第*15*章

## 第十五章 实战案例：2018腾讯社交广告算法大赛--相似人群拓展

本章将以2018年第二届腾讯社交广告算法大赛竞赛为例，进行计算广告相关的实战讲解，端到端讲解完整的实战流程与注意事项。本章将主要分为五个部分来进行讲解，分别是赛题理解、探索性分析、特征工程、模型选择、赛题总结，这是本书所有实战案例章节的组织结构，也是一场竞赛的重要组成流程。相信在本书的指引下，读者能快速地熟悉竞赛流程，并进行实战。

### 15.1 赛题理解

所谓磨刀不误砍柴工，做比赛前应对赛题相关信息进行充分了解，理解背后的需求，进而达到正确审题的目的。本次赛题的题目为基于计算广告问题的相似人群拓展，参赛者的幸运之处在于主办方拥有国内最大的社交平台，无论是其数据的高质量还是举办比赛的专业度都无可挑剔，因此以下部分多来自于腾讯官方给出的赛题说明。

#### 15.1.1 赛题背景

基于社交关系的广告（即社交广告）已成为互联网广告行业中发展最为迅速的广告种类之一。腾讯社交广告平台，依托于腾讯丰富的社交产品，植根于腾讯海量的社交数据，借助强大的数据分析、机器学习和云计算能力打造出了一个服务于千万商家和亿万用户的商业广告平台。腾讯社交广告一直致力于提供精准高效的广告解决方案，而复杂的社交场景，多样的广告形态，以及庞大的用户数据，给实现这一目标带来了不小的挑战。为攻克这些挑战，腾讯社交广告也在不断地寻找出更为优秀的数据挖掘和机器学习算法。

本次算法大赛的题目源于腾讯社交广告业务中的一个真实的广告产品——相似人群拓展（Lookalike）。该产品的目的是基于广告主提供的目标人群，从海量的人群中找出和目标人群相似的其他人群。在实际广告业务应用场景中，Lookalike 能基于广告主已有的消费者，找出和已有消费者相似的潜在消费者，以此有效帮助广告主挖掘新客、拓展业务。目前，腾讯社交广告 Lookalike 相似人群拓展产品以广告主提供的第一方数据及广告投放效果数据（即后文提到的种子包人群）为基础，结合腾讯丰富的数据标签能力，透过深度神经网络挖掘，实现了可在线实时为多个广告主同时拓展具有相似特征的高质潜客的能力。通过本次大赛，腾讯社交广告旨在挑选出更为优秀 Lookalike 算法以及遴选出杰出的社交广告算法达人。

#### 15.1.2 赛题数据

本次竞赛的比赛数据（脱敏后）抽取的时间范围是某连续 30 天的数据。总体而言，数据可以分为：训练集数据文件、测试集数据文件、用户特征文件以及种子包对应的广告特征文件四部分。

训练集数据文件 train.csv 每行代表一个训练样本，各字段之间由逗号分隔，格式为：“aid,uid,label”。其中，aid 唯一标识一个广告，uid 唯一标识一个用户。样本 label 的取值为 +1 或 -1，其中 +1 表示种子用户，-1 表示非种子用户。为简化问题，一个种子包仅对应一个广告 aid，两者为一一对应的关系。

测试集数据文件 test.csv每行代表一个训练样本，各字段之间由逗号分隔，格式为：“aid,uid”。字段含义同训练集。

用户特征文件 userFeature.data 每 行 代 表 一 个 用 户 的 特 征 数 据， 格 式 为：“uid|features”，uid 和 features 用竖线“|”分隔。其中 feature 采用 vowpal Wabbit

（https://github.com/JohnLangford/vowpal\_wabbit）格式：

“feature\_group1|feature\_group2|feature\_group3|...”。 每 个 feature\_group代表一个特征组，多个特征组之间也以竖线“|”分隔。一个特征组若包括多个值则以空格分隔，格式为：“feature\_group\_name fea\_name1 fea\_name2 …”，其中 fea\_name 采用数据编号的格式。用户特征详情见【用户特征说明】。

广告特征文件 adFeature.csv 格式为：“aid,advertiserId,campaignId,creativeId,creativeSize,

adCategoryId,productId,productType”。其中，aid 唯一标识一广告，其余字段为广告特征，各字段之间由逗号分隔。广告特征详情见【广告特征说明】。出于数据安全的考虑，我们对 uid、aid、用户特征、广告特征按照如下方式进行加密处理：

• uid：对每个用户 ID 进行 1 到 N 的随机化编号，生成一个不重复的加密 uid，N 为用户总数目（假设用户数为 100w，将所有用户随机打散排列，将其序号作为 uid，取值范围是 [1, 100w]）；

