基于拍拍贷数据的评分卡模型

By Susanna Lee

一、 背景介绍

拍拍贷是一家金融科技公司，2007 年成立于上海，并在 2017 年 11 月 10 日成功于美国纽交所上市。根据官方消息，截至 2018 年 9 月 30 日，拍拍贷累计成交额已突破 1300 亿，15-29 天及 30-59 天的逾期率分别为 0.83%和 1.21%，自 2015 年，拍拍贷推出魔镜系统用于对每一笔借款进行风险评估，通过对借款人资质审核并预测未来逾期概率，对相应风险进行定价，基于大数法则保证整体的可观收益。

本文利用拍拍贷真实业务数据，预测客户违约概率，建立评分卡模型。

1.1 数据来源

(https://www.kesci.com/home/competition/56cd5f02b89b5bd026cb39c9/content)

数据来自“魔镜杯”风控算法大赛，包含 8 万个数据样本.

1.2 数据集基本情况简介

- **master**：每一行代表一个样本（一笔成功成交借款），每个样本包含 200 多个各类字段。

字段名称	描述
idx	每一笔贷款的 unique key，可以与另外 2 个文件里的 idx 相匹配
UserInfo_*	借款人特征字段
WeblogInfo_*	Info 网络行为字段
Education_Info*	学历学籍字段
ThirdParty_Info_PeriodN_*	第三方数据时间段 N 字段
SocialNetwork_*	社交网络字段
LinstingInfo	借款成交时间
Target	违约标签（1 = 贷款违约，0 = 正常还款）

- **Log_Info**：借款人的登陆信息

字段名称	描述
ListingInfo	借款成交时间
LogInfo1	操作代码
LogInfo2	操作类别
LogInfo3	登陆时间
idx	每一笔贷款的 unique key

- **Userupdate_Info**：借款人修改信息

字段名称	描述
ListingInfo1	借款成交时间
UserupdateInfo1	修改内容
UserupdateInfo2	修改时间
idx	每一笔贷款的 unique key

1.3 总体思路

本项目首先从数据清洗开始，对缺失值、字符及空格等的处理；其次进行特征工程，包括对时间滑窗统计特征以及地理位置特征构建，对特征进行分箱和 WOE 编码；然后通过 IV、线性相关性、VIF 进行特征筛选；由于数据存在类别不平衡情况，采用了代价敏感学习；对于模型训练部分，我们采用了工业界广泛应用的逻辑回归模型、XgBoost 构建评分卡模型，生成模型报告，划分用户评级；最后改写 XbBoost 的业务评价函数建立提额模型。

二、 数据清洗

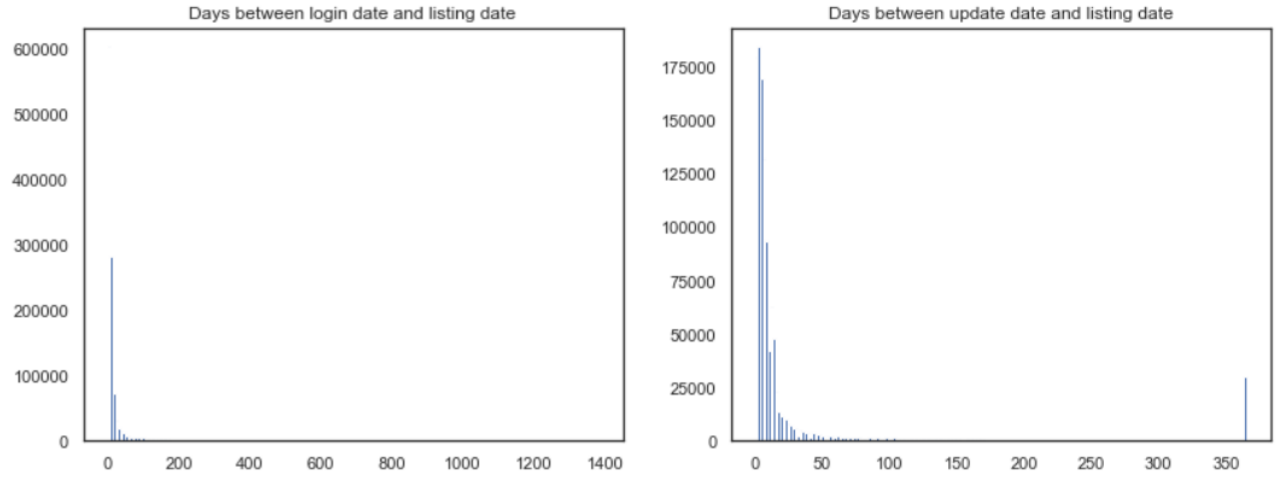
- Master 数据集：
  - WeblogInfo_1、WeblogInfo_3 字段缺失率大于 90%，需要去除。
  - UserInfo_8 中包含有“重庆”、 “重庆市”等取值，它们实际上是同一个城市，需要把 字符中的“市”、“地区”等全部去掉。
  - UserInfo_9 字段的取值包含了空格字符，如“中国移动”和“ 中国移动 ”， 利用 replace 函数将其清除。
  - UserInfo_2、 UserInfo_4、 UserInfo_8、 UserInfo_20、UserInfo_7 和 UserInfo_19 这几个地理位置字段存在不一致，比如‘恩施’和‘ 恩施土家族苗族自治州’，需要进行统一。
- Userupdate_Info 数据集:
  - UserupdateInfo1 字段中存在大小写不一致的情况，如‘_MobilePhone’和‘_mobilePhone’，利用 lower()函数全部转换为小写。
  - UserupdateInfo1 字段中'_mobilephone' 和 '_phone' 应该为一个类别，统一转换为‘_phone’。
- 剔除常变量

