微额借款用户人品预测大赛 解决方案

1. 解决方案概述

微额借款用户人品预测的数据不仅有带标签数据(1.5W)，而且还有大量无标签数据(5W)两种类型，数据存在缺失值，类别不平衡，特征高维度等特点。本次大赛主要是为了能从用户行为数据分析‘小额微贷’申请借款用户的信用状况，来判断其是否逾期。针对需要解决的问题和数据特征，我们主要从四个方面进行处理：数据预处理，特征工程，数据不平衡，以及半监督模型训练。

首先，由于数据中存在大量的缺失值，因此需要对缺失值数据进行预处理。比赛中，我们视缺失值为一种特征进行处理。统计了每个样本的缺失值个数，并进行排序，发现样本类别和缺失值个数呈规律性阶梯状关系。通过分析发现，当训练数据中缺失值个数大于某个阈值后，这部分样本会产生噪声数据，并产生过拟合现象，因此把这部分样本进行剔除。其次，生成排序特征，离散特征，计数特征，并采用基于学习模型的特征排序方法做了特征选择。在训练模型时，对模型参数加入了小范围的随机扰动，从而得到多个具有一定差异性的模型，以便对模型进行融合。然后，针对类别不平衡问题，不仅采用了传统的代价敏感学习方法，而且通过半监督方法利用无标签数据进行学习。最后，我们提出了一种暴力半监督模型，该模型相当于是一个预处理过程。与传统半监督模型不同，该模型不考虑具体打标的准确性，只考虑能否利用无标签样本的分布来改善分类器性能，其可以替换为任何分类器，更换任意评估指标，都起到了提高相应性能的作用。

通过以上处理，我们得到了单模型，多模型融合，迭代半监督，以及暴力半监督等四个模型，通过实验测试证明，暴力半监督模型更能有效的预测人品，并在竞赛中取得了线上得分 0.7341 的成绩。

2. 数据预处理

主要是缺失值的处理。赛题数据中大部分样本都有缺失值，且缺失值个数较多，有的样本甚至有上千个缺失值。常用的缺失值处理方法是缺失值填充（用同类别数据的特征均值，中值等），但由于测试集也同样有大量的缺失值，因此没有采用该方法。既然训练集和测试集都有这种特点，不如就把缺失值当成一种特征来处理。

