参赛队伍：irootech

验证集：0.8342

测试集：0.8260

队长:申将伟

所属单位：树根互联AI LAB

## 第一题 推断拒绝

### 1.数据集

**训练集**：30465个通过样本（tag=0）有标签，剩余66535个被拒绝的样本中随机抽样了3000个样本打上标签。

**验证集**（A榜）：20000个样本

**测试集**（B榜）：20000个样本（与验证集同源同分布）

**数据特点**：

1. 数据的特征没有明确含义，不利于做特征的构建有字段含义的特征不多：id、loan\_dt、tag（只有训练集有tag特征）

2.维度高，除了有字段含义的特征外，剩余的特征为f1~f6745

### 2.数据分析和处理

#### 2.1.数据探索

1. 数据集中的数据类型有三种，分别为int64、float64、string，其中int64类型的特征数量为494，float64类型的特征数量为6253，string类型特征为loan\_dt，总计6748个特征；

2. 数据的缺失的情况比较严重，我们取每一个样本的特征缺失总数作为新特征n\_null，并统计了n\_null在各个数量级的分布。观察到在0~2500的范围内n\_null呈正态分布，如图1所示。值得注意的是：训练集和测试集的分布比较吻合，于是猜测两个数据集的分布是相似的，但是仍需要进一步验证；