  原始数据中有 190 维数值型特征，通过计算每个数值型特征的标准差，剔除标准差小于 0.06 的特征。

三、 特征工程

3.1 时间滑窗统计特征

Log_Info 数据集有代表身份的 idx、代表登录日期的 LogInfo3 和操作代码 LogInfo1 和 LogInfo2，通过计算登录日期和放款日期之间的间隔天数，可以看到绝大部分的天数在 180 天以内。同样在 Userupdate_Info 数据集中计算修改日期和放款日期之间的间隔天数绝大部分的天数也在 180 天以内。



因此考虑使用 180 天作为最大的时间窗口计算新特征，所有可以使用的时间窗口可以有 7 days, 30 days, 60 days, 90 days, 120 days, 150 days and 180 days 构建时间切片，在每个时间窗口内，对 Log_Info 数据集计算：

1. 总的登录次数
2. 不同的登录方式
3. 每种登录方式的平均次数

对 Userupdate 数据集计算：

1. 更新的频率
2. 每种更新对象的种类个数
3. 对重要信息如 idnumber, hasbuycar, marriagestatusid, phone 的更新

不同的时间切片与不同的计算逻辑的交互可以产生多个特征。这些特征往往存在一定程度上的线性相关性。在接下来的多变量分析中，需要消除线性相关性对模型产生的影响。

3.2 地理位置特征衍生处理

数据提供了 6 个与地理位置相关的特征，其中 UserInfo_2、 UserInfo_4、 UserInfo_8、 UserInfo_20 四个特征为城市信息，UserInfo_7 和 UserInfo_19 为省份信息。

- **构建城市差异的特征**：考虑到本地人和外来打工人员的还款能力也许存在差异，因此我们对比不同城市特征是否相同，比如 city_diff24 为 UserInfo_2、UserInfo_4 两列的城市是否相同 city_allmatch 为 UserInfo_2、UserInfo_4、 UserInfo_8、 UserInfo 四列的城市是否都一致。
- **城市等级合并**：按照城市等级，将城市类别变量合并，一线城市北京、上海、广州、深圳合并，赋值为 1，新一线城市合并为 2，二线城市合并为 3，其他城市合并为 4。
- **城市经纬度转换**：将城市名用经纬度替换，这样就可以将类别型的变量转化为数值型的变量。