我们统计了训练集中每个样本的缺失值个数，并且按照缺失值个数从小到大排序，以序号为横坐标，缺失值个数为纵坐标，画出如下图 1：

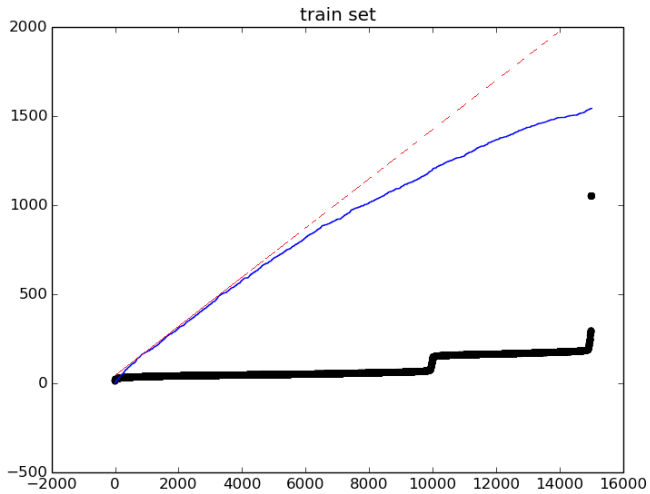

图 1

黑色曲线是缺失值散点图。蓝色的曲线是累计的负样本个数，可以发现它并不是一条严格的直线，说明样本类别与缺失值个数具有一定相关性。

我们进一步地对测试集和无标签数据集进行缺失值统计，得到下图：

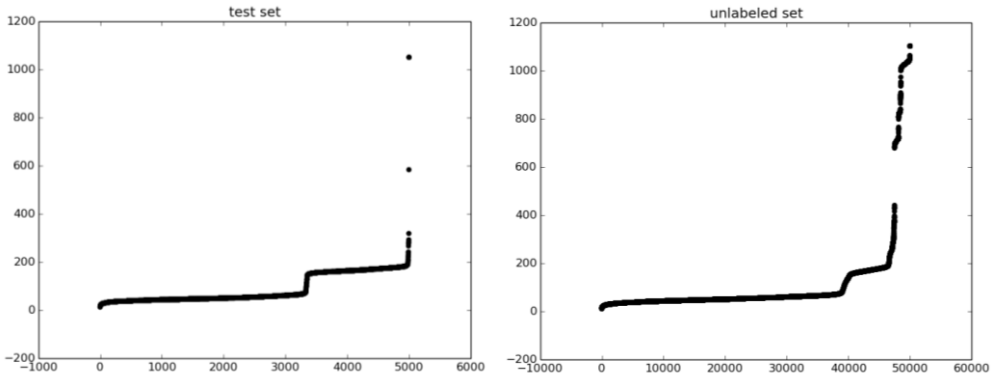

图 2

由此可知，图 1 和图 2 都呈现出了规律性的阶梯状，尤其是训练集和测试集的缺失值个数分布非常一致。所以进一步地将缺失值离散为 5 个区间，得到一个表征缺失值个数的离散特征（取值 1~5），如下图 3 所示：

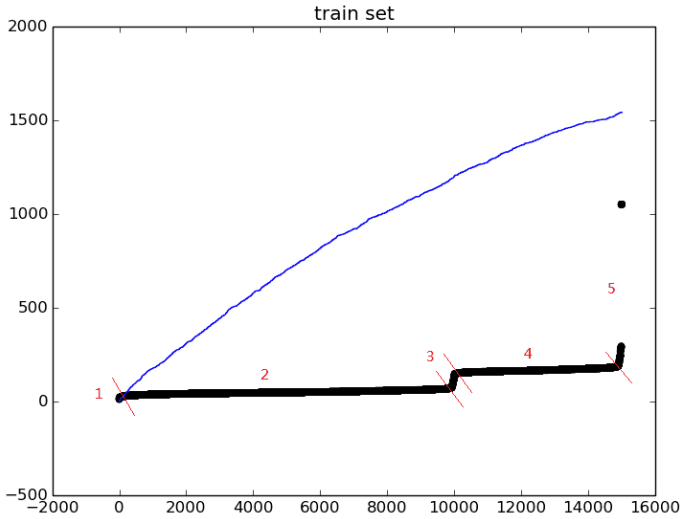

图 3

我们为什么要这么做呢？起初分析画出这条曲线时，如获珍宝，猜测这种阶梯状可能是由不同的用户群数据导致的，比如说上面的阶梯对应着男性用户，下面的阶梯对应女性用户；或者上面的阶梯对应着老用户，下面的阶梯对应新用户等等。总之，我们脑洞大开，不管这种阶梯状是什么原因造成的，我们先把它作为一项特征，让模型自己去学习。

进一步地，我们删除了训练数据中缺失值个数大于 194 个的数据（即缺失值个数的离散值为 5 对应的样本），如下图 4 所示。这部分样本包含了太多的缺失值，使模型的学习变得困难，甚至会引入噪声，造成过拟合，所以去掉这部分数据后，线上的 auc 提高了 0.002，这是一个不小的提升。

