基于大数据的
用户贷款审批结果预测

——融 360 金融数据分析大赛

2017. 03. 16

目录

1. 概述

融 360 成立于 2011 年 10 月，致力于为个人消费者和微小企业提供金融产品服务，业务范围涵盖贷款、信用卡与理财等。融 360 是中国最大的网络贷款平台，每天撮合数万笔贷款，每年撮合的贷款量达数百亿元。

融 360 属于金融服务性平台，平台的一端是亿级别有借款需求的小微企业和个人消费者，另一端是有贷款资金的万级别的金融机构（银行、小贷、担保、典当等）和百万级的金融产品，平台通过搜索和推荐服务来撮合借款用户和贷款。在此过程中，融 360 的利润来源主要有两方面，一方面是向金融机构精准推荐有效的贷款用户，每个收费在 50 元至 100 元，在总体营收中占比约 80%，另一方面是融 360 向贷款成功的用户收取贷款额的 15%作为佣金，在总体营收中占比 15%左右。所以为提升业务，融 360 既想尽可能多地把审批能通过的用户推荐给银行，即提高查全率，同时也要保证推荐给银行的用户中能审批通过的比率，也就是查准率。为了平衡查准率和查全率，我们将 F1-score 作为模型结果的衡量指标。

2. 数据预处理

此次案例包含四个表，并且表中含有大量缺失值，所以我们数据处理的重点放在如何合并表和删除缺失值。

2.1 基本数据描述

初始数据包含四个表，分别为：48433 条贷款产品记录(product_final_)、143152 条订单训练集记录(order_train)、238,958 条用户浏览习惯记录(user_final)和 1,506,097 条用户信息记录(quality_final)，为

了方便接下来的部分都用 product、order、user、quality 分别代指这些数据表。

订单训练集记录描述了订单的详细情况；贷款产品记录描述了贷款产品的基本属性，包括所属城市、产品类型、还款方式等；用户浏览记录主要包含了用户在网站的访问数据，例如各种点击率以及申请次数等；用户信息记录则包括了用户在网站某一次申请中填写的所在城市、申请类型、个人信息等数据。

2.2 数据去重及合并

对于 user 和 quality 表中重复的记录进行删除。为了便于连接，对于 user 表中的主键 user_id 重复的记录，只保留注册时间(date)距离现在最近的一条，但 order 和 user 同时含有名为 date 的变量，为了防止同名属性的出现，我们删掉了 user 中与结果无关的 date 变量。最终这两个表的数据量变为 79741 和 287743。

经过以上处理之后，首先我们将 order 和 user 数据集按照 user_id 进行左连接形成新数据集 dataset；然后按照 product_id 对 dataset 和 product 数据集进行左连接来更新 dataset；

最后一步是 dataset 与 quality 的链接，最后需要注意的是 quality 中除 user_id 之外属性 term、limit、standard_type、guarantee_type 也和 dataset 相对应，而我们预测的对象是一条订单记录而非一个用户，所以用户申请的期限、金额、产品类型和担保方式在整个借贷流程中也应前后一致，因此最后一步的连接有五个参照变量，这样比较符合实际情况。我们合并后的最终数据集 dataset，包含 188204 条记录，132 个变量，数据样例如图所示：