• aid：参考 uid 的加密方式，生成加密后的 aid；

• 用户特征：参考 uid 的加密方式，生成加密后的 fea\_name；

• 广告特征：参考 uid 的加密方式，生成加密后的各字段

用户特征包含以下特征组（feature\_group\_name），如果具体特征取值未知，均

以 0 表示：

• 年龄（age）：分段表示，每个序号表示一个年龄分段

• 性别（gender）：男 / 女

• 婚姻状况（marriageStatus）：单身 / 已婚等状态（多个状态可共存）

##### **用户特征说明**

• 学历（education）：博士 / 硕士 / 本科 / 高中 / 初中 / 小学

• 消费能力（consumptionAbility）：高 / 低

• 地理位置（LBS）：每个序号代表一个地理位置

• 兴 趣 类 目（interest）： 由 不 同 数 据 源 挖 掘 得 到 的 5 个 特 征 组， 分 别 以

interest1, interest2, interest3, interest4, interest5 表示，每个兴趣特征组包

含若干个兴趣 ID

• 关键词（keyword）：较兴趣类目更细粒度地表示用户喜好，由不同数据源挖

掘得到的 3 个特征组，分别以 kw1, kw2, kw3 表示，每个关键词特征组包含若

干用户感兴趣的关键词

• 主题（topic）：使用 LDA 挖掘的用户喜好主题，由不同数据源挖掘得到的 3

个特征组，分别以 topic1, topic2, topic3 表示

• APP 近期安装行为（appIdInstall）：63 天内安装的 APP，每个 APP 表示为一

个唯一的 ID

• APP 活跃（appIdAction）：用户使用的活跃 APP

• 上网连接类型（ct）：WIFI/2G/3G/4G

• 操作系统（os）：Android/IOS，不区分版本号

• 移动运营商（carrier）：移动运营商，移动 / 联通 / 电信 / 其他

• 有房（house）：是否有房

##### **广告特征说明**

• 广告 ID（aid）：广告是指广告主创建的广告创意（或称广告素材）及广告展

示相关设置，包含广告的基本信息（广告名称、投放时间等）、广告的推广目标、

投放平台、投放的广告规格、所投放的广告创意、广告的受众（即广告的定向

设置）以及广告出价等信息

• 广告主 ID（advertiserId）：账户结构分为四级：账户——推广计划——广告——

素材，账户和广告主是一一对应关系

• 推广计划 ID（campaignId）：推广计划是广告的集合（类似电脑文件夹功能），

广告主可以将推广平台、预算限额、是否匀速投放等条件相同的广告放在同一

个推广计划中，方便管理

• 素材 ID（creativeId）：展示给用户直接看到的广告内容，一条广告下可以有

多组素材

• 素材大小（creativeSize）：素材大小 ID，标识广告素材不同大小

• 广告类目（adCategoryId）：广告分类 ID，使用广告分类体系

• 商品 ID（productId）：推广的商品 ID，系统中用 product id 来标识

• 商品类型（productType）：广告投放目标对应的商品类型（如京东 -- 商品、

app-- 下载）

#### 15.1.3 赛题任务

相似人群拓展（Lookalike）基于广告主提供的一个种子人群（又称为种子包），自动计算出与之相似的人群（称为扩展人群）。本题目将为参赛选手提供几百个种子人群、海量候选人群对应的用户特征，以及种子人群对应的广告特征。出于业务数据安全保证的考虑，所有数据均为脱敏处理后的数据。整个数据集分为训练集和测试集。训练集中标定了人群中属于种子包的用户与不属于种子包的用户（即正负样本）。测试集将检测参赛选手的算法能否准确标定测试集中的用户是否属于相应的种子包。训练集和测试集所对应的种子包完全一致。

为了检验参赛选手的算法能否很好地理解用户以及种子人群，本次大赛要求参赛者提交的结果中，提供测试集中各种子包候选用户属于该种子包的得分（得分越高说明候选用户是某个包潜在的扩展用户的可能性越大）。大赛官网的后台算法将自动计算提交结果的得分及排名。详情可参看【评估方式】【提交方式】。

初赛和复赛所提供的种子包除量级有所不同外，其他的设置均相同。

#### 15.1.4 评价指标

对于扩展后的相似用户，如果在广告投放上有相关的效果行为（点击或者转化），则认为是正例；如果不产生效果行为，则认为是负例。每个待评估的种子包会提供如下信息：种子包对应的广告 aid 及其特征，以及对应的候选用户集合（uid 及其特征）。选手需要为每个种子包计算测试集中用户的得分，比赛会据此计算每个种子包的 AUC 指标，AUCi 表示第 i 个包的 AUC 值，并以所有待评估的 m 个种子包的平均 AUC 作为最终的评估指标：



#### 15.1.5 赛题FAQ

Q：赛题的本质任务是什么？

A：赛题的本质是通过过往的广告推送与点击记录，对未来的广告推送进行精准用户匹配，提高用户对推送广告的点击率，进而提高转化率，为广告主客户带来商业价值，并收取广告营销费用。

Q：互联网广告方面的几个评价指标有什么关联？

A：首先第一个指标是曝光量，指广告在用户面前曝光的次数，也即广告给多少用户进行了推送展示；其次便是用户看到广告后的点击率，即点击进入广告页面的用户比例；最后若用户看到了广告并购买了相应的商品，这部分比例则称为是转化率。由此可知曝光量、点击量与转化量是一个倒金字塔结构，即依次递减。当然，曝光广告除了给广告主客户带来直接的转化之外，变相也是对客户的品牌与知名度营销。

### 15.2 探索性分析

本节将对竞赛提供的可用信息与数据进行分析解读，探索发现可能的建模思路。通常来说，在内存允许的情况下，参赛者一般可以借助jupyter notebook和pandas、numpy等常见的python三方开源包进行探索性分析，这方面依据分析需要有不同的函数可以借助，其中pandas包常用的函数有read\_csv()、head()、describe()、value\_counts()、plot()、shape等。

#### 15.2.1 竞赛公开数据集组成

以初赛数据为例，可用的数据集分别有：

-train.csv 训练集

-test1.csv 测试集

-test1\_truth.csv 测试集标签

-adFeature.data 广告基本属性

-userFeature.csv 用户基本信息

#### 15.2.2 测试集与训练集

**字段取值与含义**

**训练集**

|  |  |  |  |
| --- | --- | --- | --- |
| train.csv (8798814, 3) | aid | uid | label |
| 1781 | 55796870 | 1 |
| 1596 | 7379545 | -1 |
| 1749 | 15539087 | -1 |
| 411 | 17202792 | -1 |
| 692 | 72207685 | -1 |
| ... | ... | ... |

测试集

|  |  |  |
| --- | --- | --- |
| test1.csv (2265989, 2) | aid | uid |
| 2118 | 82020764 |
| 1930 | 27498248 |
| 916 | 31121616 |
| 2197 | 53306 |
| 411 | 15707393 |
| ... | ... |

测试集标签

|  |  |  |  |
| --- | --- | --- | --- |
| test1\_truth.csv (2265989, 3) | aid | uid | label |
| 2048 | 31472425 | 1 |
| 302 | 19640598 | -1 |
| 2031 | 73688375 | -1 |
| 173 | 31556480 | -1 |
| 411 | 66501961 | -1 |
| ... | ... | ... |

可以看出，训练集与测试集中只给出了ID列与标签列，这部分腾讯公开提供的数据集中也给出了测试集的真实标签，参赛者需要明确这是一个双主键的用户与广告匹配，因此可以适当查看训练集与测试集的aid与uid的重叠情况进而判断出训练集分布与测试集的分布差异。

