率分布、卡方分布的分布对比
	distribution.xlsx	
./images	所有文件	draw.py 绘制出的图