1 概述

本题要求我们根据提供的用户数据（包括关联关系、危险行为、标签类型、app 情况，均已脱敏），通过数据挖掘技术，组合出有显著效果的特征，并利用这些特征构建模型预测用户的逾期情况。

用户是否逾期和个人信用记录、收入、所属行业等诸多个人属性强相关，因此，如何对用户的自身特征进行刻画是我们首先要考虑的问题。另一方面，人类的社会属性又决定了用户必然要同其他人建立联系，物以类聚人以群分，对用户有联系的人群进行分析也至关重要。

我们利用网络表示学习方法、文本主题模型、链接分析算法和一些统计信息对用户自身和周围联系人进行了建模，经过参数选择，得到了一个融合了多个 XGboost、lightgbm 分类器的模型。

我们在验证集上最好成绩为 0.7119，测试集上的 AUC 为 0.7149，性能稳定，位列初赛第一，领先第二名多达 62 个万分点。

2 数据分析及预处理

2.1 数据分析

本题训练集、验证集和测试集包含了 28959 个用户，训练数据由 4 部分组成，分别是危险行为、标签类型、APP 安装情况和关联关系，其中关联关系覆盖范围最广，覆盖了 28442 个用户，标签类型只覆盖了 1544 个用户，危险行为覆盖了 18735 个用户，APP 安装情况覆盖了 6966 个用户。可见，标签类型和 APP 安装情况这两类数据在数据集中比较稀疏，关联关系和危险行为覆盖了大多数用户，如何从这些数据中提取有效特征对用户建模是本题的关键，其中对网络特征的挖掘和不同数据间的关联分析是重点。

2.2 预处理

我们将拆分成多份的数据进行了合并。

3 特征工程

从用户自身和周围联系人两个角度进行特征工程。

3.1 用户自身特征

从关联关系、危险行为、标签类型、app 安装情况中提取用户自身特征。

3.1.1 关联关系

用户的关联关系是一个带权的有向无环图，首先我们利用网络表示学习算法获得每个节点（用户）的向量表示，然后利用链接分析算法 PageRank 算法和 HITS 算法计算每个节点的 PageRank 值、权威值和枢纽值，最后计算了每个节点的出度、入度和度中心性等度特征。

1) 网络表示学习

网络表示学习（Network Representation Learning，NRL），也称为图嵌入法（Graph Embedding Method，GEM）：用低维、稠密、实值的向量表示网络中的节点，这种向量含有语义关系，利于计算存储，不用再手动提特征（自适应性），且可以将异质信息投影到同一个低维空间中方便进行下游计算。

DeepWalk[1]受到词向量算法 word2vec 的启发，第一个提出了使用表示学习（或深度学习）社区的技术的网络嵌入方法。DeepWalk 通过将节点视为单词并生成短随机游走作为句子来弥补网络嵌入和单词嵌入之间的差距。然后，可以将诸如 Skip-gram 之类的神经语言模型应用于这些随机游走以获得网络嵌入。Node2Ve[2]在 DeepWalk 的基础上进行了改进，根据边的权重，计算随机游走概率矩阵，并通过超参数控制深度优先搜索和广度优先搜索。LINE[3](图 1)从一阶相似性和二阶相似性两方面对网络节点进行建模，一阶相似性表示网络节点和它周围的邻居存在相似性，如 6 和 7；二阶相似性表示网络节点和它的二阶邻居存在相似性，如 1,2,3,4。实验表明，通过一阶相似性和二阶相似性建模学习到的网络节点表示在多标签分类、链接预测等任务中取得了不错的效果。

