队伍编号：A13

四川省大学生数据科学与统计建模竞赛

复赛论文

西南财经大学　　新网银行

2022 年 11 月

（中文）论文题目（2 号华文中宋加粗）

队　　长：兰文婕

队　　员 1：陈欣阳

队　　员 2：饶翰宇

队　　员 3：

指 导 教 师：王俊

队 长 电 话：

四川省大学生数据科学与统计建模竞赛

论文原创性及知识产权声明

本人郑重声明：所呈交的竞赛论文，是本团队（本人）在指导教师的指导下进行研究工作所取得的成果。除文中已经注明引用的内容外，本论文不含任何其他个人或集体已经发表或撰写过的作品成果。对本文的研究做出重要贡献的个人和集体，均已在文中以明确方式标明，因本竞赛论文引起的法律结果完全由本人承担。

本人同意竞赛期间撰写论文的知识产权属竞赛组委会及本人共有。本人完全了解竞赛组委会有关保留、使用竞赛论文的规定，竞赛组委会有权保留并向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅。本人授权竞赛组委会可以采用影印、缩印、数字化或其他复制手段保存和汇编本竞赛论文。

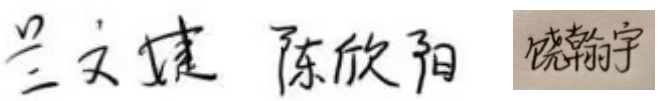
本竞赛论文属于

1、□保密，在___年解密后适用本授权书。

2、☑不保密

特此声明。

团队成员签名：



2022 年 11 月 14 日

摘要

本文中对银行客户是否会发生首次违约的二分类预测问题搭建了**Stacking+Voting 的集成方法**，对多个模型进行堆叠搭建出**三个子集成模型**，并对结果进行预测，再将每个模型的预测结果进行**投票集成**，得到最终的结果，以此增强模型的鲁棒性，最终提高了模型的拟合优度。首先在原始数据集的基础上，通过交叉验证的方式寻找出单个模型的 Baseline，评价指标为 AUC 和 F1 值。随后对单个基模型进行网格搜索找到最佳超参数，将多个基模型进行 stacking 堆叠，行成单个 stacking 模型。形成了包含不同基模型、运用不同集成方法的不同质的三个子模型，三个子集成模型分别对测试集结果预测，再将预测结果标准化后进行加权投票融合，得到最终的预测结果。

关键词：集成模型 机器学习 违约预测

目录

目录

1.研究报告背景

小微企业是国民经济的基本细胞，小微活、就业旺、经济兴。习近平总书记高度重视小微企业发展，多次作出重要指示批示，强调中小企业能办大事，要加强对困难行业和中小微企业扶持，加强对企业创新的支持，培育一批“专精特新”中小企业。李克强总理多次主持召开国务院常务会议，部署出台一系列惠企政策，纾解企业特别是民营、小微企业困难，支持小微企业健康发展。2022 年五月，国务院促进中小企业发展工作领导小组办公室印发了《加力帮扶中小微企业纾困解难若干措施》，提出“安排纾困专项资金、新增普惠型小微企业贷款 1.6 万亿元、将处于产业链关键节点的中小微企业纳入重点产业链供应链‘白名单’”等十项举措。

新冠肺炎疫情暴发以来，小微企业面临订单量减少，市场需求降温等经营窘境。与此同时，承担风险能力较弱，融资风险大等瓶颈困扰着小微企业的长足发展，也加剧了其对信贷需求的增加。但当前金融机构普遍未全面建立起专门适用于小微企业的信贷风险评估体系，小微企业本身的贷款需求也具有“短，小，频，快”，难以控制风险的特点，导致小微企业信贷风险较为突出，表现为小微企业不良贷款率高于同期商业银行其他贷款的不良率。因此，开发效果好、稳定性高的小微企业信贷风控模型来对客户的早期风险进行识别，能够提高金融机构在小微企业信贷风控中的信用风险识别和防范能力。

2.文献研究

国外学者较早开始研究小微企业的信贷风险控制，主要围绕风险分析和量化进行研究。早在 1981 年，A.Weiss 和 J.E.Stiglitz 就从理论层面指出

金融风险的产生与信贷市场上的信息不对成密切相关[1]。
Jose.A.G.Baptista 等学者对小微企业贷款风险的研究则采用多元回归分析的理论，共归因总结出八个原因进行说明[3]。Kolari 以 Logit 模型分析法为基础，分析美国大型商业银行的信贷风险评价[5]。Wang 和 Cox 通过多元判别分析法进行分析，提出监管人的尽责监督对于风险管控具有较大的意义。[6]Odom 和 Shrada 提出一个比判别分析模型更加可靠的模型，这个模型就是神经网络模型。运用该模型可以测算小微企业破产的概率。[11]She 和 Tzeng 在 2015 年开发了以决策准则为基础的计算模型，提高了贷款风险的预测准确度及防控效率。[14]

我国小微企业研究则开始得较晚。随着改革开放不断深入，以小微企业为主的民营经济越来越活跃在舞台上，国内学长们开始重视和研究小微企业的发展问题。近年来，小微企业融资难，融资贵的问题与小微企业日益迫切的贷款需求形成巨大反差。学者们对小微企业信贷风险控制的研究围绕小微企业的风险预防和评价展开。从小微企业的风险预防来看，王利军（2009）把信贷风险的控制研究放在贷前、贷中和贷后三个方面，并把这些手段运用在信贷预防上。[15]梁彩虹（2014）分析了小微企业的信贷风险特点，提出防范信贷风险需要做到客户资料审核要严，贷款审批流程要细，贷后监控要实等。[17]苏蕙和郭炜（2019）分析 Z 银行现行信贷风险评价指标体系存在不足，并进行优化，以求更科学、准确地评价小微企业信贷风险。[29] 肖宇辰和余舜基（2020）根据供应链金融模式的特点，构建中小企业信用风险评价模型，并采用主成分分析法和 Logistic 模型进行实证研究，并结合实证结果，提出相关建议。[30] 邵黎等（2020）学者认为大数据对于金融机构风险防控具有重要作用。[21]并探讨了当前金融机构大数据信贷风控的现状，结合未来金融机构大数据信贷风险防控的重点，提出商业银行改进大数据小微企业信贷风险防控模式的对策建议。国内学者在案例分析与实证分析相结合的研究中不断探索更有效的中小微企业信贷风控方法。

基于以上研究所提到的模型，金融机构也集思广益地采用统计和机器学习模型对小微企业风险进行总体评估和剖析，例如 Z 分析模型、主成分分析法、敏感性分析法、回归树分析法等，均取得了一定的成效。但是在大数据

时代背景下，因子分析、主成分分析等方法会对客户的数据进行降维，可能会造成一定的数据丢失，降低信贷风险控制的准确率和效果；Logistic 回归，决策树分析等模型又存在其迁移能力或泛化能力对于不同地区、或具有不同特征的客户进行信贷风险控制不够稳定的问题，其次单个模型本身也有一定的缺点，在学习的过程中也会出现一定的学习误差。

本研究报告旨在基于所提供的小微企业法基本信息、历史借贷信息、申请行为信息、工商司法信息以及贷款申请后的早期风险表现数据，构建效果好、稳定性高的小微企业早期风险识别模型，对客户早期风险进行识别，帮助金融机构进一步提升模型在小做企业信贷风控中的信用风险识别和防范能力。

3.赛题分析

3.1 赛题背景资料

小微风控算法大赛-早期风险识别赛道所提供的数据包含经过脱敏处理的小微企业法人基本信息、历史借贷信息、申请行为信息、工商司法信息以及贷款申请后的早期风险表现数据。本赛道需要我们基于所提供的数据，利用统计和机器学习模型，对小微企业的早期风险表现数据进行充分的探索，在此基础上构建区分效果好，稳定性高的风控模型，对客户早期风险进行识别，帮助金融机构进一步提升模型在小微企业信贷风控中的信用风险识别和防范能力。