**分布差异**

首先确认训练集与测试集中均无缺失值，且训练集正样本的比例为4.8%，这应该是经过一定的采样，在实际业务中的点击率很难达到这个水平。

然后分别与合并统计训练集与测试集中aid与uid的去重唯一取值数如下

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | train\_  nunique | test\_  nunique | all\_  nunique | duplicates | inbag\_  ratio |
| uid | 7883466 | 2195951 | 9686953 | 392464 | 18% |
| aid | 173 | 173 | 173 | 173 | 100% |

可以看出，测试集的用户只有不到18%出现在训练集中，而aid与训练集中出现的全部一样。这也比较符合商业逻辑，即在广告投放种类短时间内保持一致的情况下，通过现有投放的点击效果来对未推送过的用户进行概率匹配预测，进而提升点击率，带来商业收益。

检查完单主键的取值差异之后，需确认双主键的取值也唯一，即aid与uid的组合是唯一的，只有一个确定的标签取值，代码验证如下：

train\_nunique = train[['uid', 'aid']].drop\_duplicates().shape[0]

test1\_nunique = test1[['uid', 'aid']].drop\_duplicates().shape[0]

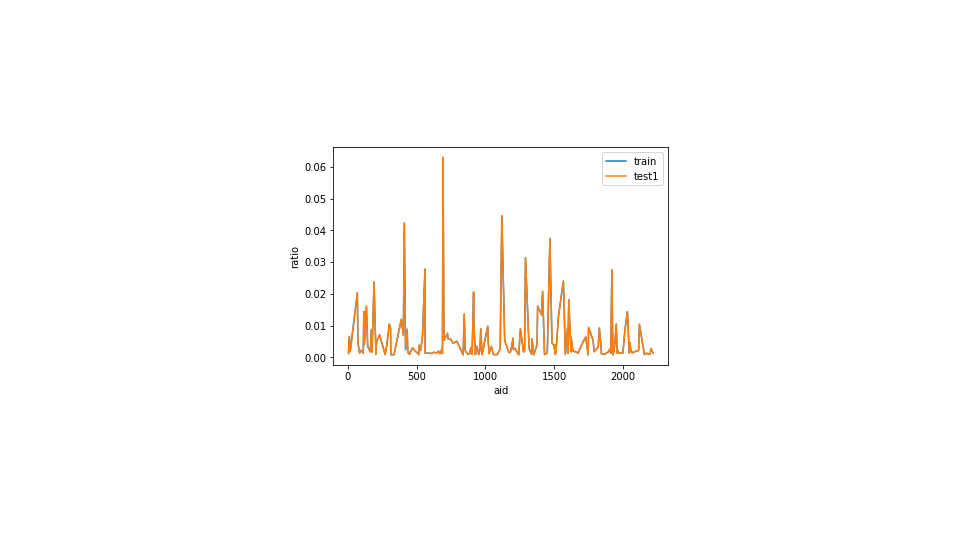
all\_nunique = test1[['uid', 'aid']].append(train[['uid', 'aid']]).drop\_duplicates().shape[0]

assert train\_nunique == train.shape[0]

assert test1\_nunique == test1.shape[0]

assert train\_nunique + test1\_nunique == all\_nunique

最后，根据上述分析，还缺一点逻辑闭环就是训练集与测试集中投放的广告aid分布是否相同，验证结果呈现如下：



由此可见，训练集与测试集中的广告分布比例基本一致。因此，重点需要考察的是不同用户对同一广告的感兴趣程度，或者说需要参赛者寻找出同一广告的用户群特点，继而通过已有的点击数据找出更多地可能对该广告感兴趣的用户，即是本次竞赛的主题：Lookalike相似人群拓展。

#### 15.2.3 广告属性

同样使用pandas.DataFrame().head()方法展示表格如下：

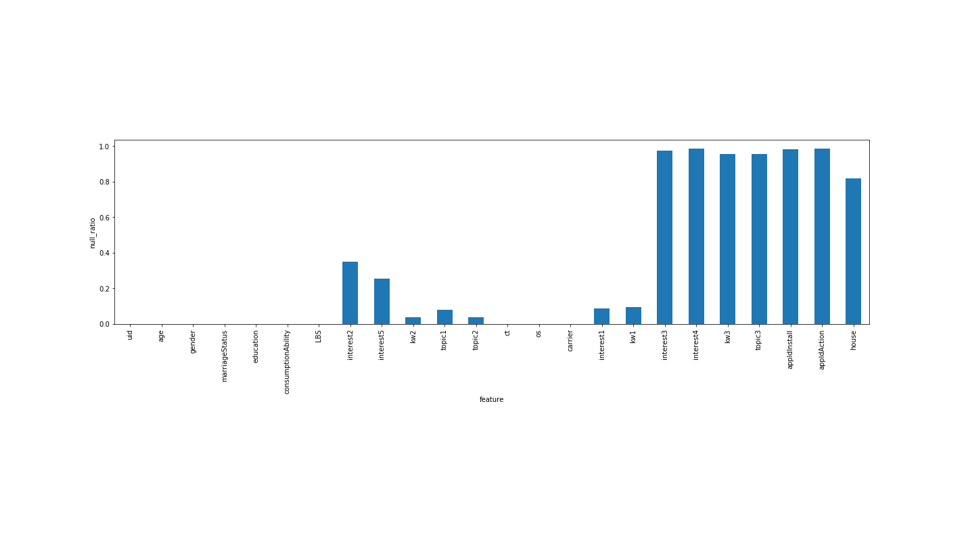
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| aid | advertiserId | campaignId | creativeId | creativeSize | adCategoryId | productId | productType |
| 2169 | 16770 | 38402 | 43877 | 35 | 89 | 9760 | 9 |
| 411 | 9106 | 163120 | 220179 | 79 | 21 | 0 | 4 |
| 894 | 452 | 38391 | 43862 | 35 | 10 | 12193 | 11 |
| 450 | 45705 | 352827 | 565415 | 42 | 67 | 0 | 4 |
| 313 | 243 | 531344 | 979528 | 22 | 27 | 113 | 9 |
| ... | ... | ... | ... | ... | ... | ... | ... |