	user_id	limit	term	standard_type	guarantee_type	product_id	date	result	pv	pv_index_loan	pv_apply_total	pv_ask	pv_calculator
1	000013046dc0ab29ef8773eb61aba623	35	36	2	2	1d1e5d92fafd3175536e1b4f1e74c9c7	1278	0	29	1	1	1	0
2	0000cfcadfa43ef8246694050811165b1	5	6	1	2	b937958e1e97402d2dfc7002ad0552e4	1241	0	18	1	1	0	0
3	000170593a3bd989db39418b2cafea22	20	36	1	2	6974ce5ac660610b44d9b9fed0ff9548	989	0	1	0	0	1	0
4	000170593a3bd989db39418b2cafea22	50	36	1	2	2723d092b63885e0d7c260cc007e8b9d	989	0	1	0	0	1	0
5	0001b6ac89d6e8bddd0b25c61fbaca54	3	36	2	2	a6c61190cd0db7154df591f918848447	1332	0	NA	NA	NA	NA	NA
6	00022ff583ac26e4d182c2c9424f42e3	5	12	2	2	db2786e8c37726a88ed240411dc38a2e	1409	0	NA	NA	NA	NA	NA
7	00031cefb8bf3010f322ae3d6b3ee943	1	12	1	2	83c3c98eb2a13893621a0a499cdf1755	1528	0	NA	NA	NA	NA	NA
8	00038e93b23e7e926bbaeb24faa5ffa8	5	12	1	2	ae1eaa32d10b6c8869817555d579fb4d8	1365	0	NA	NA	NA	NA	NA
9	00050d92f53f8afe2cacd46b2379ee94	10	12	2	2	c635c6bde43a4a92a801a23bb12220e1	1171	0	27	0	5	0	0
10	0006505b4454ffdf35a98425e6906391	5	12	1	2	6101903146e4bbf4999c449d78441606	1095	0	NA	NA	NA	NA	NA
11	0006e33fc07dca7e3b07762c6b5fc758	5	12	2	2	284d024f4ce046b80e11913d8bc11c3c	1471	0	NA	NA	NA	NA	NA
12	0006ee1acc5321260ba0fbff1fb8f649	5	12	1	2	b6aec1a501920a9b470fef2a50cff1ab	1318	0	7	0	1	0	0
13	00073f98d33ea65b550737639c766283	50	12	2	1	040521102fa2051f11acd3328dea62fd	1479	0	NA	NA	NA	NA	NA
14	00083f7b8c6170a3fc976d7b99245259	6	24	1	2	a00145ad8b73b2fb0ca7709aec5b8cdd	1388	0	2	0	0	0	0
15	0008b30fba90c724bbc1d57a1cc983de	50	120	1	1	34c19f785e19b343fe2f54b3202f632c	1165	0	32	0	5	1	0
16	0008c5c05149bf5196f038a8fd1b550d	6	12	1	2	dc20d1211f3e7a99d775b26052e0163e	1199	1	1	0	0	1	0
17	0009bc1bab5b0052c20bde1ddd2174bd	8	12	1	2	605ac7e4c16b8a013b4779b81f883e66	1435	0	6	1	1	0	0
18	0009e01cc035cea26c848ace02ba4ec3	8	24	1	2	a941493eeea57ede8214fd77d41806bc	1148	1	NA	NA	NA	NA	NA
19	000ab60e6200d8468b5a1b070479c3d3	5	12	2	1	d9210a40dfa99fc7111038d8e178c343	1327	0	NA	NA	NA	NA	NA
20	000af6d3377afd28fd0172fee0334c61	20	36	1	2	ce55b334a808bd24c4af58554521ea74	1303	0	15	0	2	0	0
21	000c7812b625690871b75791d3ac899e	2	12	1	2	9ec1994bb176cb0b7a2cec414fe8ffb2	1084	0	NA	NA	NA	NA	NA

图一： dataset 数据样例

2.3 缺失值处理

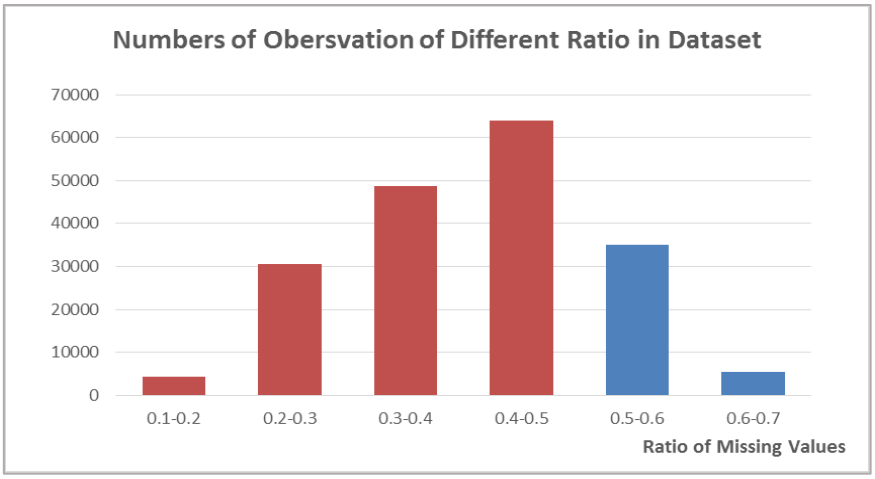
由于行、列数据中含有大量缺失值，信息可用性较小，因此将缺失值比率太高的行或列去掉。

我们取 30%为列缺失值比率阈值，若该列缺失值数量占列总数的百分比多于30%，则视为稀疏变量；同样对行进行这样的处理，不过在删除列之后，数据集中缺失值有所减少，故我们提高行缺失值比率阈值为 50%。