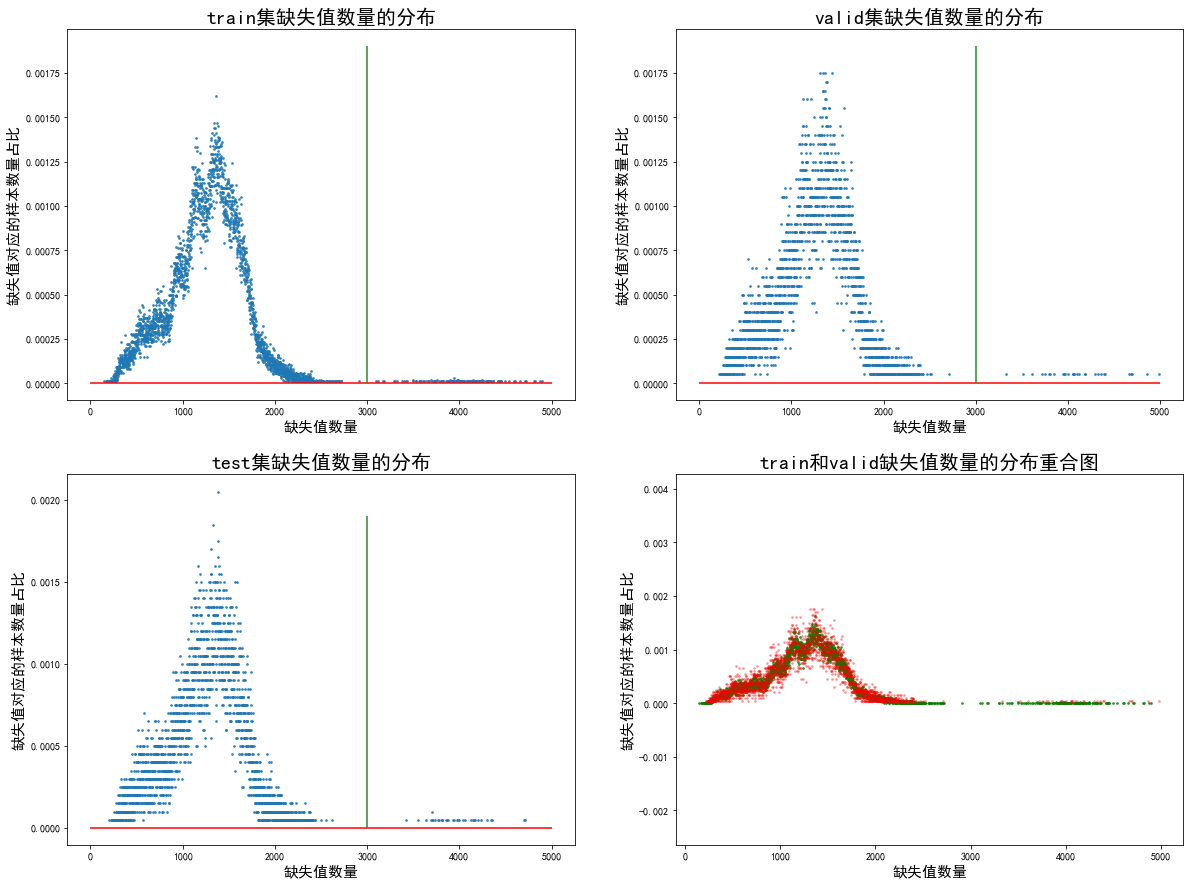


图1三个数据集的n\_null特征分布图

3. 提取了n\_null特征（每个样本的缺失特征的数量），统计了n\_null在各个数量级的分布，发现三个数据集的曲线几乎重合，于是猜想三者源于同一个分布。但是通过K-S检验，发现测试集和验证集的P值高达0.6，而训练集与验证集的P值远远小于0.05。因此我们需要从其他角度来提高我们猜想的可信度。如图2所示。

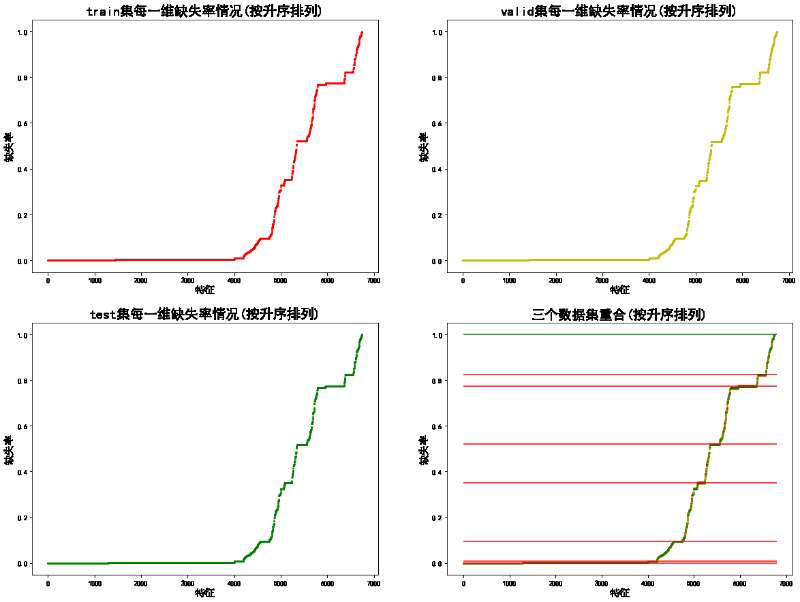


图2 各个特征的缺失率（由于特征过多，横坐标无法显示）

4. 深度挖掘loan\_dt的信息，查看数据与月份的如图3所示，训练集和测试集都分布在1~5月，两者每月样本数量占自身总体数量的比例接近，但是5月份数据过少，只有几百条左右。