四、 特征筛选

4.1 数据集中度筛选

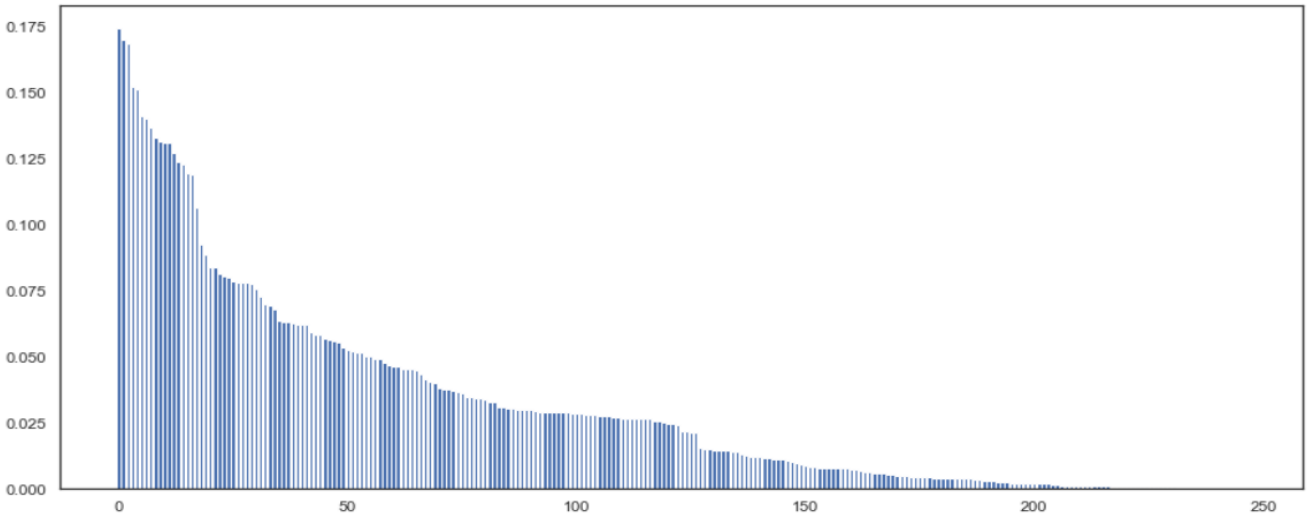
筛选出数据集中度超过 90%的特征，当“多数值”与“少数值”有显著差别，且“少数值”的坏样本率高于“多数值”，表明“少数值”的存在对应的风险很高，特征需要保留。我们用特征中“少数值”坏样本率和“多数值”坏样本率之比的对数值作为指标，超过 2.303(即两者的坏样本率比值超过 10)则表明该特征需要保留。

4.2 卡方分箱/WOE 编码

为了模型稳定性，在逻辑回归模型中我们一般都要先进行分箱处理。分箱方法分为有监督分箱和无监督分箱，我们这里采用有监督分箱里面的卡方分箱法。分箱完成后对特征进行 WOE 编码。

4.3 特征筛选

首先通过 IV 值来筛选特征，一般而言当特征 IV 值小于 0.1，则表示特征重要性程度不高，而本数据中，特征的 IV 值普遍偏低，因此我们放宽 IV 选择条件，以 0.01 位阈值进行筛选



然后检查 WOE 编码后的变量的两两线性相关性，相关性高于 0.6 的，则删除两相关特征中 IV 值较小的特征。最后利用 VIF 判断多重共线性问题。

五、 模型训练

5.1 逻辑回归模型

逻辑回归模型由于具有较好的解释性，因此在工业界评分卡模型中使用较为广泛，同时又由于逻辑回归模型结构简单，对变量要求高，因此在模型训练之前通常要对特征进行分箱和 WOE 编码处理。通过模型训练得到训练集和测试集的 KS 分别为 0.3573 和 0.3618。针对正负样本不平衡性，我们在逻辑回归中使用代价敏感采样，最终得到训练集和测试集为 0.36 和 0.3607，可以看到，训练集有千分位的提升，而测试集在千分位上下降了一个点左右。最终生成如下报告：