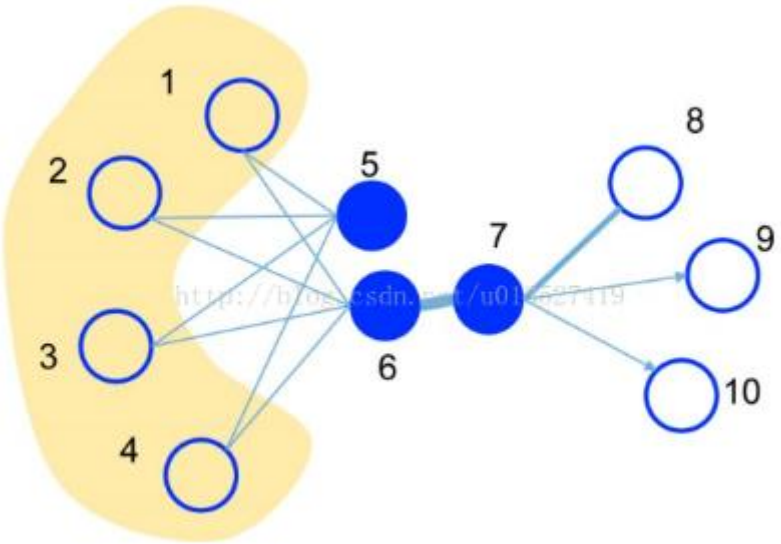

图 1 LINE

这些基于随机游走的模型都属于浅层模型，有较高的计算效率，但是由于底层网络结构的复杂性，浅层模型往往不能捕获高度非线性的网络结构。因此，出现了 SDNE[4]、GCN 等。除了网络结构信息之外，网络节点还包含很多其他属性，TADE[5]和 CENE[6]这两种模型在对节点进行嵌入的同时考虑到了文本信息。

Algorithm	Time	Micro-F1	Macro-F1
DeepWalk	271s	0.385	0.238
LINE 1st+2nd	2008s	0.398	0.235
Node2vec	2623s	0.404	0.264
GraRep	-	-	-
OpenNE(DeepWalk)	986s	0.394	0.249
OpenNE(LINE 1st+2nd)	1555s	0.390	0.253
OpenNE(node2vec)	3501s	0.405	0.275
OpenNE(GraRep)	4178s	0.393	0.230

图 2 网络表示学习方法对比

根据清华大学的总结[7](图 2)，deepwalk 在各种网络表示学习方法中具有最高的效率，鉴于本题数据量巨大，包含 3000 万节点和上亿条边，我们选择了 deepwalk 来学习网络节点

的表示。在实验中，我们训练了 128 维、192 维和 256 维的三种节点表示，deepwalk 的重要参数如所示：

表 1 deepwalk 重要的参数

No	dim	number-walks	walk-length
1	128	10	40
2	192	10	60
3	256	10	60

2) 链接分析

链接分析是指源于对 Web 结构中超链接的多维分析。当前其应用主要体现在网络信息检索、网络计量学、数据挖掘、Web 结构建模等方面。作为搜索引擎的核心技术之一，链接分析算法应用已经显现出巨大的商业价值。这里我们采用两个经典算法：PageRank 和 HITS。

PageRank 让链接来"投票"，一个页面的“得票数”由所有链向它的页面的重要性来决定，到一个页面的超链接相当于对该页投一票。一个页面的 PageRank 是由所有链向它的页面（“链入页面”）的重要性经过递归算法得到的。一个有较多链入的页面会有较高的等级，相反如果一个页面没有任何链入页面，那么它没有等级。

HTS 一个网页重要性的分析的算法。通常 HITS 算法是作用在一定范围的，比如一个以程序开发为主题网页，指向另一个以程序开发为主题的网页，则另一个网页的重要性就可能比较高，但是指向另一个购物类的网页则不一定。在限定范围之后根据网页的出度和入度建立一个矩阵，通过矩阵的迭代运算和定义收敛的阈值不断对两个向量 Authority 和 Hub 值进行更新直至收敛。

我们使用 networkx 中的相关包计算 PageRank 和 HITS。

3) 度特征

关联数据是个带权有向无环图，我们首先统计了每个节点的出度、入度。数据中包含两种可以作为权重的特征：num、weight，我们分别将 num、weight 做为权重，计算出度和入度的权重之和，然后，对度进行标准化，得到了度中心性（使用 networkx 计算）。我们还统计了用户的“好友圈”（“互粉”好友）大小。

3.1.2 危险行为

计算用户各种危险行为的总和，计算各种危险行为占危险行为总和的比例。

3.1.3 标签类型

对用户的标签类型按一级分类和二级分类分别用 CounterVectorize 处理，获得 multi-hot 表示，并统计了用户包含标签的个数。.