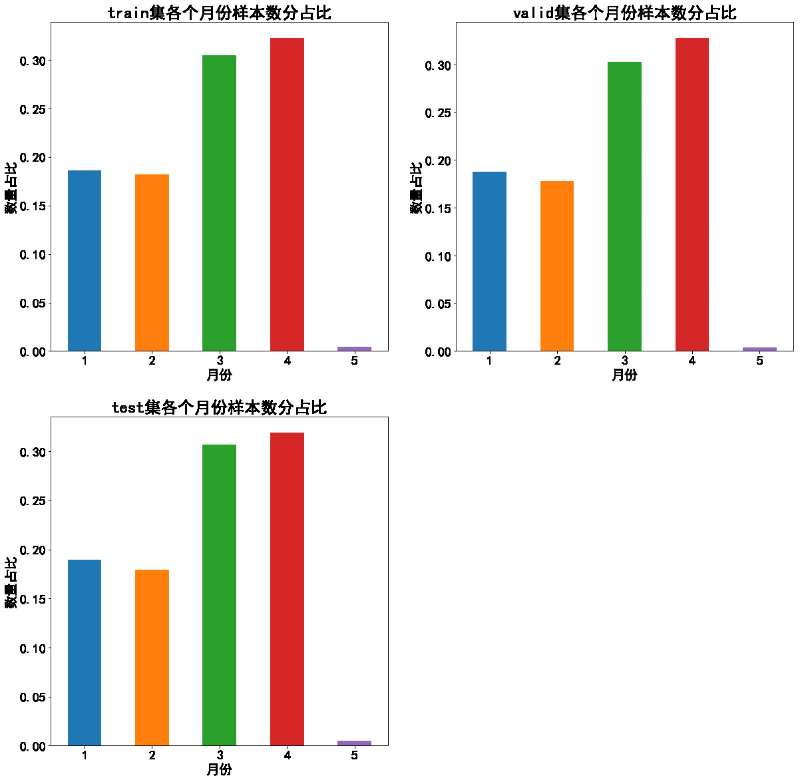


图3

5. 深度挖掘loan\_dt的信息，查看数据与日期的关系，如图4所示，三个数据集在1号到31号的分布是相似的，再次验证之前的猜想：两个数据集分布一致。

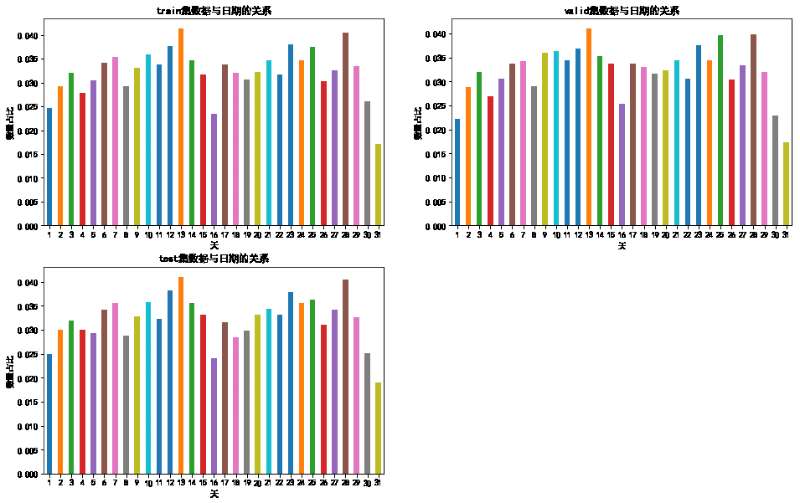


图4 月内数据数量分布

6. 深度挖掘loan\_dt的信息，查看训练集（有标签的部分）的星期数(loan\_week\_day)与逾期率(label\_ratio)的关系。如图5所示，逾期率的变化幅度随loan\_week\_day增大而增大，其中红线是基于训练集（有标签的部分）计算出来的平均逾期率。可见loan\_week\_day是一个比较有用的特征。

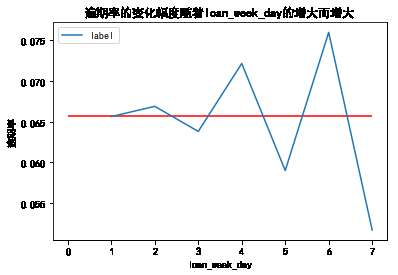


图5 逾期率随loan\_week\_day变化

7. 计算每一维的value counts，统计每一维数据众数的分布。如图6所示，在图中出现了双峰，其中右侧峰表明在相当大一部分维度上众数占了大头。于是可以构造出一类特征maj\_feature用于指示各个样本在每一维中的数值是否属于众数，然后统计各个样本特征属于众数的次数majcnt。



图6 众数数量分布

#### 2.2.特征预处理

1.将loan\_dt转化为loan\_month和 loan\_date, 后期添加了loan\_week，试图分析loan\_week和label之间的关系；

2.缺失值处理：填-1的效果就不错，因为数据绝大部分都是正数，所以填一个负数，就能很好把缺失值和正常数据给区分开；

3.将数据划分为离散值和连续值，在这里需要说明的是：不能简单地认为int64类型的特征就是连续值！也不能认为float64就是连续值。我们认为比较妥当的做法是：通过统计每一维的unique number的数量，再设定一个阈值threshold，比如说20，认为unique number<20的特征都是离散值，反之，为连续值。在图7中，阶梯型上升趋势消失在1~1.5之间，阈值可以在10^1至10^1.5之间选取。

对于离散值可以做label encode，得到data\_cat。连续值可以做分桶离散化，得到data\_con。赛后总结时，我们觉得threshold的设置可以通过交叉验证来确定，连续值分桶的数量也不应该是一成不变，应该采用IV-WOE的评价指标来确定最佳的分段方式，这样分段方式能自适应于各个特征的数值分布。