KS	BAD	GOOD	BAD_CNT	GOOD_CNT	BAD_PCTG	BADRATE
0.150	222	579	222	579	0.189	0.277
0.236	152	649	374	1228	0.319	0.190
0.290	118	683	492	1911	0.419	0.147
0.326	98	703	590	2614	0.503	0.122
0.335	68	733	658	3347	0.560	0.085
0.357	83	718	741	4065	0.631	0.104
0.347	48	753	789	4818	0.672	0.060
0.342	53	748	842	5566	0.717	0.066
0.336	52	749	894	6315	0.761	0.065
0.329	52	749	946	7064	0.806	0.065
0.308	36	765	982	7829	0.836	0.045
0.296	45	756	1027	8585	0.875	0.056
0.273	34	767	1061	9352	0.904	0.042
0.241	24	777	1085	10129	0.924	0.030
0.210	25	776	1110	10905	0.945	0.031
0.174	20	781	1130	11686	0.963	0.025
0.129	9	792	1139	12478	0.970	0.011
0.089	16	785	1155	13263	0.984	0.020
0.045	11	790	1166	14053	0.993	0.014
0.000	8	773	1174	14826	1.000	0.010

5.2 XgBoost

XgBoost 是基于树模型在训练，一般不需要分箱和 WOE 编码，此处为了对比两个模型效果，我们利用 XgBoost 对同样的特征处理数据进行训练，最终得到训练集和测试集 KS 为 0.3828 和 0.3768，最终效果比逻辑回归要好一些。生成报告：

KS	BAD	GOOD	BAD_CNT	GOOD_CNT	BAD_PCTG	BADRATE
0.173	247	554	247	554	0.210	0.308
0.269	163	638	410	1192	0.349	0.203
0.307	100	701	510	1893	0.434	0.125
0.338	93	708	603	2601	0.514	0.116
0.353	75	726	678	3327	0.578	0.094
0.364	71	730	749	4057	0.638	0.089
0.373	68	733	817	4790	0.696	0.085
0.360	45	756	862	5546	0.734	0.056
0.354	52	749	914	6295	0.779	0.065
0.327	30	771	944	7066	0.804	0.037
0.319	49	752	993	7818	0.846	0.061
0.309	48	753	1041	8571	0.887	0.060
0.273	20	781	1061	9352	0.904	0.025

0.247	30	771	1091	10123	0.929	0.037
0.216	26	775	1117	10898	0.951	0.032
0.178	17	784	1134	11682	0.966	0.021
0.133	10	791	1144	12473	0.974	0.012
0.087	9	792	1153	13265	0.982	0.011
0.044	12	789	1165	14054	0.992	0.015
0.000	9	772	1174	14826	1.000	0.012

5.3 改写 XgBoost 评价函数

假设有个提额模型，用处是给分数最高的 20%客户更高的额度，也就是使期望分数最高的 20%的客群正样本捕获率最大化，同时模型对整体正负样本有一定区分能力。XgBoost 支持自定义评价函数和损失函数，因此我们通过对评价函数改写来满足上述要求，将评价函数分为两个部分，第一部分权重为 0.5，目的是使得前 20%样本中的正样本占比最大，第二部分权重也为 0.5，目的是让模型对正负样本的识别能力得到保障。最终得到的结果为训练集 0.3908、测试集 0.3602。

KS	BAD	GOOD	BAD_CNT	GOOD_CNT	BAD_PCTG	BADRATE
0.156	229	572	229	572	0.195	0.286
0.246	156	645	385	1217	0.328	0.195
0.295	112	689	497	1906	0.423	0.140
0.328	95	706	592	2612	0.504	0.119
0.345	77	724	669	3336	0.570	0.096
0.338	51	750	720	4086	0.613	0.064
0.352	74	727	794	4813	0.676	0.092
0.356	64	737	858	5550	0.731	0.080
0.348	50	751	908	6301	0.773	0.062
0.333	**42**	**759**	**950**	**7060**	**0.809**	**0.052**
0.315	39	762	989	7822	0.842	0.049
0.297	**39**	**762**	**1028**	**8584**	**0.876**	**0.049**
0.268	28	773	1056	9357	0.899	0.035
0.242	**30**	**771**	**1086**	**10128**	**0.925**	**0.037**
0.211	25	776	1111	10904	0.946	0.031
0.170	**14**	**787**	**1125**	**11691**	**0.958**	**0.017**
0.128	13	788	1138	12479	0.969	0.016
0.092	**20**	**781**	**1158**	**13260**	**0.986**	**0.025**
0.046	9	792	1167	14052	0.994	0.011
0.000	**7**	**774**	**1174**	**14826**	**1.000**	**0.009**