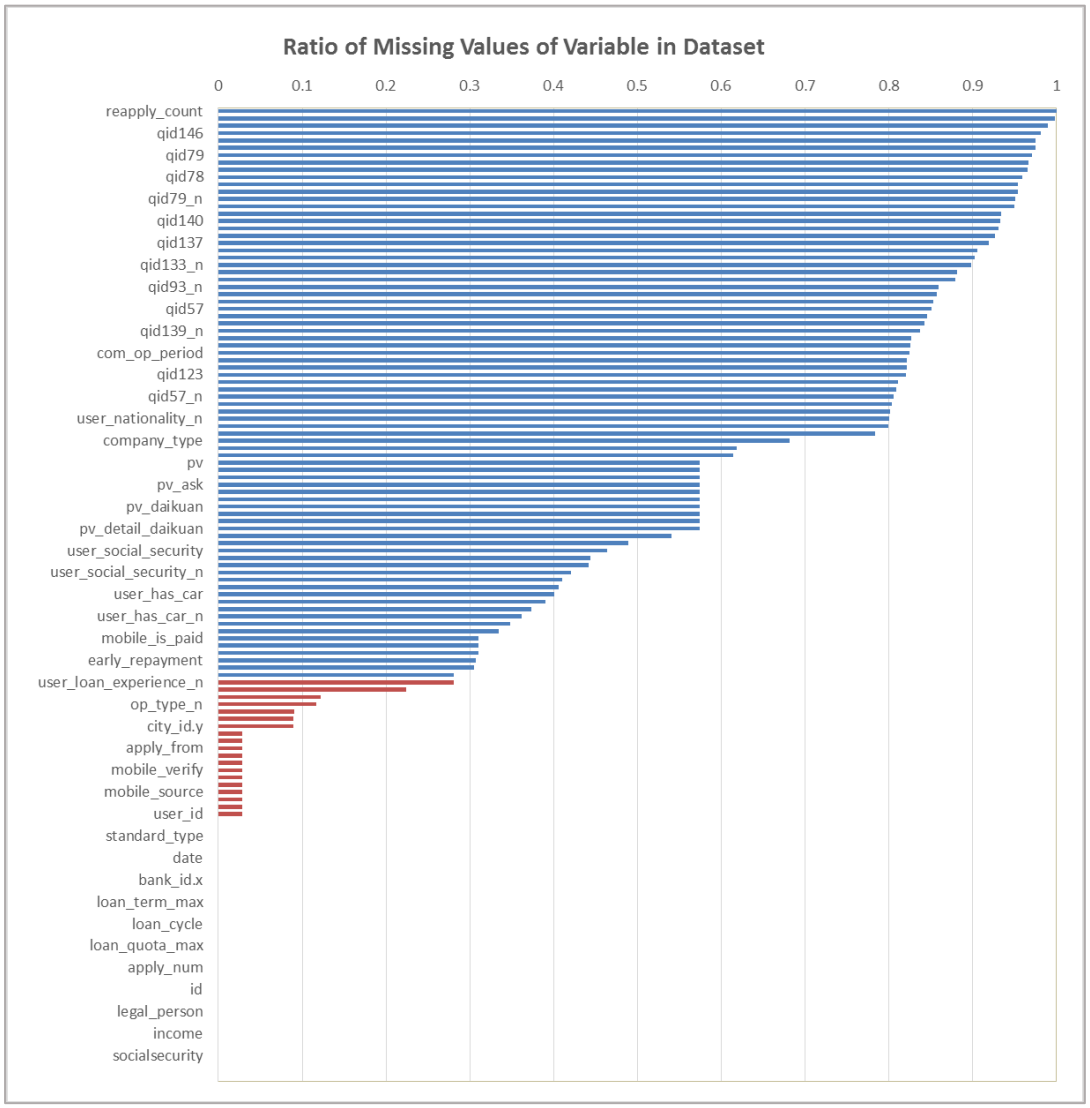
行、列缺失值比率统计分别如图二、图三所示。其中，红色部分缺失值比率未超过阈值，而蓝色部分则是比率超过了阈值的变量或记录，也就是说，蓝色部分的数据会被删除。

数据预处理并不涉及填补缺失值，因为缺失值太多，直接填补耗时长而且效果差，因此我们针对不同的算法选择不同的填补方式，在后面的模型部分会提到相关内容。

Ratio of Missing Values	No. of Obersvations	Proportion
0.1-0.2	4389	2.33%
0.2-0.3	30456	16.18%
0.3-0.4	48650	25.85%
0.4-0.5	64053	34.03%
0.5-0.6	35143	18.67%
0.6-0.7	5513	2.93%
总计	**188204**	**100.00%**



图二：行缺失值比率分析



图三：列缺失值比率分析

2.4 清洗无关变量

删除 id 类变量和重复变量不影响预测结果，dataset 含有下面七个无关变量 bank_id.x、bank_id.y、city_id.x、city_id.y、user_id、product_id、product_type.y（注册日期 date 变量在合并数据处已经被删除）。