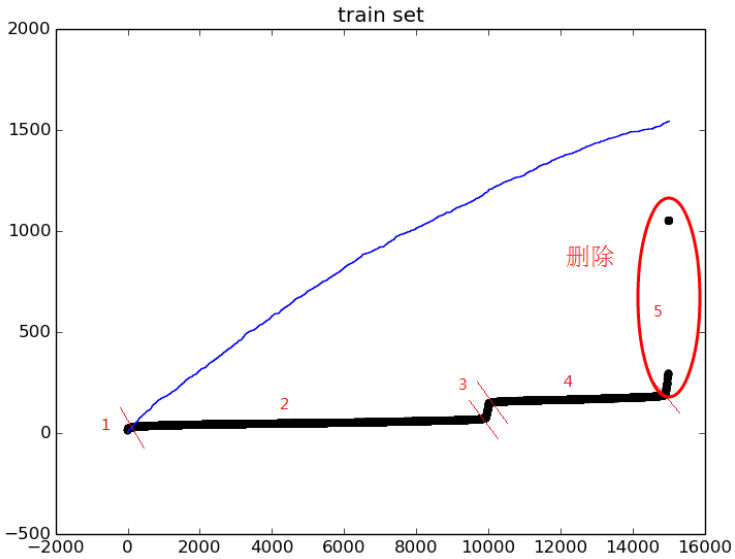

图 4

3. 特征工程

坊间戏言“特征没做好，参数调到老”，机器学习大牛 Andrew Ng 也说过“‘Applied machine learning’is basically feature engineering”，可见特征工程的重要性，我们在这部分投入了大量的时间和精力。

3.1 排序特征

对原始特征中 1045 维 numeric 类型的特征从小到大进行排序，得到 1045 维排序特征。排序特征对异常数据都有较强的鲁棒性，使得模型更加稳定，降低过拟合的风险。

3.2 离散特征

特征离散化有两种划分方式：一种是等值划分（按照值域均分），另一种是等量划分（按照样本数均分）。我们对 numeric 类型的特征采用了等量划分的离散化方式：先将每一维特征按照数值大小排序，然后均匀地划分为 10 个区间，即离散化为 1~10。

3.3 计数特征

前面已经对特征进行了离散化，以 uid 为 1 的样本为例，离散化后它的特征是 5,3,1,3,3,3,2,4,3,2,5,3,2,3,2...2,2,2,2,2,2,2，可以进一步统计离散特征中 1~10 出现的次数 $n_i(i=1,2,...,10)$，即可得到一个 10 维计数特征。基于这 10 维特征训练了 xgboost 分类器，线上得分是 0.58 左右，说明这 10 维特征具有不错的判别性。

3.4 类别特征编码

赛题数据含有 93 维类别特征，很多算法（如逻辑回归，SVM）只能处理数值型特征，这种情况下需要对类别特征进行编码，我们采用了 One-Hot 编码，得到了 01 特征，解决了分类器不能处理类别特征的问题。

3.5 交叉特征

交叉特征是我们团队前期的想法之一，但是在实现过程中，发现即使两两交叉也会生成 50W 个特征（因为不明特征含义，只能交叉全部数值特征），后来我们采用在生成特征的同时，对特征进行评估，保留生成特征的 top k，但是时间复杂度过高，就放弃这个想法了，后来实验室另一个队实现了该想法，线上也有明显提升。

4. 特征选择

前面我们头脑风暴，基于原始特征生成了排序特征，离散特征，以及一些计数特征。全

部加起来有 3000 多维，这么多维特征一方面可能会导致维数灾难，另一方面很容易导致过拟合，因此需要做降维处理，常见的降维方法有 PCA，t-SNE（计算复杂度很高）。我们尝试了 PCA，效果并不好，猜测原因是大多数特征含有缺失值且缺失值个数太多，而 PCA 前提假设数据呈高斯分布，赛题数据很可能不满足。

除了采用降维算法之外，也可以用特征选择来降低特征维度。特征选择的方法很多：最大信息系数（MIC）、皮尔森相关系数（衡量变量间的线性相关性）、正则化方法（L1，L2）、基于模型的特征排序方法。比较高效的是最后一种方法，即基于学习模型的特征排序方法，这种方法有一个好处：模型学习的过程和特征选择的过程是同时进行的，因此我们采用这种方法。

基于决策树的算法（如 random forest，boosted tree）在模型训练完成后可以输出特征的重要性，在这个比赛中我们用了 xgboost 来做特征选择，xgboost 是 boosted tree 的一种实现，效率和精度都很高，在各类数据挖掘竞赛中被广泛使用。

5. 模型设计与分析

比赛是一个不断尝试和思考的过程，数据处理、特征工程、特征选择、模型训练也是一个不断迭代更新的过程，如下表 1 记录了我们是怎么一步一步做到现在的成绩的，后面我们将对每个模型进行详细的说明。