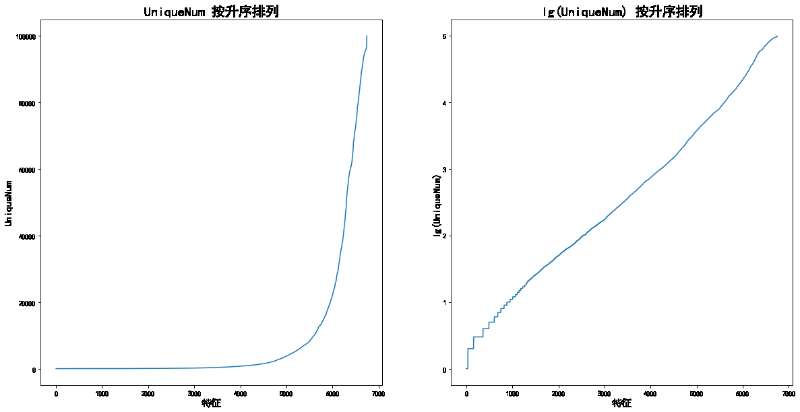


图7 特征的UniqueNum按升序排列

4.把无字段含义的特征中缺失值超过95%的特征剔除掉，特征从6745维下降到4807维，得到数据集data\_nodup。通过设置多组参数，我们来客观地对比降维前后auc分数。可以看出data\_nodup保留data\_raw的绝大部分信息。如下图8：

表1降维前后auc分数比较

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **本地验证集auc对比** | **参数组合1** | **参数组合2** | **参数组合3** | **参数组合4** | **参数组合5** |
| data\_raw | 0.6998 | 0.6119 | 0.7207 | 0.7325 | 0.6699 |
| data\_nodup | 0.7035 | 0.6002 | 0.7137 | 0.7391 | 0.6643 |

图8 降维前后auc分数比较

#### 2.3.特征提取

##### 2.3.1.数据的维度多过高，需要降维。

使用xgboost或其它树模型训练得到feature importance，然后按特征的重要性以step=100的步长不断往训练集中加入新的维度，并在本地做交叉验证输出test-auc分数，我们可以绘制auc的变化曲线，根据图来确定最佳的特征数量的大致范围。然后调整step=10，再进行精确地搜索。在后面的模型构建中，我们会详细讲到。

#### 2.4.新特征构建

**1.指示特征**

null\_sign : 给每一个特征生成一个是否为缺失值的指示变量。

maj：给每一个特征生成一个是否为众数的指示变量。

值得指出的是：指示变量不仅可以扩充特征数量，还是我们提取统计特征的基础。

**2. 统计特征**

面对这种没有明确含义的特征，怎么来构建特征呢？一个方法是提取数据的统计特征，例如之前提到的n\_null和majcnt(每一个样本在多少个维度上是属于众数)

**3.时间特征**

在数据探索的过程中，训练集和测试集在loan\_month、loan\_day、loan\_week\_day上的数据分布是相似的，故它们可以作为数据集划分的重要角度。

**4.特征交叉**

对特征做两两相乘，构造了新特征，比如f1和f2相乘，得到新特征f1\*f2。鉴于本数据集维度过高，使用feature score在前100的特征。在前100的原始特征中做选出缺失值较少的特征，一共86维。86维特征两两相乘，得到7396个新特征。之后只使用这7396个新特征训练xgboost模型，依据模型输出的特征重要性分数挑选适当数量的新特征加入到原始的特征集合中。

**5.tag特征**

对验证集和测试集中tag特征作预测。

预测tag为0或1的问题：

优势：较之label，我们可以使用全量数据，意味着我们的预测会更加准确

劣势：我们并不知道测试集上tag的分布，即tag为0：tag为1的比例

思路：

1.训练集的tag是三七分的，而我们知道三个数据集分布相似

2. 我们把tag的预测为1的概率值结果，画一个核密度估计图，如下图9，概率值成双峰态，我们去波谷作为切分点，大概是0.4左右的位置，我们发现分割点两侧的数量大致为3：7。

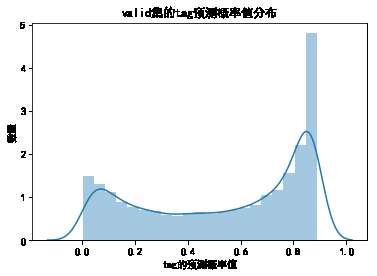


图9 验证集中tag的预测概率分布

### 3.模型构建

#### 3.1 模型选择

备选的的学习器主要分为两类：线性学习器、基于树的学习器（无监督学习不考虑 ，例如KNN）

**1.线性学习器的表现巨差**

尝试逻辑回归和SVM，其中线性核的SVM的线上AUC成绩不超过0.6，。我们认为原因有两点：一是“喂”给线性学习的数据存在离散值，没有对离散值做one-hot编码；二是对非线性数据，没法直接用线性分类器来划分，除非对特征做一些非线性的转化，比如特征做平方，。