3.1.4 App 安装情况

数据中所有用户各类 APP 有上万种，如果使用 multi-hot 进行编码将导致超过内存限制。我们统计了 graph_filter 中包含的用户的 APP 安装情况，保留用户量最多的前 4000 种，然后使用 PCA(Principal Component Analysis)、LDA(Latent Dirichlet Allocation)和 NMF(Non-negative Matrix Factorization)对经过 CountVectorize 处理得到的 multi-hot 向量进行降维。PCA 最大限度地保持了原始数据的分布，LDA 和 NMF 是常用的文本主题模型，用户的 APP 安装情况反应了用户的兴趣，主题模型学习到的低维向量能够捕捉到这种兴趣特征。除此之外，还统计了用户安装 app 的数量，安装的 app 的用户量的总和、均值、中位数和方差。

我们使用 sklearn 中的包计算 pca、lda、nmf，维度设置为 16，其余参数使用默认值。

3.2 周围联系人特征

人类有很强的社会属性，近朱者赤近墨者黑，一个人所在的圈子对他有着潜移默化、深远长久的影响。类似于 LINE 中的一阶相似性和二阶相似性，我们从一度联系人和二度联系人两个角度提取了联系人特征。如图 3 所示，2 是 1 的一度联系人，3,4 是 1 的二度联系人。

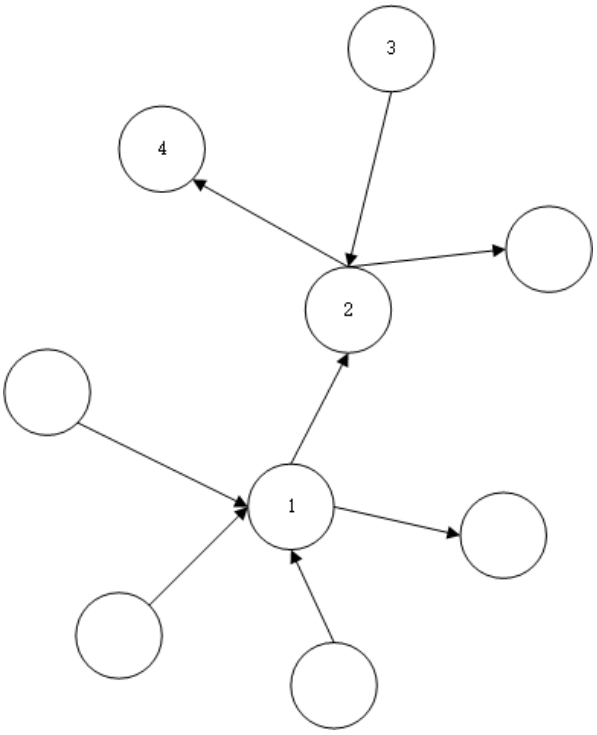

图 3 关联关系示例

3.2.1 一度联系人

如图 3 所示，每一个节点有两种边，一种是指向自己的，一种是由自己指向其他节点的，我们对这两种情况用类似的方式分开处理。首先，我们按照上一节的流程提取了和数据集（训练集、验证集、测试集）中用户有关联关系的用户的自身特征，然后根据边的权重计算加权后的均值和总和。

对于从节点 i 出发的边，按照 num 加权，节点 i 的特征F 对应的一度联系人特征F'的计算公式如下所示：

$$F_i'\_num\_mean = \frac{\sum_{j \in edge\_from_i} num_{i,j} * F_j}{\sum_{j \in edge\_from_i} num_{i,j}}$$

$$F_i'\_num\_sum = \sum_{j \in edge\_from_i} num_{i,j} * F_j$$

按 weight 加权，用类似的方式计算。

我们对 PageRank 值、HITS 提取的权威度和枢纽度、度特征、用 pca、lda、nmf 降维后的 app 向量，危险行为特征和标签类型特征都提取了一度联系人特征。

3.2.2 二度联系人

数据集中用户的二度联系人涉及范围很广，我们只计算二度联系人的链接分析产生的特征和度特征。首先对数据集中的用户的一度联系人计算一度联系人特征，然后用相同的方法计算二度联系人特征。