表 1

模型	方法	版本	线上得分
M1	Xgboost	Py_717	0.717
	Xgboost	R_7199	0.7199
	Xgboost	Java_7218	0.7218
	SVM	SVM_6938	0.6938
M2	Bagging of Xgboost（feature selected）	Py_725	0.725
M3	0.15*Py_717+0.25*SVM_69+0.6*Py_725	M_7273	0.7273
	0.15*R_7199+0.2*SVM_69+0.65*Py_725	M_7275	0.7275
	0.8* M_7275+0.2* Java_7218	M_7279	0.7279
M4	Iteration(M_7279(unlabeled))	I_M73	0.73
M5	Lb_rank(10_label(Py_717))	Final_Model	0.7341

5.1 模型 M1

如下图 5 所示，这部分包括 4 个单模型。

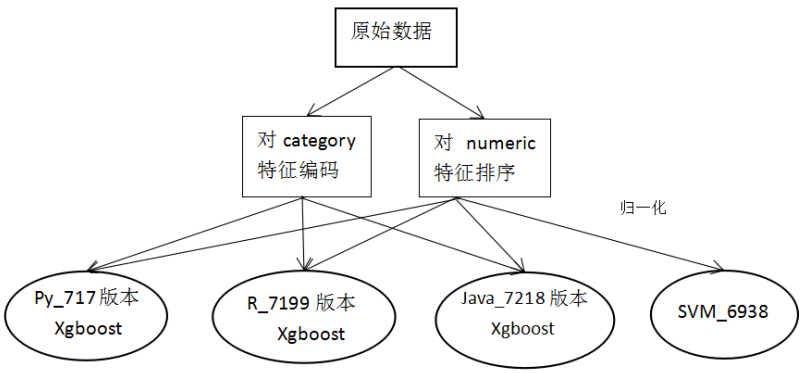

图 5

- xgboost.

使用原始数据并且将 category 特征进行编码，转换成哑变量特征（01 特征），numeric 特征不变。基于此训练得到了 3 个 xgboost（不同队员做的，参数具有差异性，从而有利于后面的模型融合）：

Python 版本的 xgboost 线上得分 0.717 (代码为分享在论坛的 top10 代码)

R 版本的 xgboost 线上得分 0.7199

Java 版本的 xgboost 线上得分 0.7218

- SVM.

使用排序特征，训练前对特征做归一化，线上得分 0.6938。

5.2 模型 M2

在基于原始特征得到排序特征、离散特征、计数特征之后，我们分别对这几种特征进行特征选择，特征选择的方法前面已有介绍，基于 xgboost 来做，训练 xgboost 的过程就是对特征重要性进行排序的过程。

得到特征的重要性之后，我们可以保留最重要的 top N1 个原始特征，top N2 个排序特征，top N3 个离散特征。（计数特征由于只有 10 维，所以不做特征选择）。

参数（N1,N2,N3)可以通过实验来确定最优的取值。但是这样很浪费时间，所以我们采用了 bagging 的思想，训练了 36 个 xgboost 模型再求其平均，每个 xgboost 的 N1 取值为 random(300,500)，即 N1 的取值是在 300~500 里随机取的。N2 的取值为 random(300,500)，N3 的取值为 random（64,100）。

此外，在训练 36 个 xgboost 时，xgboost 的参数也是在一定数值范围里随机取值的。这样一来，模型 M2 不仅仅在特征这个角度引入了多样性，也在模型参数这个角度引入了多样性，从而使得 bagging 的效果非常好。