**2.树相关的模型表现最好**

在本次比赛中我们尝试了random forest、gdbt、xgboost和lightgbm四种学习器，其中随机森林速度最快，GDBT最慢；XGBoost的效果最好，分类得更加精细；LightGBM虽然在内存使用上比较高效，但是把特征直方图化，导致信息丢失。四种学习器的比较如下表1所示：

表2 学习器比较

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **学习器** | Random Forest | GDBT | XGBoost | LightGBM |
| **参数** | n\_estimators: 200,  max\_depth: -1,  n\_jobs: 8 | n\_estimators: 200,  max\_depth: 3,  learning\_rate: 0.1 | num\_boost\_round: 200,  max\_depth: 5,  learning\_rate: 0.05,  n\_jobs: 8 | num\_boost\_round: 200,  max\_depth: 5,  learning\_rate: 0.05,  max\_bin: 200,  n\_jobs: 8 |
| **训练速度**（训练33465条有标签数据，使用所有原始特征+少量构建的特征） | 80.64 s | 1509.91 s | 62.89 s | 31.93 s |
| **训练效果**（线上最好AUC分数） | 0.5082 | 0.5288 | 0.7181 | 0.7035 |
| **原理** | Bagging | Boosting | Boosting | Boosting |

#### 3.2 本题目的难点

1. 本地做的操作在1）交叉验证 2）按照label和tag分层抽样地本地验证集 中均不能很好地得到验证，比如构建的新特征在本地做交叉验证，test-auc是有明显提升的，但是提交结果至线上验证集发现auc分数下降。

2.训练集中有60000多条unlabel数据，需要使用半监督学习。

3.训练集有tag这个特征，但测试集没有这个特征。我们的办法是使用全量训练集的特征，以tag为标签来训练模型，以用于预测测试集中的tag。

4.测试集tag的预测。Predict tag做过两次，但是两次的对应prob分布不太一致，没法判断究竟哪一个模型是更准确的。妥当的做法是在全量数据中划分一部分做为本地验证集，以此来判断。

因为模型预测出来的只是tag为1的概率，所以我们需要给出一个阈值，来划分tag为0还是1。本次比赛的做法是：基于一个先验假设条件:测试集中tag的分布与训练一致，即tag的01数量是3：7，这种方法欠妥。有一个想法是在全量训练集做5折划分，每次预测一折训练集数据（避免data leak），最后观察tag为1的概率与tag本身的分布关系,找到tag预测概率与tag真实值之间的规律。

#### 3.3 主要基础模型（用于模型融合）

##### 3.3.1 lgb\_pred0819

|  |  |
| --- | --- |
| lgb\_pred0819 | |
| 特征集 | loan\_month, loan\_day, loan\_week\_day, nodup, n\_null , tag, cross feature |
| 学习器 | 单个lightGBM |
| 线上验证AUC | 0.819 |

##### 3.3.2 xgb\_pred0829

|  |  |
| --- | --- |
| xgb\_pred0829 | |
| 特征集 | loan\_month, loan\_day, nodup，n\_null |
| 学习器 | 36个xgboost 做参数扰动，再做blending |
| 线上验证集AUC | 0.8296 |

##### 3.3.3 xgbtest1\_pred\_083046

|  |  |
| --- | --- |
| xgbtest1\_pred\_083046 | |
| 特征集 | loan\_month, loan\_day, nodup，n\_null |
| 学习器 | 36个xgboost 做参数扰动，根据mic matrix 绘制热力图，挑选相关度较低的模型融合，如图7所示。 |
| 线上验证集AUC | 0.8305 |