3.2 需要解决的问题

对比赛所提供的数据集进行充分的数据挖掘，通过挖掘出的特征集，利用统计和机器学习模型在训练集进行训练，再运用训练完成的模型对测试集

数据进行概率预测。

3.3 公私榜模型评价指标

模型的评价指标为 AUC，即分别随机从正负样本集中抽取一个正样本，与一个负样本，正样本的预测值大于负样本值的概率。

$$\mathbf{AUC} = \frac{\sum(\mathbf{pred_{pos}} > \mathbf{pred_{neg}})}{\mathbf{positiveNum} * \mathbf{negativeNum}}$$

3.4 特征工程

特征工程（feature engineering）是指从原始数据中提取特征并将其转换为适合机器学习模型的格式。为了提取客户信息和做出预测，机器学习使用数学和统计学模型来拟合数据。这些模型将特征作为输入。特征就是原始早期风险表现数据某些方面的数学表示。在我们研究的机器学习模型中，特征是客户数据和风控模型之间的纽带。“数据和特征决定了机器学习的上限，而模型和算法只是逼近这个上限而已”，正确的特征可以减轻我们构建风控模型的难度，从而使风险识别输出更高质量的效果。

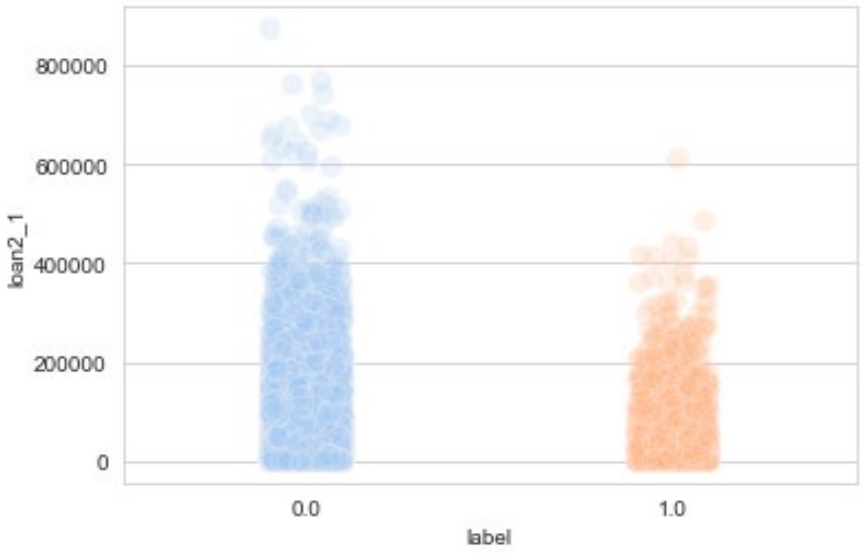
我们的特征工程包括探索性数据分析、特征理解、特征增强、特征构建和特征选择 5 个步骤，为进一步解释经过脱敏处理的早期风险表现数据并进行预测性分析做准备。

3.4.1 探索性数据分析

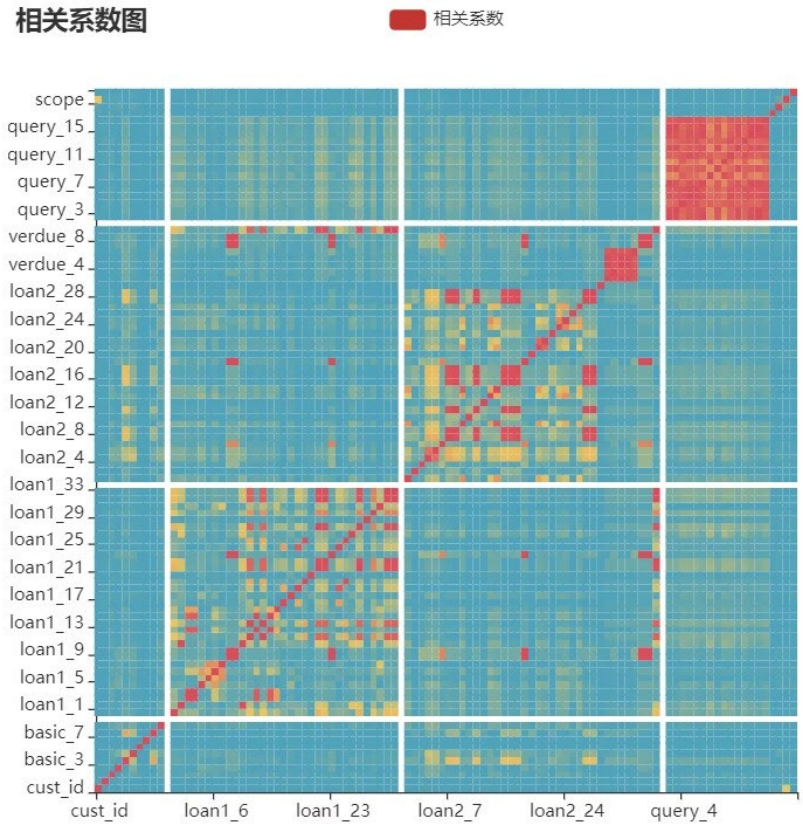
探索性数据分析（Exploratory Data Analysis）是对早期风险表现数据集进行一些基本的描述性统计，并进行可视化操作，以便更好地理解早期风险表现数据的性质。

首先我们的数据具有 100 个风险特征（人口统计特征，贷款与查询记录，

企业工商司法信息等），一个客户对应一行记录；风险标签是由客户借款后是否出现首期逾期等特征加工而成的 1 个二分类标签（0，1）；总计有 2.5 万个样本，其中训练集数量 17884，正样本占比 10%，测试集数量 7116（公榜 3522，私榜 2594），正样本占比 5%；值得注意的是相关数据经过脱敏，分层抽样、模拟转换等处理，这给我们的特征理解增加了难度，如图一所示。皮尔逊相关系数是衡量两个变量之间关系的指标，通过相关系数矩阵，我们可以清楚地看到字段之间的相关关系。除去矩阵副对角线的元素，变量之间相关程度最高集中在右上方区域，说明字段间高度相关，可以作为后续构造新的特征。如图二所示。



图一



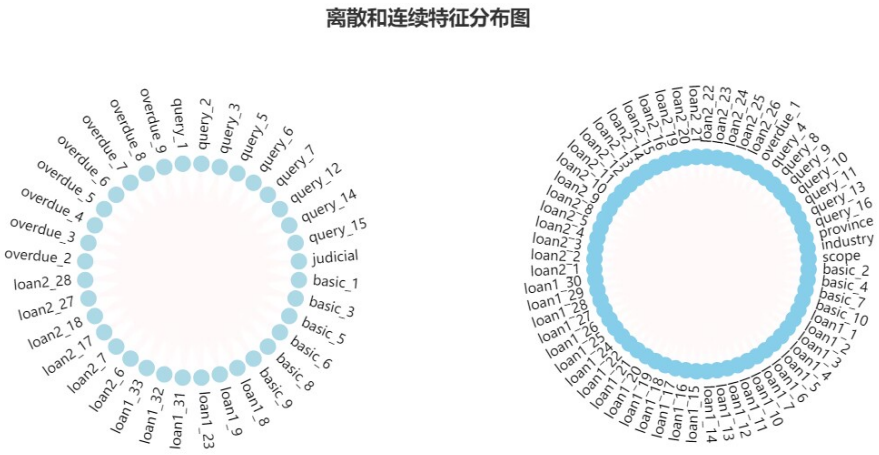
图二

3.4.2 特征理解

为了进一步理解早期风险表现数据，我们在数据集中识别并提取不同等级的数据，并用信息创造可视化图表，为后续特征工程提供指导。

首先，早期风险表现数据集全部由结构化的数据组成，训练集有 17884 行 x100 列，测试集有 3522 行 x100 列。标签‘label’列为（0，1）变量，特征列主要由‘basic’，‘loan1’，‘loan2’，‘overdue’，‘query’，‘province’，‘industry’，‘scope’，‘judicial’等数据组成。