2.5 日期数据处理

此时数据集 dataset 中只有一个时间型变量 date（来自 order 表）。如果简单的将其用天数表示所代表的信息不大，并且该数据随时间不断递增，因此我们将它化成 factor 类型，分为 7 个 level，分别代表周一至周日。我们倾向于认为，可能

在实际生活中周几对用户借贷申请有更大的影响，而且结果确实变好了。

2.6 变量 factor 化

此时 dataset 只含有 62 个变量，我们用 ls.str()查看所有变量的类型和取值，并结合它们的实际含义找出了 27 个 factor 型变量，详细见表一。

表一：factor 变量

FACTOR型变量		
result	application_type	repayment_type
guarantee_required	car	user_loan_experience
tax	income	is_paid
loan_term_type	married	guarantee_type
business_license	mobile_verify	qid77
house	product_type.x	house_register
interest_rate_type	quality	col_type
lifecost	socialsecurity	id
platform	bank	op_type

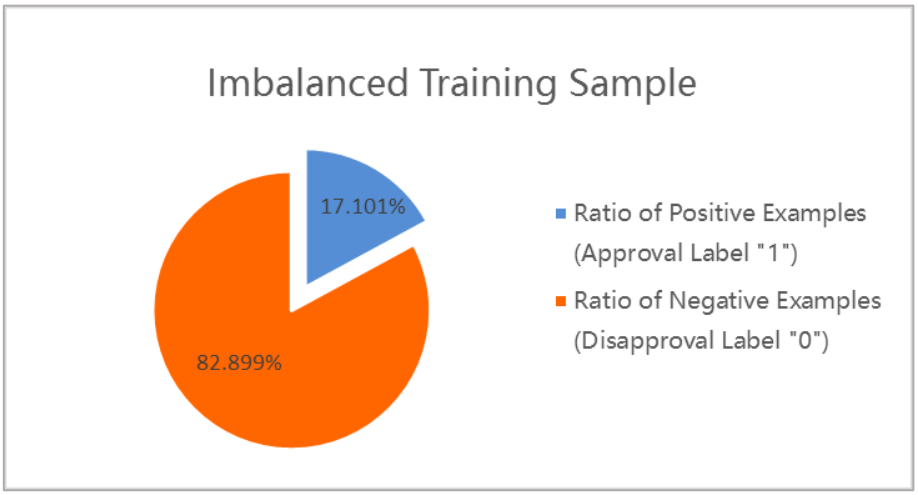
2.7 低方差滤波

除此之外，由于方差过低的数据列所含信息也较少，因为我们使用了 caret 包中的 nearZeroVar()函数，利用其默认阈值对数据集进行了低方差滤波处理，最后删去了 7 个变量。

表二：低方差变量

Variables	freqRatio	percentUnique	zeroVar	nzv
tax	31.06810204	0.001459566	FALSE	TRUE
guarantee_required	215.4723539	0.001459566	FALSE	TRUE
is_paid	19.10897729	0.001459566	FALSE	TRUE
is_p2p	0	0.000729783	TRUE	TRUE
legal_person	73.02863317	0.001459566	FALSE	TRUE
user_social_security_n	24.11980033	0.001459566	FALSE	TRUE
qid77_n	26.54383736	0.001459566	FALSE	TRUE
mobile_is_paid	48755	0.001459566	FALSE	TRUE

经过整个数据预处理过程后，dataset 数据集还剩下 54 个变量，137027 条记录。其中，正负样本比例如下图所示，相比于初始样本的 17.02%/82.97%变化很小。



图四：dataset 中正负样本比例

3. 模型

我们将上述最终数据集 dataset 通过分层抽样的方式，按照 3:7 分为测试集 testset 与训练集 trainset。我们将用训练集中数据训练学习器，用测试集来检验泛化性能。

在本部分，我们将介绍两种学习器：随机森林与 xgboost 模型，给出其运行结果，并分别加以分析。

3.1. 随机森林

随机森林是用随机的方式建立包含很多决策树的森林，每棵树之间没有关联。随机森林算法适用于存在大量特征的数据集，并且具有很好的抗噪声能力，故这里使用随机森林分类器做预测。