3.3 小结

从用户自身和周围联系人提取的特征见表 2：

表 2 特征总结

来源	记号	说明
关联关系-网络表示学习	dp_128	
	dp_192	
	dp_256	
关联关系-链接分析	pr	PageRank 值
	hits	HITS 算法计算的权威值和枢纽值
关联关系-度特征	dc	度中心性
	graph_info	出度入度的统计量
危险行为	risk	
标签类型	symbol	

app 安装情况	app_pca_16	
	app_lda_16	
	app_nmf_16	
	app_info	app 的统计信息
一度联系人	pr_1d	pagerank 值
	hits_1d	枢纽值和权威值
	dc_1d	度中心性
	graph_info_1d	
	app_graph_pca	
	app_graph_lda	
	app_graph_nmf	
	symbol_graph	
	risk_graph	
二度联系人	pr_2d	PageRank 值
	hits_2d	枢纽值和权威值
	dc_2d	度中心性
	graph_info_2d	出度入度的统计量

4 交叉验证

4.1 采用的模型

我们采用数据 XGBoost 和 lightgbm 进行预测，根据分类器和特征组合不同，设计了 6 种模型，特征组合的细节见代码。使用的分类器及特征维数见表 3：

表 3 分类器及特征维数

分类器	记号	特征维数
Lightgbm	lgb_572	572
	lgb_585	585
	lgb_628	628
	lgb_692	692
XGBoost	xgb_572	572
	xgb_652	652

4.2 参数调优

采用 hyperopt 进行参数调优，该包利用模拟退火的思想进行参数搜索，比网格搜索更好。XGBoost 的参数搜索空间见表 4。

表 4 XGBoost 参数搜索空间

max_depth	6-32
learning_rate	0.015-0.025
reg_alpha	0-50

reg_lambda	300-1000
colsample_bytree	0.6-0.9
colsample_bylevel	0.6-0.9

Lightgbm 的参数搜索空间见

表 5 Lightgbm 的参数搜索空间

boosting_type	goos
num_leaves	60-200
max_bin	300-1000
learning_rate	0.015-0.025
min_child_weight	0-20
reg_alpha	0-50
reg_lambda	100-1000
colsample_bytree	0.6-0.9

通过参数调优，每种模型我们选取交叉验证中 AUC 最高的 5 组参数，得到了 30 个模型。

5 模型融合

AUC 评价指标和预测结果的排序有关，我们对每个模型的结果进行排序，取 rank 的倒数，作为除概率外的另外一种特征，30 个基分类器共产生了 60 维特征。二层分类器我们选取了 lr 这种简单的分类器，为了避免多重共线性对 lr 的误导，我们在融合之前进行了特征选择。

5.1 特征选择

根据皮尔逊相关系数、卡方值、随机森林产生的特征重要性和 lr 的权重，对 60 维特征进行排序。经过尝试，选取前 9 个特征能在交叉验证中达到最好效果。

5.2 Stacking

Stacking 是一种常用的模型融合方法，能够增加模型稳定性，提高模型效果，原理如图 4 所示，简单地讲就是将多个基分类器对训练集的预测作为特征，在此基础上训练一个二层分类器分类器，在训练二层分类器的时候我们只用筛选出来的 9 个特征。

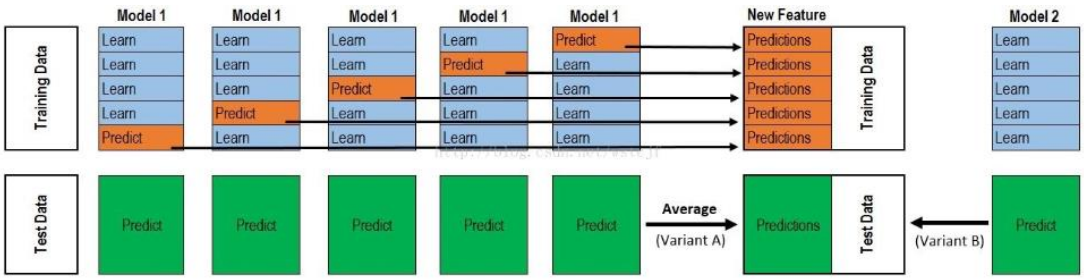

图 4 stacking 原理

6 代码说明

6.1 目录及文件说明

根目录下包含 data/、features/、models/、model_output/四个目录和一个名叫 feature_eginneering.ipynb 的文件。