随后我们对数据集中的定量数据和定性数据进行了统计如图三所示：

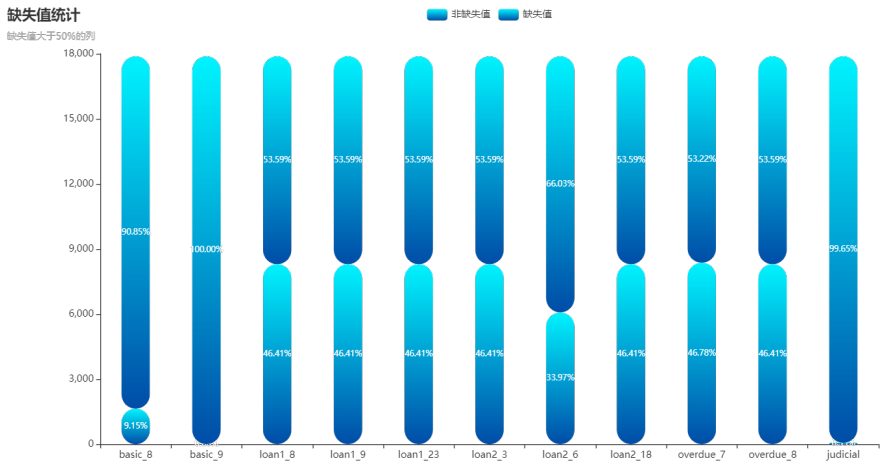


图三

对于定量数据，我们在后续的处理中会进行加减乘除等运算，来强化其特征；对于定性数据，我们会采用分箱、WOE 编码等方法来进行特征处理，以提高风控模型的区分效果和稳定性。

3.4.3 特征增强

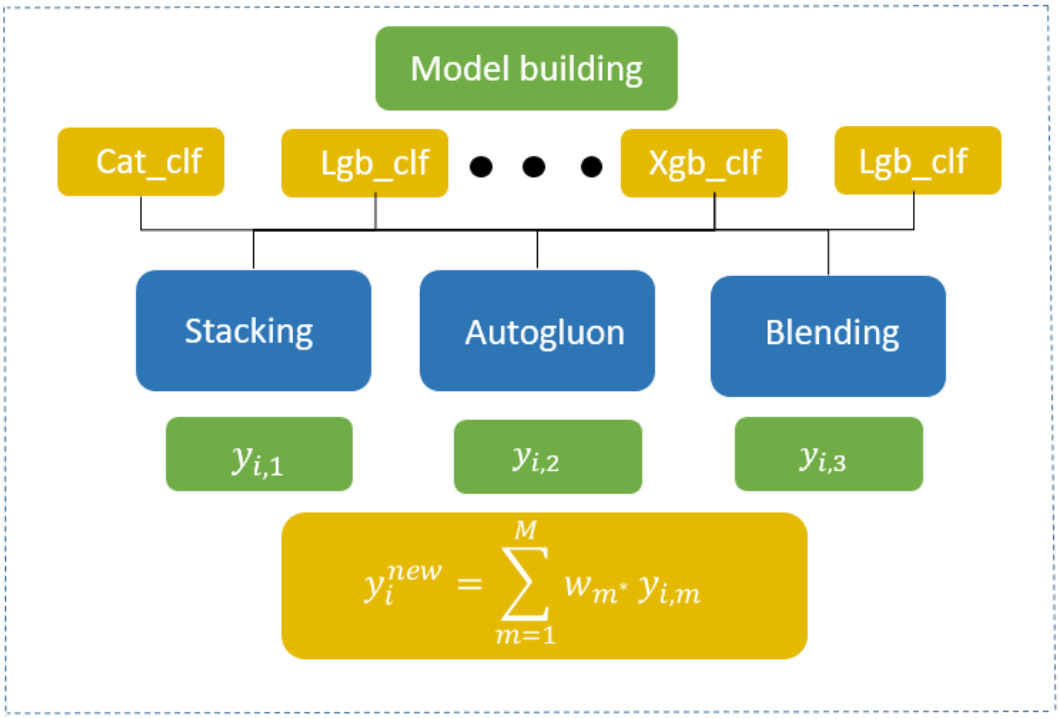
在这个阶段我们通过进行数据清洗，增强数据的方式来让模型表现出更好的效果。数据清洗是指调整已有的列和行，我们首先是识别数据中的缺失值，了解缺失的数据是什么，才决定下一步如何处理这些数据。如图四所示：



图四

4.模型建立

在这一部分，我们构建了一个复杂的三层集成模型。在第一层中，我们选取了大量的基础模型，包括在处理分类问题性能优越的 catboost、xgb、lightgbm 等机器学习模型，并利用 5 折交叉验证与网格搜索法对基础模型进行参数寻优。在模型第二层，我们将最优参数以及模型放入第二层的三个子模型中利用 Stacking、Bagging、Blending 等方法进行训练，分别得到三个测试集的预测数据。在模型的第三层中，我们对以上得到的三个预测数据进行加权投票，并得到最终的训练结果。模型结构图如图五所示：



图五

首先我们在此声明模型构建中使用的符号含义：

表格 1:符号含义

符号	说明	单位
Cat_clf	Catboostclassifier	-
Xgb_clf	XGBClassfier	-
lgb_clf	LGBMClassifier	-
Ext_clf	ExtratreeClassifier	-
Log_clf	Logistic regression classification	
Param{}	参数空间	-
clf	任意机器学习基础模型	-
X_train	训练集特征空间数据集	-
y_train	训练集标签	-
X_test	测试集特征空间数据集	-
y_prob	测试集预测标签	-

4.1 基础模型

4.1.1 cat_clf

Catboost 也是 Boosting 族算法的一种，是一种能够很好地处理类别特征的梯度提升算法库。是在 GBDT 算法框架下的一种改进实现，是一种基于对称决策树（oblivious trees）算法的参数少、支持类别型变量和高准确性的 GBDT 框架，能高效合理地处理类别型特征，处理梯度偏差（Gradient bias）以及预测偏移（Prediction shift）问题，提高算法的准确性和泛化能力。

Parameter space:cat_clf
params = {'depth':[3,1,2,6,4,5,7,8,9,10], 'iterations':[250,100,500,1000], 'learning_rate':[0.03,0.001,0.01,0.1,0.2,0.3], 'l2_leaf_reg':[3,1,5,10,100], 'border_count':[32,5,10,20,50,100,200], 'ctr_border_count':[50,5,10,20,100,200], 'thread_count':4}

4.1.2 xgb_clf

XGBoost 所应用的算法就是 gradient boosting decision tree，既可以用于分类也可以用于回归问题中。Gradient boosting 是 boosting 的其中一种方法，Gradient boosting 就是通过加入新的弱学习器，来努力纠正前面所有弱学习器的残差，最终这样多个学习器相加在一起用来进行最终预测，准确率就会比单独的一个要高。然而 gradient boosting 的实现是比较慢的，因为每次都要先构造出一个树并添加到整个模型序列中，XGBoost 很好地解决了上述问题，其重要特点就是计算速度快，模型表现好。

Parameter space:xgb_clf

```
params = {'n_estimators':[2,5,10,50,200,400],
          'max_depth':[1,3,10,20],
          'gamma': np.arange(0.0,40.0,0.005),
          'learning_rate':[0.03,0.001,0.01,0.1,0.2,0.3],
          'subsample':[1,0.9,0.5,0.1],
          }
```

4.1.3 lgb_clf

LigthGBM 也是 boosting 集合模型中的一种方法，它和 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。

Parameter space:lgb_clf

```
params = {'n_estimators':[2,5,10,50,200,400],
          'max_depth':[1,3,10,20],
           'gamma': np.arange(0.0,40.0,0.005),
          'learning_rate':[0.03,0.001,0.01,0.1,0.2,0.3],
          'subsample':[1,0.9,0.5,0.1],
          }
```