可以看到数据全部进行了脱敏处理，但并不妨碍参赛者对字段含义的理解，在15.1.2里面列出了详细的广告特征说明，参赛者可结合说明自行查看数据。

#### 15.2.4 用户信息

由于数据文件格式为.data，不利于参赛者直接进行分析统计，因此先将其转换为csv格式的文件，此处转换代码来自开源：

<https://github.com/liupengsay/2018-Tencent-social-advertising-algorithm-contest>



将原始数据转换成pandas.dataframe格式后，可以方便地进行分析，由于字段过多，这里只截图部分字段进行展示如下，

其中除了用户标识uid之外，其他字段均为用户属性，其中又分为如age、gender、marriageStatus、education、consumptionAbility、LBS等这类每个用户只有一个取值的单变量属性，和interest2、interest5、kw2这类每个用户会有多个取值的多变量属性。关于多变量属性的处理会借助自然语言处理相关的算法，这方面将在后文特征工程中进行着重讲解。

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| uid | age | gender | marriage Status | education | consumption Ability | LBS | interest2 | interest5 | kw2 |
| 72068206 | 4 | 2 | 10 | 1 | 1 | 317 | 79 6 | 77 53 109  30 6 59 | 15215 80808 114283  71854 34525 |
| 44661871 | 5 | 1 | 11 | 7 | 1 | 458 |  | 77 52 100  72 131 37  116 4 79  71 109 8  69 41 6  46 62 121  74 59 25  129 21 | 15571 92783 34154  33457 31671 |
| 3036658 | 3 | 1 | 11 | 7 | 1 | 682 | 47 22 58  24 79 73  9 46 32  70 20 6  33 50 49  30 16 2  61 14 | 100 72 80  131 37 116 108 79 29  8 113 6  132 42 46  59 129 103 | 11395 79112 82720  87384 56195 |
| 77640094 | 5 | 2 | 6 13 | 7 | 2 | 809 | 79 46 6  18 29 | 72 92 37  116 47 78  71 8 30  50 6 111  27 75 46  38 40 52  131 61 41  121 59 24  25 10 129 | 846 95597 33055  41919 81422 |
| 68081558 | 4 | 1 | 10 | 2 | 0 | 774 |  |  | 9429 114069 97540  77212 114164 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |

#### 15.2.5 训练集与测试集特征拼接

经过训练集、测试集、广告属性、用户信息的熟悉之后，参赛者可以明确了解到几个表格之间的关联关系，即是以训练集与测试集的ID列为基础，进行广告属性与用户信息的关联后，形成常规意义上的带有ID列与标签的特征宽表，除了多变量特征可能还需要额外处理之外，其余特征可直接用于建模。

由于原始数据相对较大，简单起见，也是对于部分刚入门的参赛者来说可能手上还没有足够的算力资源，为了方便参赛者快速理解并且跑通demo起见，本书在这一环节对训练集与测试集进行了1%的随机采样，使得大数据问题转换为小数据，能够快速进行相关数据探索性分析、特征工程以及模型搭建的快速验证。方案确定后，若有足够的资源就可进行全量数据建模。

train = train.sample(frac=0.01, random\_state=2020).reset\_index(drop=True)

test1 = test1.sample(frac=0.01, random\_state=2020).reset\_index(drop=True)

同时，为了建模方便，需要将样本标签label中代表负样本的-1用0代替，同时也记录测试集的真实标签，方便后续进行建模验证对比。最后根据特征的单变量多变量属性进行特征类别的区分，识别方法如下，

cols = train.columns.tolist()

cols.sort()

se = train[cols].dtypes

text\_features = se[se=='object'].index.tolist()

discrete\_features = se[se!='object'].index.tolist()

discrete\_features.remove('aid')

discrete\_features.remove('uid')

discrete\_features.remove('label')

由此可得数据集得多变量特征text\_features共计有16个，分别为['appIdAction', 'appIdInstall', 'ct', 'interest1', 'interest2', 'interest3', 'interest4', 'interest5', 'kw1', 'kw2', 'kw3', 'marriageStatus', 'os', 'topic1', 'topic2', 'topic3']，而单变量特征discrete\_features则有14个，分别为['LBS', 'adCategoryId', 'advertiserId', 'age', 'campaignId', 'carrier', 'consumptionAbility', 'creativeId', 'creativeSize', 'education', 'gender', 'house', 'productId', 'productType']

#### 15.2.6 基本建模思路

通过简单的数据探索性分析以及表格的拼接处理，参赛者可以感知到这个数据结构十分清晰，就两类特征，即单变量离散特征discrete\_features与多变量文本特征text\_features，因此可以本章将考虑引入可以直接支持text\_features的catboost进行建模。

### 15.3 特征工程

本节将在数据探索性分析的基础之上，进行一些特征的提取，由于本次竞赛的数据特点极具代表性，即是一种除了ID列和标签以外，其余列均为特征列，且这里的特征列均为离散列，包含了单变量与多变量两种。这种数据的组织形式与第八章Elo实战案例是两种典型的场景，Elo里面的原始数据是用户的一些行为记录，建模之前需要进行相应的特征设计与提取。当然并不是说本章写的实战案例就不需要特征设计与提取了，只是在这里的特征工程会和Elo里面有一些不同，本节将以此为例，介绍另一套常用的特征设计与提取方案，其中通用特征与业务特征均是对单变量字段的信息提取处理，而文本特征则是针对多变量字段。

#### 15.3.1 通用特征

直观上来讲，对于单变量的离散字段，一般的普通模型如LR、RF、GDBT等在进行训练的时候没办法进行区分和处理，因此需要对这类离散字段进行变换处理，使得其能够用具有大小意义的连续列进行表征，继而采用模型进行量化区分学习。本节将介绍三类常见特征的含义以及提取方式。