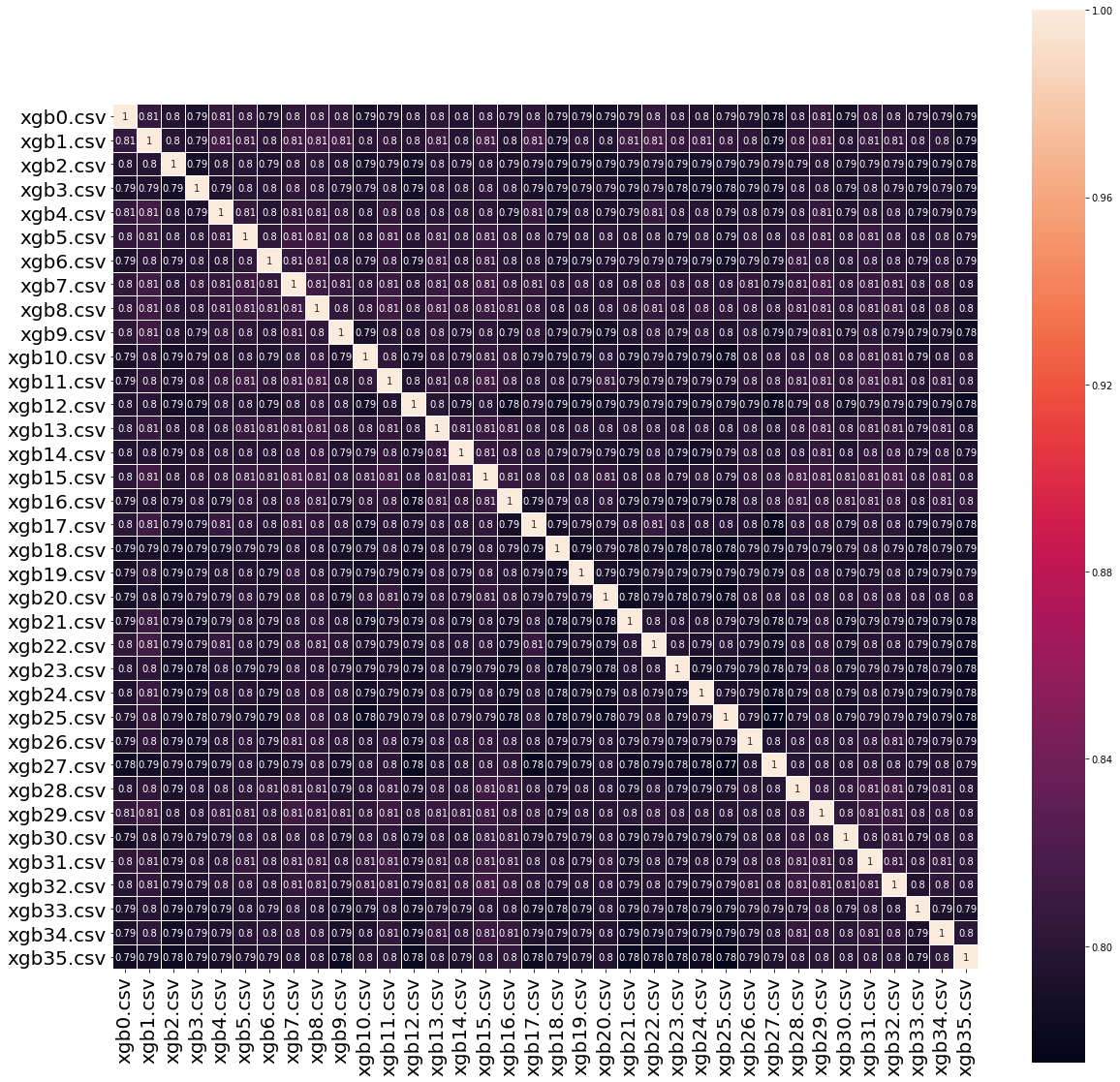


图10 36个xgboost预测结果的热力图

##### 3.3.4 gbc\_pred0819

|  |  |
| --- | --- |
| gbc\_pred0819 | |
| 数据集 | loan\_month, loan\_day, nodup，n\_null，tag |
| 学习器 | 单个GDBT |
| 线上验证集AUC | 0.819 |

##### 3.3.5 xgb\_pred\_330\_08242

|  |  |
| --- | --- |
| xgb\_pred\_330\_08242 | |
| 数据集 | 取“特征大集合”中特征重要性分数top330，图8（step=100）和图9（step=10）展示了交叉验证的test-auc随所取特征数增加的变化趋势。 |
| 学习器 | 单个xgboost |
| 线上验证集AUC | 0.8242 |

注：nodup, tag, loan\_month, loan\_day, loan\_week\_day, null\_sign, majcnt, cross\_feature 为“特征大集合”