4.1.4 ext_clf

Extremely Randomized Trees Classifier 是一种集成学习技术，它将森林中收集的多个去相关决策树的结果聚集起来输出分类结果。极度随机树的每棵决策树都是由原始训练样本构建的。在每个测试节点上，每棵树都有一个随机样本，样本中有 k 个特征，每个决策树都必须从这些特征集中选择最佳特征，然后根据一些数学指标(一般是基尼指数)来拆分数据。这种随机的特征样本导致多个不相关的决策树的产生。

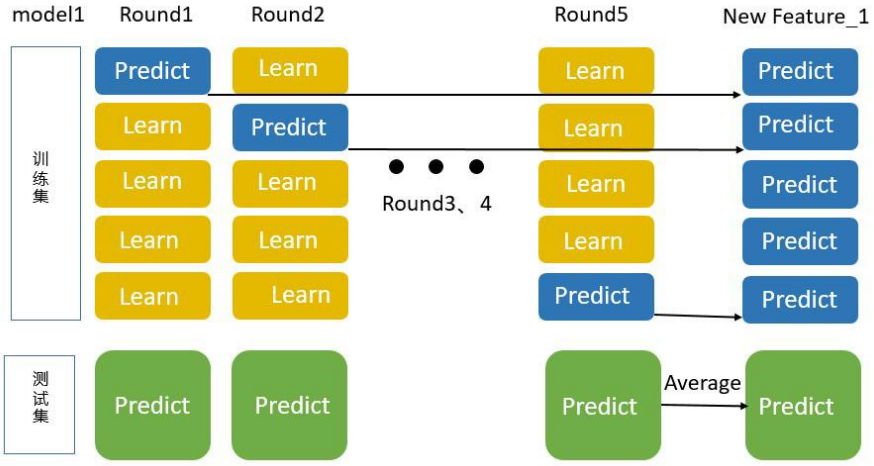
Parameter space:ext_clf

```
params = {'n_estimators':[2,5,10,50,200,400],
          'max_depth':[1,3,10,20],
          'learning_rate':[0.03,0.001,0.01,0.1,0.2,0.3],
          'min_samples_split': [2, 5, 8, 13, 34,100],
          'min_samples_leaf':[2, 5, 8, 13, 34,100],
          'max_features': ['auto', 'sqrt', 'log2', 2, 8, 13, 34, None]
          }
```

4.2 集成模型

4.2.1 模型一 Stacking

我们利用 Stacking 的思想建立模型一，选取上文中的 cat_clf、lgb_clf、xgb_clf 为第一层三个基础模型，将训练好的所有基模型对整个训练集进行预测，第 j 个基模型对第 i 个训练样本的预测值将作为新的训练集中第 i 个样本的第 j 个特征值，最后基于新的训练集进行训练，流程如图所示。同理，预测的过程也要先经过所有基模型的预测形成新的测试集，最后在模型第二层利用 log_clf 再对测试集进行预测。流程图如图六所示：



图六

模型 1：分层模型 stacking 算法

输入： X_train； y_train; x_test;param{};models={Cat_clf;Xgb_clf;Lgb_clf}
Initialize: oof_cat = []; oof_lgb = []; oof_xgb = [](Three empty column);
test_output_df=columns['lgb','xgb','cat']
for l=1 **to** L do{Stacking}
  for i =1 **to** n **do** {n-repeated}
  随机将数据集分成 5 折$\{X^j, Y^j\}_{j=1}^{5}$
    for j=1 **to** k **do** {k-fold bagging}
      for each model type_m **in** models **do**
        在数据集 X^{-j}, Y^{-j}上训练第 m 个模型
        用X^j预测$\hat{Y}_{m,i}^{j}$
      end for
  end for
OOF 平均预测值$\hat{Y}_m = \{\frac{1}{n}\sum_i \hat{Y}_{m,i}^{j}\}_{j=1}^{k}$
y_prob=logesiticClassifier{X, $\{\hat{Y}_m\}$}
end for
输出： 测试集预测概率

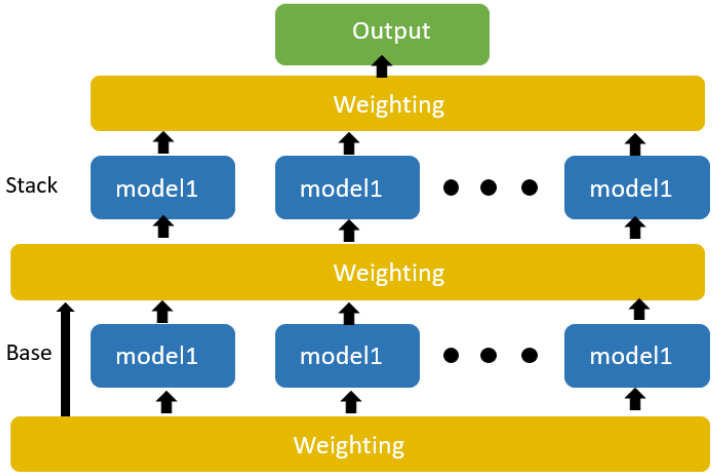
4.2.2 模型二 Stacking

在第二个 stacking 子模型中，我们采用了 Catboost、ExtraTrees、Lightgbm 分类器集成的方法，同时将网格搜索的超参数带入到模型之中，输出层为逻辑回归。相比于单个模型预测的方法，Stacking 后模型拟合优度有所提高，准确率、F1 值、AUC 等指标均得到提升。

4.2.3 模型三 AutoGluon

自动机器学习（AutoML）是将机器学习应用于现实问题的端到端流程自动化的过程。与现有的主要模型/超参数选择的 AutoML 框架不同，AutoGluon 能成功的整合多个模型并将他们堆叠在多个层中。已有实验表明，AutoGluon 组合的多个模型比人工调参的模型表现更优。在这里我们使用 AutoGluon 来自动选择模型并进行训练，在计算资源足够的情况下用一定的训练时间省去人工调参，并追求更好的模型输出。

首先 AutoGluon 会对数据进行两个阶段的预处理：模型无关的预处理（将输入特征转换为适用所有模型）和特定模型的预处理（只应用于用于训练特定模型的数据副本）。我们知道，将多个模型的预测结合在一起的集合优于单个模型，通常会大大减少最终预测的方差（Dietterich，2000）。常见的模型集成的方法有：bagging，boosting，stacking 以及 weighted combinations。在 AutoGluon 中，一组独立的“基础”模型被按照通常的方式单独训练，然后使用基本模型的聚合预测作为其特征来训练一个“stacker”模型。如下图所示，第一层有多个基本模型，其输出被连接起来，然后馈送到下一层，下一层本身包含多 stacker 模型。接着这些 stacker 作为基础模型输出到一个新的层。最后的 stacking 层应用集合选择（ensemble selection）以加权的方式聚合堆叠模型的预测。流程图如图七所示：



图七

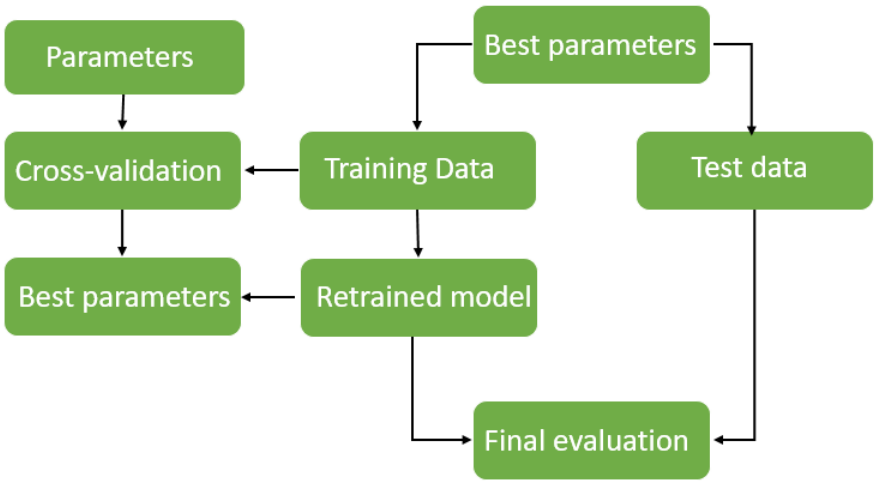
模型 1: AutoGluon 算法

输入: X_train； y_train; x_test;param{};family of models M， # of layers L
Preprocess data to extract features
for l=1 **to** L do{Stacking}
  for i =1 **to** n **do** {n-repeated}
  Randomly split data into k chunks $\{X^j, Y^j\}_{j=1}^{5}$
    for j=1 **to** k **do** {k-fold bagging}
      for each model type_m **in**M **do**
        Train a type-m model on X^{-j}, Y^{-j}
        Make predictions$\hat{Y}_{m,i}^{j}$ on OOF data X^j
      end for
    end for
end for
Average OOF predictions $\hat{Y}_m = \{\frac{1}{n}\sum_i \hat{Y}_{m,i}^{j}\}_{j=1}^{k}$
$X \leftarrow$ concentrate$(X, \{\hat{Y}_m\})$
end for
输出: 测试集预测概率