3.1.1 缺失值填补

由于随机森林内置一个函数 na.roughfix()用以填补缺失值，对于数值型变

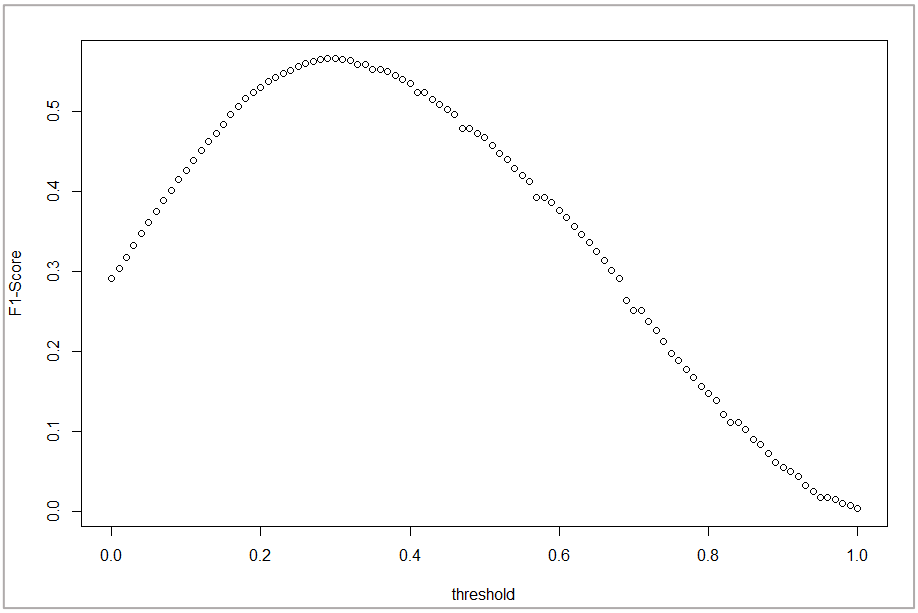
量，用列数据中位数来填补 NA；对于 factor 型变量，用频率最高的 level 来填补 NA。我们分别对训练集以及测试集填补缺失值，生成了 nf.train 以及 nf.test 数据集。

3.1.2 评估泛化性能和重要变量选取

1) F1-score

经过 3.1.1 部分的处理后，随机森林模型选用 100 棵树进行运行，然后用训练好的模型对 nf.test 测试集预测，在这里需要注意的一点是，虽然随机森林可以选择预测结果直接返回 0/1 变量，但我们试验之后发现效果不佳，因此设置返回 0~1 的浮点数，自行设置阈值并用函数寻找最优的 F1-score。

阈值与 F1-score 值的关系如图五所示，最终运行出的 F1-score 结果为 0.5657251，最优阈值为 0.31。



图五：随机森林中阈值与 F1-score 值关系

2) 混淆矩阵

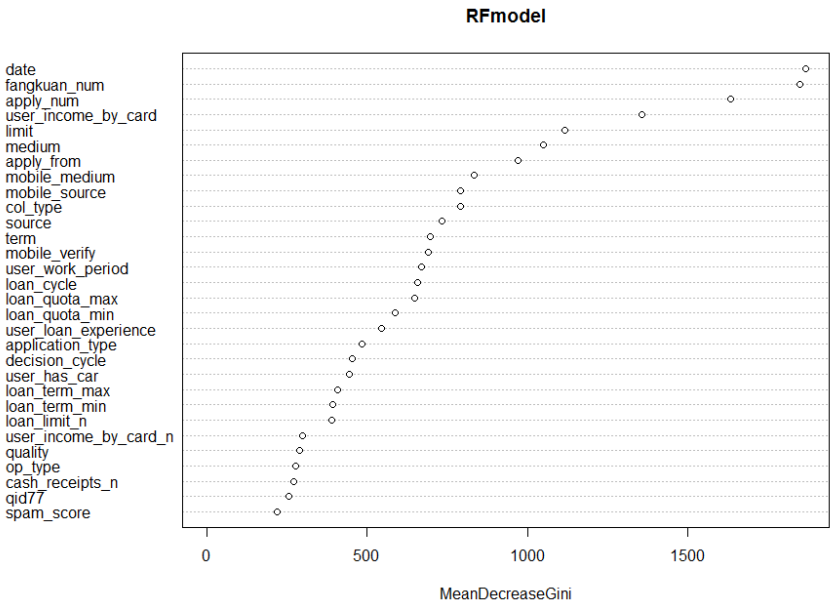
模型最后生成的混淆矩阵如表三所示，

表三：随机森林混淆矩阵

RandomForest（不拆分order）		
预测值	真实值	
	0	1
0	30199	2916
1	3179	3936

3) 关键变量

图六显示了前三十个重要变量，其中 MeanDecreaseGini 通过基尼指数计算每个变量对分类树上每个节点的观测值的异质性影响。



图六：运用随机森林选取的重要变量

3.2. Xgboost

xgboost（extreme gradient boosting）是 Gradient Boosting Machine 的一个 c++实现。它是一种速度快效果好的 boosting 模型，做分类任务时基本原理就是将很多分类准确率较低的树模型组合起来成为一个准确率高的分类器，一般比 randomForest 包占用内存更少并且运行速度更快。