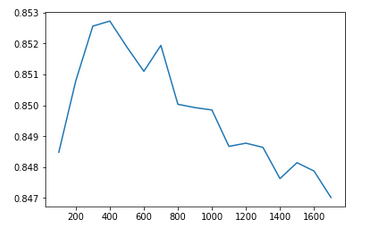


图8 粗搜索

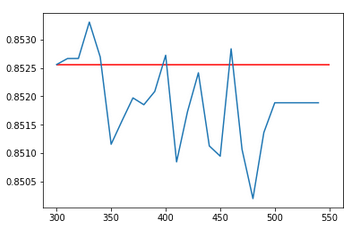


图9 精搜索

##### 3.3.6 lgb\_pred\_en\_8079

|  |  |
| --- | --- |
| lgb\_pred\_en\_8079 | |
| 数据集 | loan\_month, loan\_day, nodup，n\_null，tag |
| 学习器 | 20个lightGBM |
| 线上验证集AUC | 0.8079 |

##### 3.3.7 semi\_pred1

|  |  |
| --- | --- |
| semi\_pred1 | |
| 数据集 | loan\_month, loan\_day, nodup，n\_null，tag（33465条有标签数据+6653个负例（unlabel数据）+ 1997个正例（unlabel数据）） |
| 学习器 | 单个xgboost |
| 线上测试集AUC | 0.8200左右 |

Semi\_pred1模型构建的大体思路如下图11所示：

1. 使用通过的数据（tag=0）+少量有标签的拒绝数据（tag=1）训练 KGB（Known Good and Bad）模型
2. 使用全量训练数据，以tag作为预测标签，训练AR(Accept and Reject)模型，我们认为

这里学出来的AR模型在一定程度上能够代表平台自身的历史模型。接下来，分别使用KGB模型和AR模型对无标签的部分进行预测，分别得到概率形式的和，使用图中的公式计算出，把作为unlabel数据集的最终预测概率。

1. 传统的半监督方法，尽量取置信度较高的部分。取预测结果的10分位数之前的样本作为新的负例样本；取预测结果的97分位数作为正例样本。



图11 KGB和AR模型融合的思路

#### 3.4 集成学习

**1.Stacking**

第一层我们采用了xgboost、random forest、lightgbm三个模型。使用5折法预测训练集的标签和测试集的标签。

第二层使用第一层的训练集预测结果作为输入特征，使用多种模型进行训练，然后做模型blending，线上验证集AUC得分为0.8297。

总结：stacking技术对xgboost、lightgbm之类的强学习器基本没有提升，而且参数的调整也是一个大问题，需要对多个模型

**2.Blending：多个模型预测结果的线性组合。**

首先要肯定的是：在大多数情况模型预测结果融合后的结果都会好于单个模型的。其次，在选模型的选取方面，一般注意两个点：1)模型本身的素质要尽可能好，即AUC分数应该尽可能高； 2)模型之间的相关度不宜过大，我们通过两两计算模型预测结果的最大信息系数（MIC）来得到MIC matrix，再根据MIC matrix绘制热力图如下图12所示。