6.1.1 Data/

保存训练数据、deepwalk 获得的 embedding、链接分析算法的结果放在 edge/目录下。为了减小空间，删除了原始的 embedding 文件，经过过滤的 embedding 保存在 features/目录中。

6.1.2 Features/

保存特征工程的结果。

6.1.3 Models/

保存模型代码和 stacking 代码

6.1.4 Model_output/

保存模型交叉验证的结果和对测试集的预测。

6.1.5 feature_engineering.ipynb

特征工程的代码。

6.2 硬件配置及软件版本

采用的编程语言是 python3.6，在 jupyter notebook 上进行分析、开发和调试，服务器配置及使用的主要 Python 包的版本如表 6 和表 7 所示：

表 6 服务器配置

CPU	Intel(R) Xeon(R) CPU E5-2650 @ 2.00GHz * 4 (32 核)
GPU	无
内存大小	256GB
磁盘大小	2.7TB
操作系统	Ubuntu 16.04.4 LTS (GNU/Linux 4.4.0-124-generic x86_64)

表 7 主要 python 包及版本

包名	版本
Pandas	0.23.4
Numpy	1.11.2
Sklearn	
Networkx	1.11
Lightgbm	2.1.2
Xgboost	0.80
hyperopt	0.1.1
category_encoders	1.2.6

6.3 代码运行说明

首先运行 feature_engineering.ipynb 中的 build_graph，然后用 deepwalk 进行网络表示学习，将结果保存到 data/目录下，再运行完 feature_engineering.ipynb 中的剩余代码，完成特征工程。

通过参数调优得到的参数记录在 models 下各个模型文件中，直接运行 xgb 和 lgb 的 6 个文件即可。最后运行 stacking_test_random.ipynb 进行模型融合，得到最终结果，最终结果保存在 model_output/random/stacking/sub_test_random.txt 中。

由于网络表示学习、链接分析算法需要花费数天时间，我们将题目需要的网络节点表示、链接分析结果和特征工程提取到的特征也提交了，验证无需重新跑这部分代码。6 种模型的结果也提交了，可以直接 stacking 进行验证。

代码及特征上传到了百度云，如果有问题，请及时和我们联系，谢谢！

链接：https://pan.baidu.com/s/1WWoNMT645VMOaLv3MqUUbA 密码：9ced

7 总结

7.1 优势

1. 从用户本身和周围联系人两个角度进行建模，思路清晰。
2. 采用了多种维度的网络节点表示，通过融合提升了模型性能。

3. 注重单模型本地 cv 和线上成绩，有效避免了过拟合。

7.2 不足

1. 对网络特征的挖掘还有欠缺，网络表示学习方面，node2vec 的代码经过修改应该也可以适应这种大规模网络，没有使用分布式计算平台做社区发现、紧密中性度等较为复杂的图算法。
2. 尝试的模型较为单一。

参考文献

[1] Perozzi B, Al-Rfou R, Skiena S. DeepWalk:online learning of social representations[C]// Acm Sigkdd International Conference on Knowledge Discovery & Data Mining. ACM, 2014:701-710.

[2] Grover A, Leskovec J. node2vec: Scalable Feature Learning for Networks[C]// Acm Sigkdd International Conference on Knowledge Discovery & Data Mining. ACM, 2016:855-864.

[3] Tang J, Qu M, Wang M, et al. LINE:Large-scale Information Network Embedding[C]// International World Wide Web Conferences Steering Committee, 2015:1067-1077.

[4] Wang D, Cui P, Zhu W. Structural Deep Network Embedding[C]// ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, 2016:1225-1234.

[5] Yang C, Zhao D, Zhao D, et al. Network representation learning with rich text information[C]// International Conference on Artificial Intelligence. AAAI Press, 2015:2111-2117.

[6] Sun X, Guo J, Ding X, et al. A General Framework for Content-enhanced Network Representation Learning[J]. 2016.

[7] https://github.com/thunlp/OpenNE