3.2.1. 模型参数设置

R 中的 xgboost 函数含有多个参数，主要分为通用参数、Booster 参数和学习目标参数，其中可以通过调节 booster 参数处理类别不平衡和过拟合问题。

1)max_depth 为树的最大深度，可以用来避免过拟合，调试后使用默认值 25

2)objective 定义模型中需要被最小化的损失函数，其中"binary:logistic"代表二分类的逻辑回归并返回预测的概率

3)scale_pos_weight 主要针对类别不平衡问题，可以控制正负样本的平衡，取值为训练集中负、正样本数的比例

4)eval_metric 用来指定验证数据的评估指标，这里使用"AUC"

5)max_delta_step 允许我们估计每棵树的权值，在数据不平衡和使用逻辑回归的情况下，设置为一个正值可以帮助更新并且模型更为保守,

6)subsample 是指训练实例的子样品比，设置值为 0.8，表明 xgboost 可以随机收集 80%的数据来防止过度拟合

7)eta 代表更新步长止，通过减少每一步的权重可以提高模型的鲁棒性，防止过拟合，最终值为 0.1

8)min_child_weight 决定最小叶子节点权重，这个参数可以用于避免过拟合，最终值为 2

3.2.2. 模型泛化性能评估

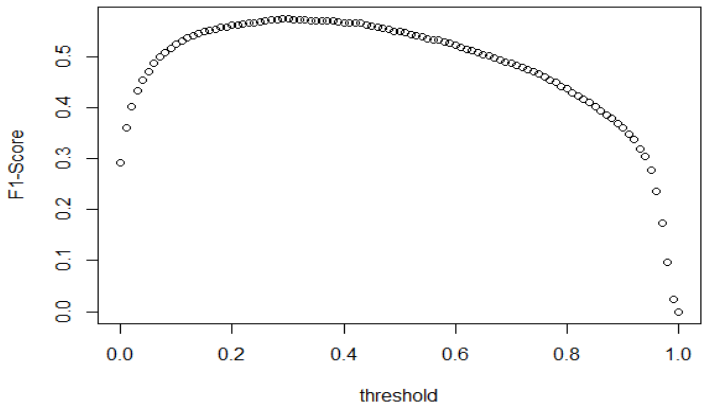
设置上述参数后，用训练集运行模型，迭代 100 次得出结果。

由于 xgboost 可以自行处理缺失值，因此我们不再进行填补，直接将训练集和测试集矩阵化。调参之后正样本阈值与 F1-score 的关系如图七所示，最优 F1-score 为 0.5745312，最终准确率为 0.8465，混淆矩阵如表四所示。

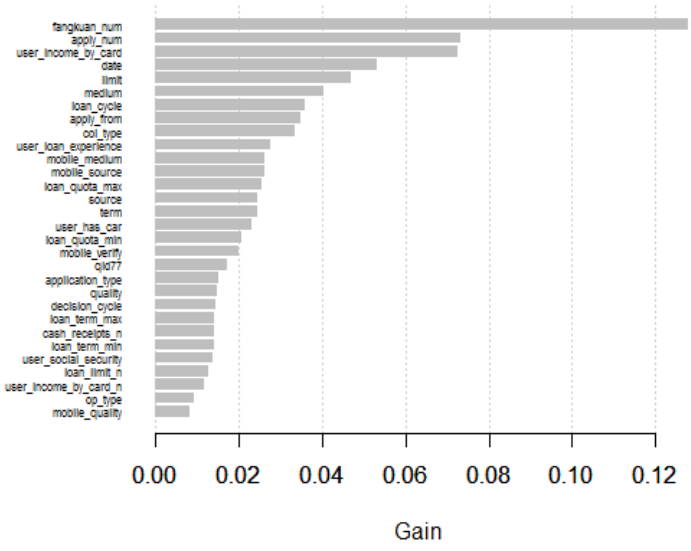
表四：xgboost 混淆矩阵

XGBoost（不拆分order）		
预测值	真实值	
	0	1
0	30540	2770
1	3538	4279

xgboost 的重要变量是以 Gain 为指标，Gain 表示决策树在进行该特征属性选择划分前和划分后的信息差值，前三十个重要变量如图八所示，其中前五个重要变量分别是 date、fangkuan_num、apply_num、user_income_by_card、limit。



图七：xgboost 中阈值与 F1-score 值关系



图八：xgboost 选出的重要变量

3.3. 模型结果比较

表五显示了 xgboost 和随机森林的运行结果，可以发现两种模型下的查准率

和查全率大致相同，前五个重要变量也只是排序不同。

表五：xgboost 与随机森林对比

	xgboost	RandomForest
Precision	0.547396700	0.553197470
Recall	0.607036459	0.574430823
F1-score	0.574531200	0.565725100
重要变量	fangkuan_num	date
	apply_num	fangkuan_num
	user_income_by_card	apply_num
	date	user_income_by_card
	limit	limit