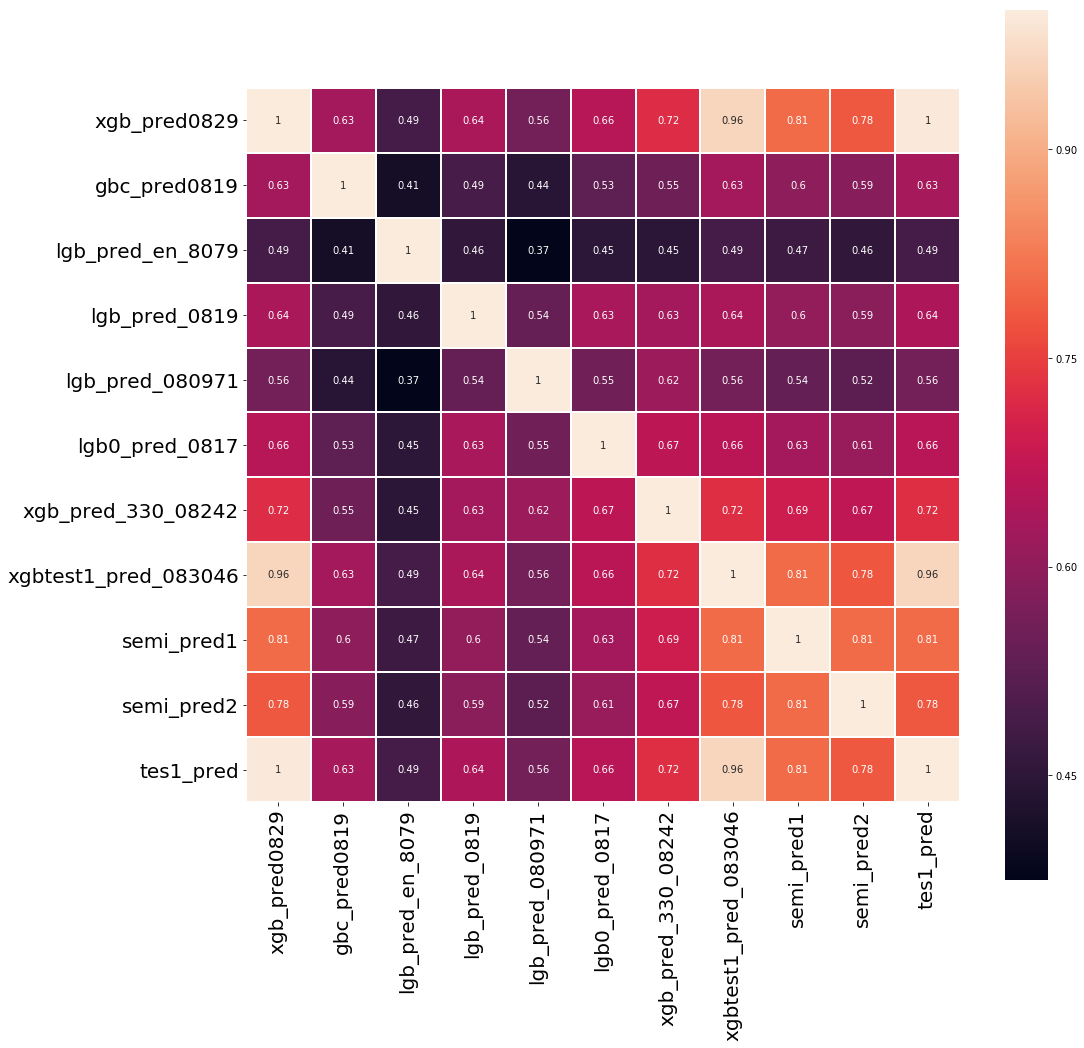


图12 基础模型预测结果的热力图

#### blending最优组合

表2 模型融合过程

|  |  |  |  |
| --- | --- | --- | --- |
| 融合步骤 | 计算公式 | 说明 | 测试集AUC |
| Step1**（三个臭皮匠顶个诸葛亮）** | **Test1 = (3\*xgb\_pred0829.prob + gbc\_pred0819.prob + lgb\_pred\_en\_8079.prob + lgb\_pred\_0819.prob + lgb\_pred\_080971.prob )** | 使用上述的预测结果，做线性组合，给予auc高的模型更多的权重。Test1是在验证集上得分最高的组合。 | 0.82399 |
| Step2 | **Test1 = (3\*xgb\_pred0829.prob.rank() + gbc\_pred0819.prob.rank()+ lgb\_pred\_en\_8079.prob.rank()+ lgb\_pred\_0819.prob.rank() + lgb\_pred\_080971.prob.rank() )** | 在Step1的基础上，使用排序融合，使得所有的预测结果都能在同一个“平台”上进行融合。 | 0.8253 |
| Step3 | **Test2 = (test1\_pred.prob.rank() + gbc\_pred0819.prob.rank() + lgb\_pred\_0819.prob.rank() + xgb\_pred\_330\_08242.prob.rank() + semi\_pred2.prob.rank())** | Test2把Test1与其他得分比较高的模型做融合，融合的方式有别于Test1，使用的融合Rank average。 | 0.8260 |

### 工具

1.环境：python 3.6 + jupyter notebook

2.python包（常见的就不说了）：

Joblib工具包，主要有两个实用功能：1）可以保存所有python对象，对于体积较大的数据集，可以采用lz4算法压缩+序列化写入，较pandas读写，平均快5倍；

2）对常见的循环语句，能采用解析式的方式方便地多进程化。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **本地验证集auc对比** | **参数组合1** | **参数组合2** | **参数组合3** | **参数组合4** | **参数组合5** |
| 6745维 | 0.6998 | 0.6119 | 0.7207 | 0.7325 | 0.6699 |
| 4807维 | 0.7035 | 0.6002 | 0.7137 | 0.7391 | 0.6643 |