如下图 6 是模型 M2 的流程图：

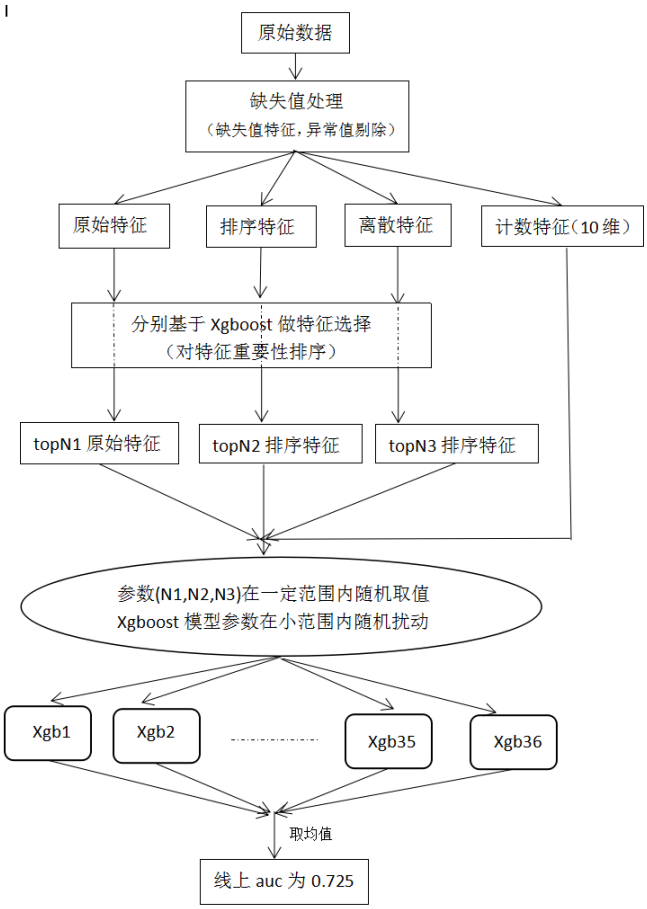

图 6

5.3 模型 M3

模型融合，方式为简单加权融合。

模型融合要取得较好的结果，要求单模型具有多样性（即差异性）。为了直观地观察单模型之间的差异性，我们计算了它们两两之间的最大信息系数（MIC），以混淆矩阵的形式画出（颜色越浅，表示相关性越小），如下图 7 所示。

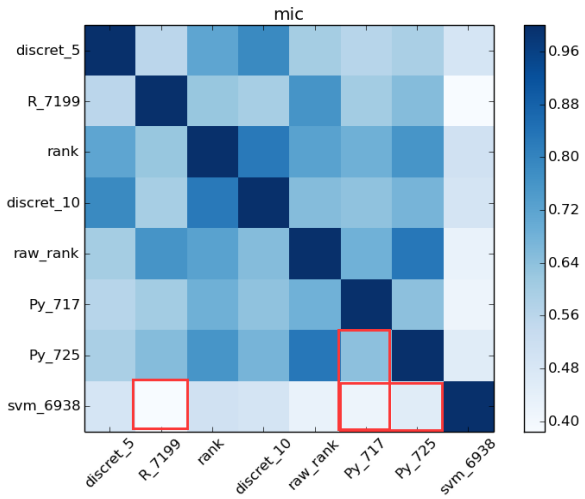

图 7

由图可知，红色框出的部分是颜色较浅的，即 Py_725、SVM_69、Py_717、R_7199 这几个模型之间差异性较大，所以我们使用这 4 个模型进行加权融合,以下两种是我们尝试过的

较为不错的融合方案：

0.15*Py_717+0.25*SVM_6938+0.6*Py_725　线上得分 0.7273

0.15*R_7199+0.2*SVM_6938+0.65*Py_725　线上得分 0.7275

得到 0.7275 的结果后，进一步地与单模型结果 Java_7218 进行加权融合：

0.8* M_7275+0.2* Java_7218　线上得分 0.7279

5.4 模型 M4

迭代半监督，利用最好的模型预测无标签数据，调整阈值 a，b 将样本添加到训练集，其中 01 样本比例为 5：1~9：1。如果线上有提高，将这部分样本添加到训练集继续训练融合来预测无标签数据，重复迭代。线上得分 0.73。