4. 3 网格寻优调节参数&交叉验证模型评估

在相同的数据集上同时调节模型参数同时对其进行测试，会发生过拟合的错误。为了避免这种情况，我们在训练（监督）机器学习模型时，采用将部分可用数据保留为测试集 X_test、y_test 的做法。

下图八为交叉验证的流程示意图，最佳参数可以通过网格搜索技术确定。

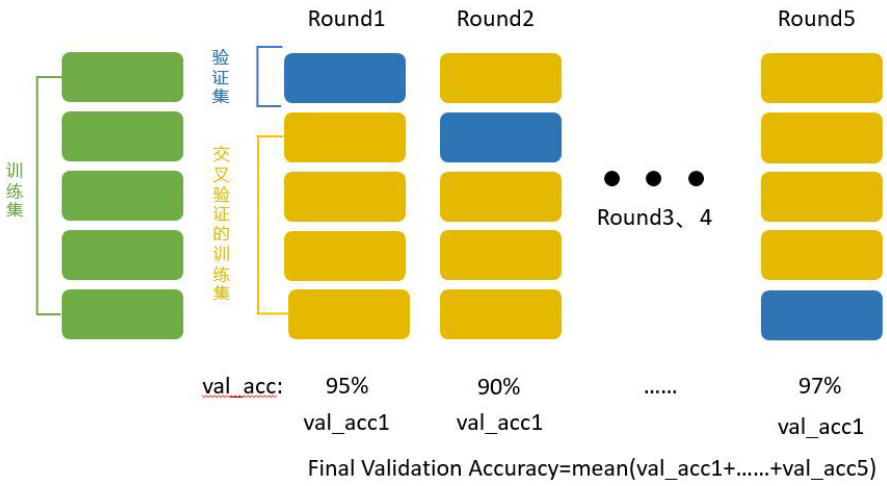


图八

基于上述原因，同时为了避免公私榜分数的剧烈变动，我们采用 k_fold 交叉验证（CV）进行参数调节。在 CV 法中，我们首先将训练集随机拆分为 k 个较小的集合。对于 k 个集合中的每一个，我们采取以下方法进行处理：

1. 在每一折的训练中，将 k-1 折数据用为训练集数据；
2. 训练完成的模型在剩余的一折数据上进行验证，计算 AUC。

示意图如图九所示：



图九

使用交叉验证的最简单方法是在分类器和数据集上调用 cross_val_score 辅助函数。

4.4 模型加权投票集成

在子模型相关性较低时，我们可以通过加权平均进一步提升模型的预测性能。

4.4.1 标准化

得到第二层三个子模型的预测结果后，我们对三个预测数据进行加权融合，因为当平均多个来自不同模型的输出时，会存在一些问题。不同预测器会产生过高或者过低的预测概率，或者在一定范围内预测结果表现不佳。例如：

Cust_id	label	Cust_id	label
1	0.310056	1	0.57
2	0.310002	2	0.12
3	0.310012	3	0.03
AUC	0.782		0.775

左边的模型在 AUC 的评价指标下，取得了比左边模型更高的得分。但当他们进行简单的平均后，投票并不会进一步提升模型的预测效果。为了解决这个问题，我们先将预测结果在[0,1]之间进行标准化，得到一个均匀分布的预测值，再对结果进行平均。

$$\widehat{y_i} = \frac{y_i - \max\{Y\}}{\max\{Y\} - \min\{Y\}}$$

其中，y_i为预测集中第 i 个对象的预测评分，$\widehat{y_i}$为经过标准化后第 i 个对象的预测评分，Y 为原始预测集。

4.4.2 加权平均法

子模型的预测准确率越高，其权重应该越大，进一步看，当模型效果较差的子模型想要否定预测结果较好的模型时，只有当大多数模型都同意另一种选择。这样通过加权平均的方法可以为模型带来一定效果的提升。

首先获得子模型的公榜得分后，根据公榜得分计算第 m 个子模型的权重：

$$w_m = \frac{AUC_m}{\sum_{m=1}^{M} AUC_m}$$

其中w_m是第 m 个子模型的权重，AUC_m为第 m 个子模型的公榜得分。

$$y_i^{new} = \sum_{m=1}^{M} w_m * \hat{y}_{m,i}$$

其中$\hat{y}_{m,i}$是第 m 个子模型经过标准化后对第 i 个对象的结果预测值，AUC_m为第 m 个子模型的公榜得分，y_i^{new}是经过标准化加权平均后的最终预测值。

5.结果分析

5.1 基础模型的最优参数以及 local cv

5.1.1 基础模型调参

通过交叉验证以及网格搜索调参后，基础模型调参结果见附录文件。

5.1.2AutoGluon 调参

由于 AutoGluon 会自动训练基础模型并进行集成，我们在用早期风险表现数据对模型进行训练和预测时并不需要手动调参或者使用网格搜索来指定超参数（当然指定超参数这一步是可以在 AutoGluon 中实现的）。在实际训练中，我们只需要指定 fit()函数所需的参数即可。具体为 auto_stack=True，这一步会让模型自动对基础模型进行 stacking 和 bagging 的操作，和 presets='best_quality'，AutoGluon 会自动输出在验证集上达到最好效果的模型。

最后传入得模型 num_stack_level=0, num_bag_folds=8, num_bag_sets=20，经过每个基础模型 time_limit=300 的训练，模型自动选择 catboost 模型来进行验证，在本地验证集上得到了较好的实验结果。由于参数是模型自行选择的，我们推测增加 num_stack_level 可以提升模型在本地训练集上的表现。当指定 num_stack_level 分别为 1，2，3 时，模型在本地验证集上的得分都出现了下降，且随着 num_stack_levle 增大，得分下降更多。造成这种结果的原因可能是数据集较小，模型在拟合的过程中出现了过拟合，从而造成了得分的下降。

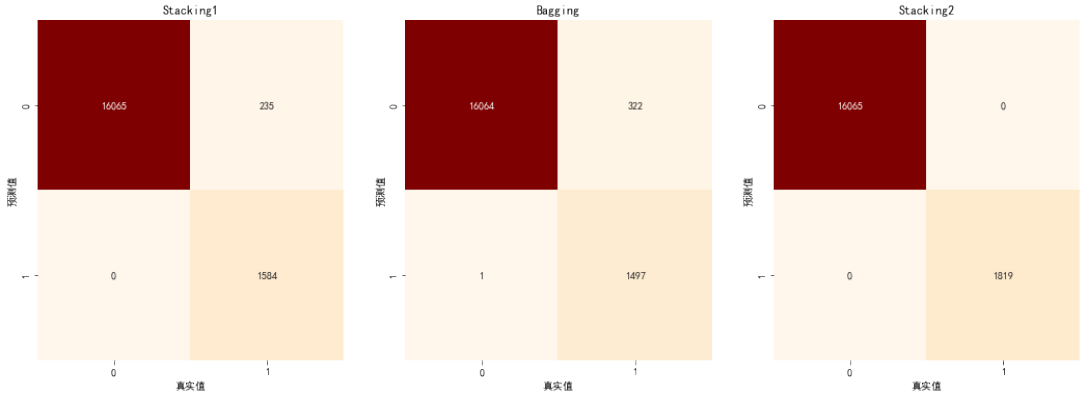
5.2 三个不同质子模型评价与对比分析

5.2.1 评分比较

通过对三个子集成模型进行训练，我们在本地的训练集上得到了以下的结果。

Stacking 模型一的 F1 为:1.00000,AUC 为:1.00000,召回率为:1.00000，精确率为：1.00000。Stacking 模型二的 F1 为：0.96181，AUC 为：0.93540，召回率为：0.87081，精确率为：1.0。AutoGluon 的 F1 为：0.94633，AUC 为：0.91146，召回率为：0.82298，精确率为：0.99933。