4. 扩展

经过对大赛文件的分析，我们发现在实际比赛中最终测试集（如图九）包含的数据属性与订单训练集记录（order_train）一致，只有用户 id（user_id）、产品 id（product_id）、申请日期（date）、申请期限（term）、申请金额（limit）五个属性，因此我们在 2.2 中所进行的数据连接及预处理方法实际上是不适用于该案例的。若按照以上方法，预测前进行连接时会造成 test 数据集中部分数据丢失，因此无预测结果。

test

Filter

	user_id	product_id	date	term	limit
1	b40109bbfd7c40d020510578a3c30642	cf41e53e71fd537c038ffd266049faa5	1645	12	3.0
2	e707a6bddf84bf6e0fb1d4a9d4769c5d	2557911c1bf75c2b643afb4ecbfc8ec2	1640	12	50.0
3	11b4c689145b3aee665fd882fdd0b51b	b132ecc1609bfcf302615847c1caa69a	1567	12	5.0
4	e8f9aab6c32a975b7a61cc2df60e0e70	83c3c98eb2a13893621a0a499cdf1755	1569	12	5.0
5	1204085c63a8ca003a4327a8c62e9282	a96f8008297c6154537a77ade637d60a	1604	18	2.0
6	dab0279074bb312ae52058e78409f27a	ac0adaa01d622b692ec359294c2c8138	1593	12	5.0
7	9156a95b1c5ff01e1a9e8df7ea1bb42f	16c7dd6b8eb5d8ac77082f70cd26c28a	1581	12	5.0
8	7959da3e4c7b33163567abbf6231b5fa	46a93c9637d3b33e987a6b8bfdccd87c	1596	12	1.0
9	5947fd47e019eca0a2129399e51758c8	67f7fb873eaf29526a11a9b7ac33bfac	1593	12	3.0
10	55c4cea3b212aace6909314f8ef3c087	aff2ec0b273d9b5fe4754a604f765c38	1565	12	5.0

图九：order.test 示例

为了更好的与该大赛要求契合，我们决定换一种方式进行数据预处理之后再对以上模型进行检验。

4.1. 数据处理

首先我们利用 plyr 包中的 count 函数统计了 quality 数据集中的完全重复行，新增了一列 freq 属性用以记录行数，同时去除了 quality 中完全重复的行。为了保证 order 数据集的完备性，并且保证在之后的连接过程中能够与 order 数据集一一对应，我们保留了 user_id 相同的观测行中缺失值相对最少的那一行行。user 数据集处理方法同上（2.2）。

对初始数据进行处理之后，我们按照 3:7 的比例对 order 数据集进行划分训练集测试集，分为 order.train 以及 order.test，这与大赛的设置相似。

然后我们首先按照 user_id 对 order.train 和 user 数据集进行左连接形成 trainset，然后按照 product_id 对 trainset 和 product 进行左连接更新 trainset，最后按照 user_id 将 trainset 和 quality 进行左连接，生成了最终的初始 trainset 数据集，包含 100207 条记录，138 个变量。同理对 order.test 进行连接，最后生成了最终的初始 testset 数据集，包含 42945 条记录，138 个变量。

然后我们将 testset 作为 out of sample 不作任何处理，只对 trainset 进行数据预处理，处理步骤同上（2.3~2.7）。

经过处理，最终生成的 trainset 包含 100207 条记录，47 个变量，最终 testset 数据集，包含 42945 条记录，47 个变量。

4.2. 对比结果

表六：最优结果对比

index	不拆分order表		拆分order表	
	xgboost	randomforest	xgboost	randomforest
Sensitivity	0.8962	0.9048	0.6954	0.7589
Specificity	0.6059	0.5744	0.6698	0.5895
Balanced Accuracy	0.7510	0.7396	0.6826	0.6742
Accuracy	0.8465	0.8485	0.6910	0.7301
F1-score	0.5745	0.5657	0.4278	0.4270

从对比结果可以看出，完全模拟大赛设置处理后进行训练后的学习器效果明显下降，这是因为此种方法处理的训练集与测试集相关性较小，但是这种情况下训练的学习器更加具适用性与泛化能力。

5. 总结

我们最终得到的结果如下表所示：

表七：最终结果

index	xgboost
Precision	0.5474
Recall	0.6070
Balanced Accuracy	0.7510
Accuracy	0.8465
F1-score	0.5745