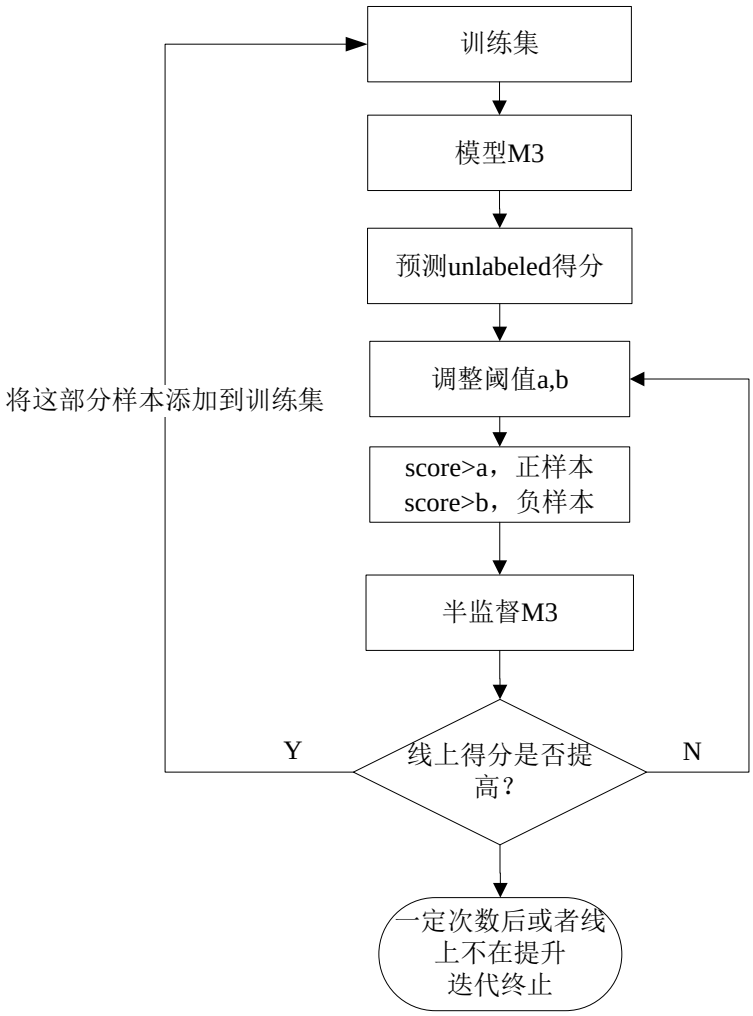

图 8

5.5 模型 M5

暴力半监督，每次从无标签样本中选择 10 个样本，有 1024 种打标签方式。使用单模型 Py_717（多模型融合复杂度太高）训练 1024 次并在测试集上测试，选择线下（train:test 为 1:9 的比例）性能提升最多的那组标签。

将 5000 组*10 的数据取 top500 组*10。在这 5000 个样本中，每次选择部分样本（20~50）添加到训练集观测线上表现，保留提分的样本。最终将这部分样本添加到训练集中，运行 M3 得到 Final_Model，线上得分 0.7341。