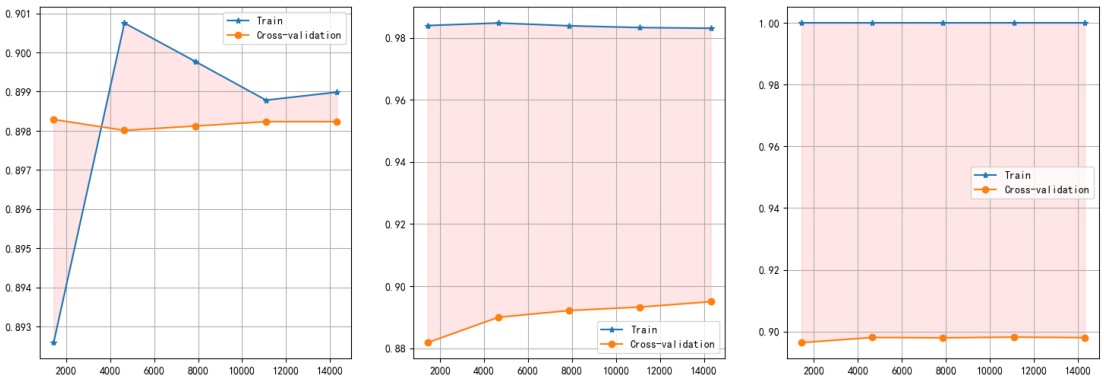
混淆矩阵如下图十所示：



图十

虽然 Staking 模型一的本地表现优于另外两个模型，然而其公榜得分较另两个模型更低，下一步我们针对模型公榜得分进行加权投票融合。

5.2.2 学习曲线比较



图十一

由上图十一可知，Staking 模型一的稳定性优于另外两个模型，AutoGluon 模型稳定性次之，Staking 模型二的稳定性最低。

5.3 模型最终得分评价

在模型得分中，由本文集成模型预测的结果在公榜中取得了 0.7866，排名第四，私榜得分 0.7865，排名第二。为了评价模型的稳健性，结合公榜与私榜分数的变化，我们构建了以下的指标：

$$\Delta AUC = |AUC_{public} - AUC_{private}|$$

其中AUC_{public}是模型公榜得分，$AUC_{private}$是模型私榜得分。ΔAUC变动越小表示模型的稳定性越强。我们对前二十名的分数变化进行了统计得到以下统计图：

Public	AUC_{public}	$AUC_{private}$	ΔAUC
1	0.7924	0.7743	0.0181
2	0.7922	0.7881	0.0041
3	0.7867	0.7862	0.0005
4	**0.7866**	**0.7865**	**1E-04**
5	0.7856	0.7812	0.0044
6	0.7852	0.783	0.0022
7	0.7848	0.7703	0.0145

8	0.7841	0.7846	0.0005
9	0.7836	0.7746	0.009
10	0.7825	0.7816	0.0009
11	0.7823	0.7857	0.0034
12	0.7812	0.7674	0.0138
13	0.7806	0.7764	0.0042
14	0.7806	0.7774	0.0032
15	0.7804	0.7739	0.0065
16	0.78	0.7763	0.0037
17	0.7793	0.7745	0.0048
18	0.7791	0.7841	0.005
19	0.7788	0.7756	0.0032
20	0.7787	0.7828	0.0041
MIN			**0.0001**
MAX			**0.0181**
AVE			**0.00531**

由上述表格可知，模型得分波动最大达到了 0.0181，平均模型得分波动为 0.00531，本文提出的集成模型得分 0.0001，波动远远低于平均水平，体现了集成模型强大的稳健性。

5.4 集合模型优势分析

与纠错码原理类似，集成模型通过遵循少数服从多数原则能够减少错误率，它能在模型间相关度较低时能取得更好的结果。首先，由于模型自身的学习特点，不同模型在学习时可能会犯不同的错误，而且这些错误多是在局部发生的。如果使用多个模型来进行投票，出错的可能性就更小了，相对而言，模型预测的正确率就会得到提高。其次，由于相关性低的模型犯的错误的类型更加不同，集成低相关性的模型结果会增加模型的正确率。通常我们希望模型越好，其权重就越高。当表现较差得模型需要否决表现最好的模型时，唯一的办法时它们集体同意另一种选择。因此我们将表现最好的模型投票赋以更多的权重，我们期望这样的集成能够对表现最好的模型进行一些修正，带

来一些小的提高。

5.5 模型拓展

5.5.1 增加子模型个数

跟重复码相似，随着编码重复次数的增加对错误的校正能力也会增加，因此集成通常可以通过提高集成成员的个数来提升准确率。当子模型个数进一步上升时，每个单独模型的方差应该有所降低，最终的 AUC 排名评分应该提升。因为不同结构模型之间进行学习时犯的错不一定会重叠，所以子模型之间需要结构差异化（类似的模型容易犯类似的错），而且模型本身性能的要好，类比神经网络，特征提取网络一定要强，这样才能产生好的效果。总的来说，我们在实际预测中可以通过增加模型性能强的子模型，来学习器减少错误学习产生的影响，从而产生更好的效果。

5.5.2 加权平均法修正

虽然在比赛中，我们能够根据公榜得分，对结果进行加权平均，以得到更高的分数。但我们这种方法在私榜未知时，也只能确定这种方法会提高模型的稳定性。和在实际过程中一样，各个子模型在测试集效果是未知的。对于金融机构来讲，实际情况是复杂且不可预知的。为了提高风控模型的稳定性，在实际运用过程中，金融可以采取简单的平均法，也可以提升模型的稳定性，为模型带来性能提升。

5.5.3 模型自动化

构建成的集成模型分为三层，集成模型复杂繁重，若能将上述集成学习器进一步 stacking 的过程工程化，以供学者或业界研究员直接调用，自动将不同质学习器进行集成，形成更稳定的模型，将大大降低训练成本。

参考文献

[1]Stiglitz, J. E., & Weiss, A. (1981). Credit rationing in markets with imperfect information. *The American economic review*, *71*(3), 393-410.

[2]Baptista, J. A., Ramalho, J. J., & Vidigal da Silva, J. (2006). Understanding the microenterprise sector to design a tailor-made microfinance policy for Cape Verde. *Portuguese Economic Journal*, *5*(3), 225-241.

[3]Kolari, J., Glennon, D., Shin, H., & Caputo, M. (2002). Predicting large US commercial bank failures. *Journal of Economics and Business*, *54*(4), 361-387.

[4]Q Wang, Cox S. (2014). Risk Management in U.S. Bank. *Journal of Banking in Finance*, 11(3):47-54.

[5]Odom M D, Shara R A. (1990). Neural Network Model for Bankruptcy. Prediction Proceedings of the IEEE International Joint-Conferenceon Neural Networks, (2):163-168.

[6]Shen, K. Y., & Tzeng, G. H. (2015). A decision rule-based soft computing model for supporting financial performance improvement of the banking industry. *Soft Computing*, *19*(4), 859-874.

[7]王利军. (2009). 商业银行小企业信贷风险的管理与控制. 内蒙古财经大学学报(2), 97-99.