##### **Count**

首先有一种简单的计数特征，这主要是为了衡量某个单变量离散字段的出现频次，即代表某个样本的属性是属于大众还是偏小众，通常只需要使用pandas.series的value\_counts()方法，这与后面文本特征所需要介绍用到的多变量字段的CountVectorize相对应，这类特征在长尾分布等类似的数据取值分布中体现最为明显。体现在本次竞赛当中，这种Count可以称之为Exposure曝光量，可以是单个字段的曝光量，也可以是多个字段组合的高阶曝光量，下面以字段为例，给出Count特征的输入输出。

输入：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| uid | age | gender | education | consumptionAbility |
| 10971433 | 1 | 1 | 1 | 2 |
| 7657617 | 2 | 2 | 6 | 1 |
| 39031487 | 1 | 2 | 7 | 0 |
| 3671870 | 5 | 2 | 1 | 1 |
| 81984219 | 1 | 2 | 6 | 1 |
| 1014652 | 1 | 1 | 3 | 1 |
| 41398762 | 5 | 1 | 7 | 1 |
| 55309167 | 4 | 1 | 7 | 1 |
| 71280113 | 5 | 2 | 7 | 1 |
| 16224045 | 4 | 1 | 7 | 1 |

输出：

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| exposure\_age | exposure \_gender | exposure \_education | exposure \_consumptionAbility | exposure \_age\_and \_gender | exposure \_age\_and \_education | exposure\_age \_and\_ consumptionAbility | exposure \_gender \_and\_ education | exposure\_gender \_and\_ consumptionAbility | exposure \_education \_and\_ consumptionAbility |
| 4 | 5 | 2 | 1 | 2 | 1 | 1 | 1 | 1 | 1 |
| 1 | 5 | 2 | 8 | 1 | 1 | 1 | 2 | 4 | 2 |
| 4 | 5 | 5 | 1 | 2 | 1 | 1 | 2 | 1 | 1 |
| 3 | 5 | 2 | 8 | 2 | 1 | 3 | 1 | 4 | 1 |
| 4 | 5 | 2 | 8 | 2 | 1 | 2 | 2 | 4 | 2 |
| 4 | 5 | 1 | 8 | 2 | 1 | 2 | 1 | 4 | 1 |
| 3 | 5 | 5 | 8 | 1 | 2 | 3 | 3 | 4 | 4 |
| 2 | 5 | 5 | 8 | 2 | 2 | 2 | 3 | 4 | 4 |
| 3 | 5 | 5 | 8 | 2 | 2 | 3 | 2 | 4 | 4 |
| 2 | 5 | 5 | 8 | 2 | 2 | 2 | 3 | 4 | 4 |

以exposure\_age字段为例，年龄age的取值样例有1、2、4、5，其对应的数量分别为4、1、2、3，其余字段含义相似，exposure\_age\_and\_gender则是age与gender的组合计数，单变量的曝光量特征计算只需要一个字段，二阶以上的曝光量特征计算则需要用到两个字段。这样，原本不具有大小关系的取值被映射成了数量值，直观上也能代表用户在比如age这一维度上是属于大众还是小众，一定程度上可以反映用户的年龄区分度参赛者应该可以想到，在这基础上甚至可以做3阶及以上的特征，当然这就容易带来维度爆炸的问题，这也是在自然语言处理领域有关文本特征提取的N-Gram算法。

##### **Nunique**

第二类则是唯一取值数nunique特征，指两个单变量离散字段之间，唯一值的个数，两个单变量之间可以有包含关系也可以互相独立，通常来说，在二者具有包含关系且不同分支之间的唯一值个数差异较大的时候建模效果比较明显。举例来说，若用户的地理位置属性中含有脱敏的城市ID与地铁线路信息，那么一般来说地铁线路多的城市经济更繁华，人口可能也更多，这就可以给用户附加额外的地理属性带来的挖掘信息。

下面以训练集的['age', 'education', 'adCategoryId', 'advertiserId']这几个字段给出输入输出的样表，并进行相关解读。

输入：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| uid | age | education | adCategoryId | advertiserId |
| 10971433 | 1 | 1 | 21 | 10055 |
| 7657617 | 2 | 6 | 81 | 2509 |
| 39031487 | 1 | 7 | 142 | 8203 |
| 3671870 | 5 | 1 | 27 | 327 |
| 81984219 | 1 | 6 | 142 | 5459 |
| 1014652 | 1 | 3 | 51 | 25485 |
| 41398762 | 5 | 7 | 21 | 44008 |
| 55309167 | 4 | 7 | 24 | 6946 |
| 71280113 | 5 | 7 | 59 | 8350 |
| 16224045 | 4 | 7 | 4 | 158679 |

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| nunique \_education \_in\_age | nunique\_ adCategoryId \_in\_age | nunique\_ advertiserId \_in\_age | nunique\_ age\_in\_ education | nunique\_ adCategoryId \_in\_education | nunique\_ advertiserId \_in\_education | nunique\_ age\_in\_ adCategoryId | nunique\_ education\_in \_adCategoryId | nunique\_ advertiserId\_ in\_adCategoryId | nunique\_ age\_in\_ advertiserId | nunique\_ education\_ in\_advertiserId | nunique\_ adCategoryId \_in\_advertiserId |
| 4 | 3 | 4 | 2 | 2 | 2 | 2 | 2 | 2 | 1 | 1 | 1 |
| 1 | 1 | 1 | 2 | 2 | 2 | 1 | 1 | 1 | 1 | 1 | 1 |
| 4 | 3 | 4 | 3 | 5 | 5 | 1 | 2 | 2 | 1 | 1 | 1 |
| 2 | 3 | 3 | 2 | 2 | 2 | 1 | 1 | 1 | 1 | 1 | 1 |
| 4 | 3 | 4 | 2 | 2 | 2 | 1 | 2 | 2 | 1 | 1 | 1 |
| 4 | 3 | 4 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 2 | 3 | 3 | 3 | 5 | 5 | 2 | 2 | 2 | 1 | 1 | 1 |
| 1 | 2 | 2 | 3 | 5 | 5 | 1 | 1 | 1 | 1 | 1 | 1 |
| 2 | 3 | 3 | 3 | 5 | 5 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 2 | 2 | 3 | 5 | 5 | 1 | 1 | 1 | 1 | 1 | 1 |