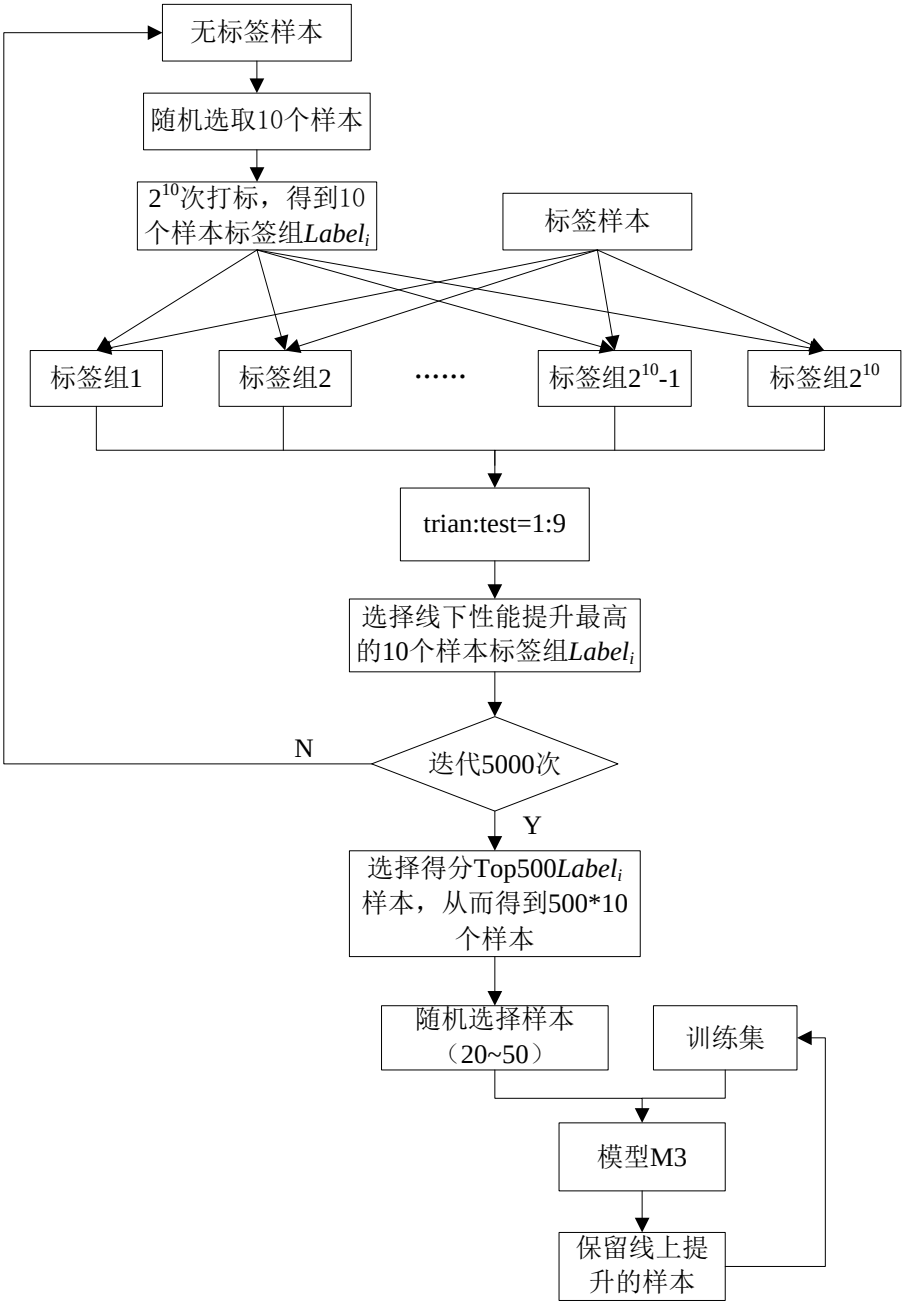

图 9

5.6 模型性能分析

（1）单模型，xgboost，全部训练集 1000+维特征 8 线程 2 分钟内可以运行完成。

（2）多模型融合，时间为模型数目*2 分钟。

（3）迭代半监督，时间为多模型融合的时间，依赖线上反馈结果。

（4）暴力半监督，训练集测试集 1：9 的比例训练和测试一次需要 1s 。10 个样本 1024 次迭代为一个回合需要 20 分钟左右，5W 个无标签样本全部打标需要(50000/10)*20 分钟/20 线程=83 小时，4 天内能全部运行完成。剩下的时间为多模型融合时间，依赖线上反馈结果。

6. 创新点

1. 缺失值的处理。发现了训练数据和测试数据的样本缺失值个数的分布特性，呈现阶

梯状，并利用这种阶梯状对缺失值个数离散化为取值 1~5 的特征。进一步地，发现缺失值个数大于 194 的数据中包含太多噪声，删除后模型有较大提升。

2. 引入排序特征与离散特征，对异常数据更鲁棒。auc 本质是排序，需要从排序角度去优化算法。我们将特征全部排序，用排名作为特征，并进一步地离散化特征，使得模型更加稳定。

3. 设计了计数特征，对离散特征进一步统计计数特征 n1~n10，仅用这 10 维特征可得到线上 auc 值 0.58。

4. 差异性模型融合，在特征，算法，参数这几个维度都引入了多样性：随机选择 topN 特征，运用多种不同算法，模型参数加入小范围的随机扰动。这些措施都增加了模型之间的差异性，使得融合结果有很大提高。

5. 加权融合时，采用最大信息系数（MIC）度量不同模型的差异性，通过混淆矩阵可视化，方便选取有效的单模型组合。

6. 暴力半监督，虽然效率低，但是能有效提高线上得分，充分利用了无标签样本的分布信息来改善分类器性能。

7. 从 1 月 2 日第一次榜首开始，保持排行榜榜首将近 2 个月

附:代码详细说明