[8]梁彩红. (2014). 论商业银行小微企业信贷风险管理. 上海金融(9), 3.

[9]苏蕙, & 郭炜. (2020). 银行小微企业信贷风险评价指标优化. 财会月刊(1), 6.

[10]肖宇辰, & 余舜基. (2020). 基于供应链金融模式下的中小企业信用风险评估. 中国商论(20), 2.

[11]邵黎、张仿龙、陈俊、张林欢. (2020). 基于商业银行视角的大数据信贷风控研究. 国际金融(11), 6.

附录

附录 1

介绍： 支撑材料的文件列表

1. 寻找baseline+网络调参.ipynb：寻找baseline+网络调参代码；
2. Stacking_模型二.ipynb：特征工程、stacking2；
3. Stacking_模型一.ipynb：stacking模型一；
4. AutoGluon.ipynb；Autogluon；
5. Voting.ipynb:标准化加权投票；

附录 2

介绍： 包的调用

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from xgboost import XGBClassifier
from lightgbm import LGBMClassifier
from catboost import CatBoostClassifier
from sklearn.ensemble import
BaggingClassifier,RandomForestClassifier,ExtraTreesClassifier,GradientBoostingClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split,GridSearchCV
from sklearn.ensemble import StackingClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline
from imblearn.over_sampling import SMOTE
from sklearn.metrics import
confusion_matrix,f1_score,recall_score,auc,classification_report,roc_auc_score,precision_score
from sklearn.model_selection import learning_curve
```

附录 3

介绍： 交叉验证

```
'''交叉验证寻找基模型'''
def find_base(data,model_list:list,cv=3):
    x_train = data.drop('label',axis=1)
    y_train = data.label

    result = list(map(lambda x:cross_val_score(x,x_train,y_train,cv=cv),model_list))
    result = dict(zip(model_list,result))

    return result
```

附录 4

介绍： 网络搜索调参

```
'''网格'''
def grid_search(x_train,y_train,model:BaseEstimator,params,cv=5):
    estimator = GridSearchCV(model,params,cv=cv)
```

```
    estimator.fit(x_train,y_train)

    return estimator
```

附录 5

介绍： 特征工程

```
'''特征工程'''
def process_data(path,train=True):
    data = pd.read_csv(path)

    '''去掉一列都为相同的列'''
    data.drop(columns=['basic_9','loan1_33','query_1'],inplace=True)

    '''-99太多的列'''
    null_many_columns = []
    for column in data.columns:
        if dict(data[column].value_counts()).get(-99,0) / len(data) > 0.5:
            null_many_columns.append(column)
    print('-99多于50%的列：{}'.format(null_many_columns))

    '''去掉-99太多的列'''
#       data.drop(columns=null_many_columns,inplace=True)


    '''
    1.填充缺失值
    2.对填充了缺失值的列进行标注
    '''
    for column in data.columns:
        if column in ['label','judicial','province','industry','scope','cust_id']:
            continue
        temp = data[column]
        temp.drop(np.where(temp==-99)[0],inplace=True)
        if column in ['loan2_2','loan2_4,','loan2_5','loan2_11','loan2_23','loan2_24']:
            data[column+'_sign'] = data[column].apply(lambda x:1 if x==-99 else 0)
            data[column].replace(-99,temp.mean(),inplace=True)

        elif column in
['basic_8','loan1_8','loan1_9','loan1_23','loan1_26','loan2_3','loan2_6','loan2_7',\

'loan2_8','loan2_10','loan2_12','loan2_15','loan2_16','loan2_17','loan2_18','loan2_19',\

'loan2_22','loan2_25','loan2_26','loan2_27','loan2_28','overdue_7','overdue_8']:
            data[column+'_sign'] = data[column].apply(lambda x:1 if x==-99 else 0)
            data[column].replace(-99,temp.mode().values[0],inplace=True)

        else:
            data[column+'_sign'] = data[column].apply(lambda x:1 if x==-99 else 0)
            data[column].replace(-99,temp.mode().values[0],inplace=True)
```

```
    data['judicial'].fillna(0,inplace=True)

    '''对可能是年龄的列分箱'''
    data['basic_2'] = pd.cut(data['basic_2'],[0,18,30,40,50,65,100],labels=[0,1,2,3,4,5])
    data['basic_2'] = data['basic_2'].astype('int32')

    '''值全为相同的列'''
    all_same_columns = []
    for column in data.columns:
        if len(data[column].unique()) == 1:
            all_same_columns.append(column)
    print('值全为相同的列：{}'.format(all_same_columns))

#                 data.drop(columns=column,inplace=True)
#     data.drop(columns=['basic_8','loan1_31','loan1_32','overdue_5','overdue_9',
#
'query_2','cust_id_sign','basic_1_sign','basic_2_sign','basic_5_sign','basic_6_sign'],
#                                   inplace=True)

    '''SMOTE采样'''
#     if train:
#         transfer = SMOTE(sampling_strategy={1:2000,0:16065},random_state=22)
#         f,l = transfer.fit_sample(data.drop('label',axis=1),data.label)
#         data = f
#         data['label'] = l
    '''加入woe编码并将区间转化为分箱'''
#     if train:
#         woe = pd.read_csv('clean_train_woe.csv')
#     else:
#         woe = pd.read_csv('clean_test_woe.csv')
#     data = pd.concat([data,woe],axis=1)

#     data['basic_4_woe'].replace(['[1.0, 10.5)','[10.5, 26.5)','[26.5, 33.5)','[33.5, 38.5)','[38.5,
85.5)','[85.5, 1481.1)'],[0,1,2,3,4,5],inplace=True)
#     data['basic_7_woe'].replace(['[1.0, 2.5)','[2.5, 8.5)','[8.5, 11.5)','[11.5, 14.5)','[14.5,
20.5)','[20.5, 213.1)'],[0,1,2,3,4,5],inplace=True)
#     data['loan1_1_woe'].replace(['[0.0, 5.5)','[5.5, 7.5)','[7.5, 12.5)','[12.5, 23.5)','[23.5,
49.5)','[49.5, 1329.1)'],[0,1,2,3,4,5],inplace=True)
#     data['loan1_29_woe'].replace(['[0.0, 4.523)','[4.523, 7.886)','[7.886, 10.76)','[10.76,
12.321)','[12.321, 33.873)','[33.873, 239.1)'],[0,1,2,3,4,5],inplace=True)
#     data['loan2_11_woe'].replace(['[0.0, 0.000131)','[0.000131, 0.0745)','[0.0745,
0.36)','[0.36, 0.382)','[0.382, 0.723)','[0.723, 95.932)'],[0,1,2,3,4,5],inplace=True)
#     data['loan2_12_woe'].replace(['[0.0, 0.000379)','[0.000379, 0.851)','[0.851,
0.944)','[0.944, 1.071)','[1.071, 1.849)','[1.849, 6686.394)'],[0,1,2,3,4,5],inplace=True)

    '''构造新特征'''
    loan1 =
data[['loan1_2','loan1_3','loan1_4','loan1_7','loan1_13','loan1_14','loan1_17','loan1_20']]
    loan2 =
data[['loan2_20','loan2_13','loan2_23','loan2_24','loan2_14','loan2_26','loan2_21','loan2_25']]

    loan = pd.DataFrame({'loan1':loan1.sum(axis=1),'loan2':loan2.sum(axis=1)})
    loan['loan1'] = loan['loan1'].apply(lambda x:0 if x<.0 else x)
    loan['loan2'] = loan['loan2'].apply(lambda x:0 if x<.0 else x)
```