输出：

以nunique\_education\_in\_age为例，age为1的用户学历有四种，分别是1、7、6、3，而age为4的用户学历只有7这1种，某种程度上反映了不同age的学习分布差异。这是在education层面对age进行的表征，反过来同样可以从age层面对education进行表征。然而这部分信息同样只是一阶的表达，这类特征的包含项与被包含项都可以做更高阶的延伸。

##### **Ratio**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ratio\_exposure\_age\_in\_education | ratio\_exposure\_education\_in\_age | ratio\_exposure\_age\_in\_adCategoryId | ratio\_exposure\_adCategoryId\_in\_age | ratio\_exposure\_age\_in\_advertiserId | ratio\_exposure\_advertiserId\_in\_age | ratio\_exposure\_education\_in\_adCategoryId | ratio\_exposure\_adCategoryId\_in\_education | ratio\_exposure\_education\_in\_advertiserId | ratio\_exposure\_advertiserId\_in\_education | ratio\_exposure\_adCategoryId\_in\_advertiserId | ratio\_exposure\_advertiserId\_in\_adCategoryId |
| 0.5 | 0.25 | 0.5 | 0.25 | 1 | 0.25 | 0.5 | 0.5 | 1 | 0.5 | 1 | 0.5 |
| 0.5 | 1 | 1 | 1 | 1 | 1 | 1 | 0.5 | 1 | 0.5 | 1 | 1 |
| 0.2 | 0.25 | 1 | 0.5 | 1 | 0.25 | 0.5 | 0.2 | 1 | 0.2 | 1 | 0.5 |
| 0.5 | 0.333333333 | 1 | 0.333333333 | 1 | 0.333333333 | 1 | 0.5 | 1 | 0.5 | 1 | 1 |
| 0.5 | 0.25 | 1 | 0.5 | 1 | 0.25 | 0.5 | 0.5 | 1 | 0.5 | 1 | 0.5 |
| 1 | 0.25 | 1 | 0.25 | 1 | 0.25 | 1 | 1 | 1 | 1 | 1 | 1 |
| 0.4 | 0.666666667 | 0.5 | 0.333333333 | 1 | 0.333333333 | 0.5 | 0.2 | 1 | 0.2 | 1 | 0.5 |
| 0.4 | 1 | 1 | 0.5 | 1 | 0.5 | 1 | 0.2 | 1 | 0.2 | 1 | 1 |
| 0.4 | 0.666666667 | 1 | 0.333333333 | 1 | 0.333333333 | 1 | 0.2 | 1 | 0.2 | 1 | 1 |
| 0.4 | 1 | 1 | 0.5 | 1 | 0.5 | 1 | 0.2 | 1 | 0.2 | 1 | 1 |

结合前面提到Count部分的二阶特征的两个特征之间的交互可以做Ratio特征，它Count计数与Nunique唯一值两种取值均为整数不同，计算所得到的值为0到1之间的小数。与前面两部分的输入类似，下面给出输出示例：

以ratio\_exposure\_age\_in\_education为例，表示education为1的用户之中，age为1和5的各自比例为0.5，而education为6的用户之中，age为1和2的各自比例同样为0.5。

#### 15.3.2 业务特征

在上一节介绍了有关单变量离散字段包括Count、Nunique和Ratio在内的三种通用特征，接下来结合本次竞赛介绍另一种需要用到标签的统计特征，在分类模型里面都可能会用到，实际是想查看每个离散字段的不同取值的标签分布比例，投射到本次竞赛的场景，便是点击率。

##### **CTR**

在介绍点击率特征的计算之前，首先要明确过拟合over-fitting以及泄露leak的两个概念，过拟合是指模型在训练的时候对训练集过度拟合，导致其泛化性能较差，在测试集与训练集的分布特别是特征与标签的联合分布差异较大时，泛化性能表现极差，而泄露是指模型训练时在特征里面掺杂了标签的信息，导致标签在某种程度上化为了特征的一部分，因此模型在学习时效果极佳，然而同样的问题是测试集的标签未知且可能存在分布差异，同样会导致模型的泛化性能不佳甚至极差，也会导致过拟合。因此，在使用标签进行相关特征的提取和处理时要极度小心，既要加强特征对标签的表达，又不能过度表达导致标签信息泄露和过拟合。

为了一定程度上避免标签泄露，可以借鉴五折的思想进行交叉点击率统计，这样每个样本的点击率特征就未使用其标签的信息。具体算法步骤如下：

1、将训练集随机分成N等份

2、对上述的每一份训练集，对应的点击率特征由其余的N-1份训练进行统计映射，同时也得到一个测试集的点击率特征映射结果。

3、步骤2进行完后，可以得到整个训练集对应的点击率特征，同时对N次不同的N-1份训练集对测试集的点击率特征映射结果取均值，就能得到测试集对应的点击率特征。

点击率特征计算的输出示例：

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ctr\_of\_age | ctr\_of\_education | ctr\_of\_adCategoryId | ctr\_of\_advertiserId | ctr\_of\_age\_and\_education | ctr\_of\_age\_and\_adCategoryId | ctr\_of\_age\_and\_advertiserId | ctr\_of\_education\_and\_adCategoryId | ctr\_of\_education\_and\_advertiserId | ctr\_of\_adCategoryId\_and\_advertiserId |
| 0.042529989 | 0.053007136 | 0.035608309 | 0.060606061 | 0.06122449 | 0.026490066 | 0 | 0.056737589 | 0.090909091 | 0 |
| 0.06519337 | 0.040662651 | 0.166666667 | 0.166666667 | 0.042635659 | 0.333333333 | 0.333333333 | -1 | -1 | 0.166666667 |
| 0.043840178 | 0.040197461 | 0.068027211 | 0.054901961 | 0.040522876 | 0.054054054 | 0.054794521 | 0.04109589 | 0.044776119 | 0.081632653 |
| 0.038692098 | 0.053069719 | 0.042576419 | 0.039325843 | 0.034246575 | 0.055555556 | 0.047619048 | 0.033898305 | 0 | 0.030864198 |
| 0.044276458 | 0.044182622 | 0.079470199 | 0.044117647 | 0.040540541 | 0.05 | 0.083333333 | 0.285714286 | 0.125 | 0.06122449 |
| 0.042529989 | 0.063131313 | 0.040650407 | 0.026200873 | 0.025641026 | 0.027522936 | 0 | 0 | 0 | 0.026200873 |
| 0.038897893 | 0.041592606 | 0.040029652 | 0.059322034 | 0.046136101 | 0.025787966 | 0.049382716 | 0.030487805 | 0.05785124 | 0.059322034 |
| 0.039726878 | 0.040197461 | 0.063917526 | 0.063860668 | 0.032110092 | 0.069148936 | 0.053763441 | 0.073333333 | 0.066326531 | 0.063860668 |
| 0.038692098 | 0.044054441 | 0.055944056 | 0.071428571 | 0.042253521 | 0.017857143 | 0.020408163 | 0.05 | 0.06 | 0.071428571 |
| 0.046085859 | 0.041548295 | 0.045602606 | 0.053061224 | 0.037558685 | 0.047619048 | 0.0546875 | 0.044117647 | 0.045454545 | 0.053061224 |