虽然运行多次最优 F1-score 都是 0.5745312，但由于 xgboost 自动缺失值的时候具有随机性，结果可能会出现偏差，但是能确定的是误差小于 0.01。

最后，参考融 360 官网数据可知，2015 年 5 月融 360 推出天机大数据风控系统，同一类用户，传统粗放式的风控算法，贷款获批率在 15%左右，而使用大数据模型结合人工后获批率可以达到 30%以上，这里的获批率就是指查准率(Precision)——实际通过的订单数量占预测通过的订单数量比率，而我们模型中查准率为 0.5745，获批率有所提升。

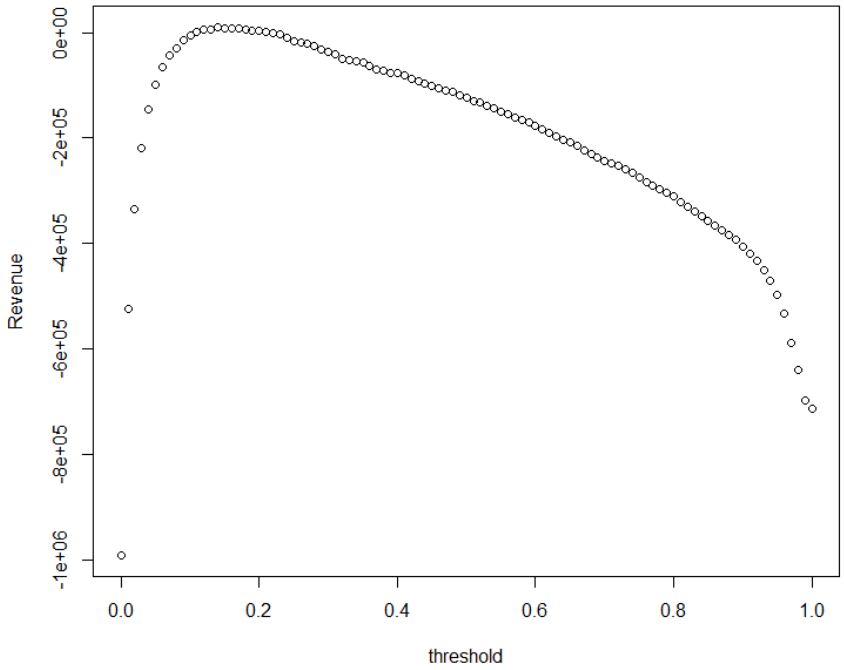
另外，联系我们最初所说的融 360 盈利模式可以对此进行利润分析，如公式所示，

$$\begin{aligned} Rev = \text{TP*Referral fee} + \sum_{TP} \alpha * \text{limit[i]} \\ + \text{FP*Referral fee} - \text{FP*Penalty fee} \\ - \text{FP*Referral fee} + \sum_{FP} \alpha * \text{limit[i]} \end{aligned}$$

1. 所有被预测为 1 实际也为 1 的用户，即 TP 部分，这些用户都将被推荐给银行机构，融 360 从中收获 50~100 元的推荐费，另外，融 360 还会向贷款成功的用户收取贷款额的 15%作为佣金。
2. 对于被预测为 1 实际也却为 0 的用户，即 FP 部分，虽然推荐给银行机构会使融 360 获得推荐费，但由于最后贷款未能获批，这会影响银行机构对公司的信用程度，我们暂且用一个惩罚费用来反映信用受损程度。
3. 对于被预测为 0 但实际为 1 的用户，即本来可以获批但却被学习器认为不可以的用户，也就是图中的 FN 部分，我们将同时损失推荐费与佣金。

在实际情况中，我们可以通过改变阈值调整混淆矩阵取值，从而达到利润最大化。

我们按照网上所给出的数据，设置推荐费为 100，惩罚费用为 150，佣金比率为 15%，最后得到的利润与阈值变化关系如图十所示。最优阈值为 0.15 在此阈值下得到的 f1score 为 0.53 此时的 f1score 是略低于最优 f1score 的，但是这种情况下的利润是可以达到最优的。



图十：利润与阈值变化

最后，我们总结了两点不足：

1. 对数据的预处理与清洗需要更为细致。

   在挑选构建模型的变量的过程中，我们对特征的选择进行的还不够深入，无可避免的存在疏忽和遗漏，变量的选择不准确可能在一定程度上对我们模型的有效性和准确性加以影响。

2. 对模型的解读能力需要提升

   在建模过程中，我们尝试通过调参优化，但由于缺乏对模型深入解读的能力，只是通过反复的试探修改，根据结果来确定最优，调参结果并不理想，因此，还需要在今后的学习中加强对模型的认识程度。