```
    data[['loan1','loan2']] = loan
    data.fillna(0,inplace=True)

    return data
```

附录 6

介绍： 网络搜索调参

```
'''网格'''
def grid_search(x_train,y_train,model:BaseEstimator,params,cv=5):
    estimator = GridSearchCV(model,params,cv=cv)
    estimator.fit(x_train,y_train)

    return estimator
```

附录 7

介绍： voting

```
result_1=pd.read_csv("result1.csv")
result_2=pd.read_csv("result2.csv")
result_3=pd.read_csv("result3.csv")
def MaxMinNormalization(x):
    """[0,1] normaliaztion"""
    x = (x - np.min(x)) / (np.max(x) - np.min(x))
    return x
result_1["label"]=MaxMinNormalization(result_1["label"])
result_2["label"]=MaxMinNormalization(result_2["label"])
result_3["label"]=MaxMinNormalization(result_3["label"])
result_4=result_1
result_4["label"]=0.33*result_1["label"]+0.33*result_2["label"]+0.34*result_3["label"]
result_4.to_csv('result14.csv',index=False,encoding='utf-8')
```

附录 8

介绍： AutoGluon

```
import autogluon
import bokeh
import pandas as pd
# 载入包
from autogluon.tabular import TabularDataset, TabularPredictor
# 载入训练数据
train_data = TabularDataset(r"df_train_a1.csv")
# 建模
predictor =
TabularPredictor(label='label').fit(train_data.drop(['cust_id'],axis=1),auto_stack=True,time_limit
=120, presets='best_quality')
# (note: specifying presets='best_quality' in fit() simply sets auto_stack=True).
# 载入测试数据
test_data = TabularDataset(r"df_test_a1.csv")
results = predictor.fit_summary(show_plot=True)
```

```
leaderboard = predictor.leaderboard(train_data.iloc[:,1:],silent=True)
pred_probs = predictor.predict_proba(test_data)
pred_probs.insert(0,'cust_id',value=test_data['cust_id'])
pred_probs.drop([0],axis=1,inplace=True)
pred_probs.columns=['cust_id','label']
pred_probs
pred_probs.to_csv('result.csv',index=False,encoding='utf-8')
```

附录 9

介绍： stacking model 1

```
import numpy as np
import pandas as pd
import xgboost as xgb
import lightgbm as lgb
import catboost as ctb
from sklearn.model_selection import KFold

from xgboost import XGBClassifier
from lightgbm import LGBMClassifier
from catboost import CatBoostClassifier
from sklearn.ensemble import ExtraTreesClassifier
train_data_X =pd.read_csv('train.csv')
test_data = pd.read_csv("test.csv")
train_data_y = train_data_X['label']
train_data_X = train_data_X.drop(['cust_id','label'],axis=1)
oof_lgb = np.zeros(train_data_X.shape[0])
oof_xgb = np.zeros(train_data_X.shape[0])
oof_cat = np.zeros(train_data_X.shape[0])
oof_rft = np.zeros(train_data_X.shape[0])
oof_ext = np.zeros(train_data_X.shape[0])

test_output_df = pd.DataFrame(columns=['lgb','xgb','cat'],index=range(test_data.shape[0]))
test_output_df = test_output_df.fillna(0)
kfold = KFold(random_state=800,shuffle=True)
lgb_reg = lgb.LGBMRegressor(max_depth=8,
                            n_estimators=3000,
                            subsample=0.8,
                            colsample_bytree=0.8,
                            random_state=32)

cat_reg = ctb.CatBoostRegressor(learning_rate=0.1,
                                depth=8,
                                random_seed=32,
                                )

xgb_reg = xgb.XGBRegressor(max_depth=8,
                           learning_rate=0.1,
                           n_estimators=3000,
                           n_jobs=4,
                           colsample_bytree=0.8,
                           subsample=0.8,
                           random_state=32,
                           )
```

```
from sklearn.ensemble import
RandomForestRegressor,ExtraTreesRegressor,GradientBoostingRegressor

ext_reg =    ExtraTreesRegressor(max_depth=8,
                                 min_samples_split=100,
                                    min_samples_leaf=20,
                                    random_state=32,)
rft_reg=RandomForestRegressor(max_depth=8,
                                 min_samples_split=100,
                                    min_samples_leaf=20,
                                    random_state=32,)

for train_idx, valid_idx in kfold.split(train_data_X):
      train_x = train_data_X.loc[train_idx]
      train_y = train_data_y.loc[train_idx]
      valid_x = train_data_X.loc[valid_idx]
      valid_y = train_data_y.loc[valid_idx]
      lgb_reg.fit(train_x,train_y,
eval_set=[(train_x,train_y),(valid_x,valid_y)],early_stopping_rounds=30)
      xgb_reg.fit(train_x,train_y,
eval_set=[(train_x,train_y),(valid_x,valid_y)],early_stopping_rounds=30)
      cat_reg.fit(train_x,train_y,
eval_set=[(train_x,train_y),(valid_x,valid_y)],early_stopping_rounds=30)
      oof_lgb[valid_idx] = lgb_reg.predict(valid_x)
      oof_xgb[valid_idx] = xgb_reg.predict(valid_x)
      oof_cat[valid_idx] = cat_reg.predict(valid_x)
      test_output_df['xgb'] += xgb_reg.predict(test_data.drop('cust_id',axis=1))
      test_output_df['lgb'] += lgb_reg.predict(test_data.drop('cust_id',axis=1))
      test_output_df['cat'] += cat_reg.predict(test_data.drop('cust_id',axis=1))

test_output_df['lgb'] = test_output_df['lgb'] / 5
test_output_df['xgb'] = test_output_df['xgb'] / 5
test_output_df['cat'] = test_output_df['cat'] / 5
oof_df = pd.DataFrame({'xgb':oof_xgb,'lgb':oof_lgb,'cat':oof_cat})
from sklearn.linear_model import LogisticRegression

log_r = LogisticRegression()
log_r.fit(oof_df, train_data_y)
final_pre1 = log_r.predict(test_output_df)
from sklearn.linear_model import LinearRegression


lrg = LinearRegression()
lrg.fit(oof_df, train_data_y)
final_pre2 = lrg.predict(test_output_df)
final_pre3 = np.where(final_pre2>=0.5, 1, 0)
#final_pre2.to_csv('x.csv')
submission=pd.DataFrame(final_pre2)
submission.to_csv('result.csv',index=False)
```

附录 10

介绍： stacking model 2

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from xgboost import XGBClassifier
from lightgbm import LGBMClassifier
from catboost import CatBoostClassifier
from sklearn.ensemble import
BaggingClassifier,RandomForestClassifier,ExtraTreesClassifier,GradientBoostingClassifier
from sklearn.linear_model import LogisticRegression

from sklearn.model_selection import train_test_split,GridSearchCV
from sklearn.ensemble import StackingClassifier
from sklearn.ensemble import RandomForestClassifier


from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline

from imblearn.over_sampling import SMOTE

from sklearn.metrics import
confusion_matrix,f1_score,recall_score,auc,classification_report,roc_auc_score,precision_score
from sklearn.model_selection import learning_curve
data = process_data('df_train_a1.csv')
test = process_data('df_test_a1.csv',train=False)
estimators = [('cat',CatBoostClassifier(random_seed=22)),
                    ('extra',ExtraTreesClassifier(random_state=22)),
                    ('lgm',LGBMClassifier(learning_rate=0.01, max_depth=5, n_estimators=900,
num_leaves=63,random_state=22)),
                      # learning_rate=0.01, max_depth=5, n_estimators=900, num_leaves=63)
                     ]

stacking = StackingClassifier(estimators)
stacking.fit(data.iloc[:,2:],data.label)
y_prob = model.predict_proba(test.iloc[:,1:])
result = pd.DataFrame(y_prob[:,1],columns=['label'])
result.insert(0,'cust_id',value=test['cust_id'])
result.to_csv('result.csv',index=False,encoding='utf-8')
```

经世济民

敦敦以求

西南财经大学

SOUTHWESTERN UNIVERSITY OF FINANCE AND ECONOMICS

校址：四川成都温江柳台大道555号
电话：028-87092032　传真：028-87092632
邮编：611130
网址：http://www.swufe.edu.cn
Address: Liulin Campus (Main Campus): 555, Liutai Avenue,
Wenjiang District, Chengdu, Sichuan, P. R. China
Tel: 86-28-87092032　Fax: 86-28-87092632　Postcode: 611130