#### 15.3.3 文本特征

在前面两节，本书介绍了有关单变量离散字段的一些特征提取方式，然而本章介绍的竞赛当中还有一类就是多变量离散字段，如兴趣、关键词、和主题等等，如何对这类字段进行处理和特征工程也是非常值得探讨的一点，本节将引入自然语言处理的相关算法，将这类多变量离散字段作为文本特征进行处理。

##### **Sparse**

首先介绍scipy的稀疏矩阵结构，它是一种不同于pandas的DataFrame的数据存储方式，系数矩阵的点在于它的总维度会很高，但是每个用户只在其中一小部分存在取值，因此在保持超高维度的同时又不会使得内存花销过多，这里将从三个方面进行稀疏矩阵特征的生成。

##### **OneHotEncoder**

OneHotEncoder也叫独热编码，是指对单变量离散字段进行编码处理，形成稀疏矩阵的结构，简单来说，就是把一个唯一值个数为N的单变量离散字段变成N维的0、1向量，存储为稀疏矩阵结构，在这里使用DataFrame的格式给参赛者示例如下：

输入：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| uid | age | education | adCategoryId | advertiserId |
| 10971433 | 1 | 1 | 21 | 10055 |
| 7657617 | 2 | 6 | 81 | 2509 |
| 39031487 | 1 | 7 | 142 | 8203 |
| 3671870 | 5 | 1 | 27 | 327 |
| 81984219 | 1 | 6 | 142 | 5459 |
| ... | ... | ... | ... | ... |

输出：

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| uid | age\_1 | age\_2 | age\_5 | education\_1 | education\_6 | education\_7 | ... |
| 10971433 | 1 | 0 | 0 | 1 | 0 | 0 | ... |
| 7657617 | 0 | 1 | 0 | 0 | 1 | 0 | ... |
| 39031487 | 1 | 0 | 0 | 0 | 0 | 1 | ... |
| 3671870 | 0 | 0 | 1 | 1 | 0 | 0 | ... |
| 81984219 | 1 | 0 | 0 | 0 | 1 | 0 | ... |
| ... | ... | ... | ... | ... | ... | ... | ... |

可以看到，经过独热编码后，原本不具有量化大小关系的单变量离散字段转换成了0、1表示的多个连续字段，可以直接用于如线性模型LR之类不能直接支持离散字段的模型。

##### **CountVectorizer**

同样地，既然单变量离散字段可以进行0、1值连续特征的转换，对于多变量离散字段同样有对应的变换方式，即是CountVectorizer，直观上理解，这就是对多变量的每个字段进行计数表示样本在某个取值上的出现次数，当然，本次竞赛数据由于单个用户在如interest上的多值并不会有重复，所以转换后的取值依然只是0、1，下面给出示例，

输入：

|  |  |
| --- | --- |
| uid | interest1 |
| 10971433 | 70 86 109 76 45 28 29 49 5 18 72 36 11 |
| 7657617 | 70 100 47 76 28 33 106 29 59 49 27 7 9 17 56 36 11 |
| 39031487 | 70 109 76 48 28 106 49 122 6 119 5 17 56 116 36 11 |
| 3671870 | 70 76 59 49 36 11 |
| 81984219 | 109 59 49 89 111 |
| ... | ... |

输出：

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| uid | interest1\_100 | interest1\_106 | interest1\_109 | ... | interest1\_72 | interest1\_76 | interest1\_86 | interest1\_89 |
| 10971433 | 0 | 0 | 1 | ... | 1 | 1 | 1 | 0 |
| 7657617 | 1 | 1 | 0 | ... | 0 | 1 | 0 | 0 |
| 39031487 | 0 | 1 | 1 | ... | 0 | 1 | 0 | 0 |
| 3671870 | 0 | 0 | 0 | ... | 0 | 1 | 0 | 0 |
| 81984219 | 0 | 0 | 1 | ... | 0 | 0 | 0 | 1 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... |

##### **TfidfVectorizer**

TfidfVectorizer是一种有关词频的统计向量，和CountVectorizer相同的是特征维度一致，不同的是CountVectorizer计算的是样本在不同维度上的数量值，而TfidfVectorizer计算的是频率，和上述表格同样的输入下，其输出如下，

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| uid | interest1\_100 | interest1\_106 | interest1\_109 | ... | interest1\_72 | interest1\_76 | interest1\_86 | interest1\_89 |
| 10971433 | 0 | 0 | 0.252324646 | ... | 0.376766097 | 0.212263447 | 0.376766097 | 0 |
| 7657617 | 0.333382442 | 0.268970855 | 0 | ... | 0 | 0.187821853 | 0 | 0 |
| 39031487 | 0 | 0.279262135 | 0.231812799 | ... | 0 | 0.195008235 | 0 | 0 |
| 3671870 | 0 | 0 | 0 | ... | 0 | 0.403947142 | 0 | 0 |
| 81984219 | 0 | 0 | 0.378901612 | ... | 0 | 0 | 0 | 0.565768282 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... |

看到这里，参赛者可能自然而然地会产生出一个疑问，那就是文本特征这样的处理办法无疑会产生出超高维度的特征，有可能会引起性能上的问题，针对这一风险，除了采用稀疏矩阵作为存储的数据结构之外，还有一种辅助办法便是进行一定程度的降维，去除冗余的极度稀疏的维度或者经过特征变换映射到低维空间，从而获得计算速度与内存占用上的优化。

#### 15.3.4 特征降维

##### **TruncatedSVD**

Scikit-learn是一个强大的机器学习python开源包，里面包含了常用的各种模块，其中特征分解模块包含多个针对特征降维的算法，用于处理不同种类和形式的特征，本书为了方便参赛者快速熟悉算法流程和技巧，事先对比赛数据进行了采样，约10W量级的数据，然而经过文本特征的处理参赛者可以发现其特征维度爆炸到了25W+，这给建模带来了极大的性能挑战，因此考虑事先进行一定程度的降维。Scikit-learn的decomposition就有对稀疏矩阵结构进行降维的TruncatedSVD算子，可以指定主成分的特征数量进行矩阵输出，其用法也与文本特征的处理算子类似。

#### 15.3.5 特征存储

需要注意的是，通常在竞赛当中为了取得更好地成绩，会将测试集的字段信息加入特征的计算处理，但在实际业务运用中，这种做法是不可能实现的，有的竞赛也会明确要求不能使用到测试集的字段信息进行特征工程。经过前面几节的特征处理之后，除了原始数据的特征以外，有额外的三个特征文件生成，其具体描述如下：

|  |  |  |  |
| --- | --- | --- | --- |
| 简称 | 训练集 | 测试集 | 特征描述 |
| sample | train\_sample.csv | test\_sample.csv | 原始数据 |
| feature | train\_sample\_feature.csv | test\_sample\_feature.csv | exposure+ratio +nunique+ctr +labelencoder |
| sparse | train\_sample\_sparse.csv | test\_sample\_sparse.csv | onehotencoder +countvectorizer +TfidfVectorizer |
| svd | train\_sample\_svd.csv | test\_sample\_svd.csv | sparse+TruncatedSVD |

### 15.4 模型训练

#### 15.4.1 LightGBM

LightGBM模型在训练的时候可以支持categorical\_feature，但前提是需要进行labelencoder编码处理后才能加入，而feature特征模块包含有单变量离散字段的labelencoder，再加上经过降维处理后的多变量离散字段的稀疏矩阵特征svd，采用五折交叉验证训练模型后的预测效果如下，

|  |  |  |
| --- | --- | --- |
| Submission and Description | CV Score | Test Score |
| feature + svd + LightGBM | 0.6792185063662405 | 0.6186427592654342 |

可以看到明显有训练集过拟合现象，有可能是feature里面的特征过拟合以及svd降维后的信息丢失过于严重造成的。

#### 15.4.2 CatBoost

Catboost是本章想要重点介绍的一个模型，因为它直接能够支持文本特征即多变量字段特征的处理和建模，只是用原始数据便可以进行训练和建模，同样采用五折交叉验证训练模型，其预测效果如下，

|  |  |  |
| --- | --- | --- |
| Submission and Description | CV Score | Test Score |
| sample + CatBoost | 0.649001669883728 | 0.6650086607248658 |

#### 15.4.3 XGBoost

CatBoos能够对text\_features和cat\_features直接进行支持的原因是其在内部同样对这些字段进行了稀疏处理，因此可在外层利用Scikit-learn的相关算子进行处理之后再使用XGBoost建模，加上feature一起建模的效果如下：

|  |  |  |
| --- | --- | --- |
| Submission and Description | CV Score | Test Score |
| feature + XGBoost | 0.6790458 | 0.67671527 |

### 15.5 模型融合

#### 15.5.1 加权融合

按照测试集得分数进行简单加权融合，其效果如下，

|  |  |  |
| --- | --- | --- |
| Submission and Description | Train Score | Test Score |
| data['randomforest']\*0.2+data['lightgbm']\*0.3 + data['xgboost']\*0.5 | 0.6815701250536 | 0.682082289028 |

#### 15.5.2 Stacking融合

由于CatBoost里面之利用了原始的特征便有不错的效果，因此考虑增加其他两个模型的stacking数值特征加入到CatBoost进行融合训练，其预测效果如下，

|  |  |  |
| --- | --- | --- |
| Submission and Description | CV Score | Test Score |
| feature + XGBoost | 0.7078828275203705 | 0.6744532384583376 |

### 15.6 赛题总结

#### 15.6.1 更多方案

##### **GroupByMean**

由单变量离散字段与标签组合进行CTR特征提取可以想到，多变量离散字段的稀疏矩阵化后是类似于标签的0、1列，因此可以考虑统计单变量离散字段在某个多变量离散字段取值的均值，即groupby(cat\_features)[text\_features].mean()的形式，比如age为5的人群中interest1包含109的人数比例。

##### **N-Gram**

在做countvectorizer特征提取的时候，本书采用了默认的参数，即ngram=(1,1)，还没有尝试采取更高阶的ngram进行统计，高阶的ngram本质上是增加了一层特征组合，使得同一个多变量离散字段的信息捆绑在了一起，比如标识出了哪些人同时喜欢跑步与骑自行车。

#### 15.6.2 知识点梳理

##### **特征工程**

本章在特征工程方面从通用特征、业务特征以及文本特征三个方面介绍了常见的特征提取办法。其中通用特征主要是单变量离散字段之间的交互统计，包括Count、Nunique与Ratio三种；而业务特征部分介绍了行业场景与领域知识结合的点击率CTR特征，这也是类需要结合标签去做的特征；最后一个文本特征的处理部分介绍了几种不同的稀疏矩阵生成方式，这部分在处理大规模单变量与多变量离散字段时尤其有用。

##### **建模思路**

本章的赛题代表了一种典型的数据组织形式与表格数据结构，针对这类数据可以抽象出相对通用的特征工程方法，这是本书采取这个赛题进行讲解的一部分原因。另一方面，本次赛题是以用户和广告同时作为建模对象，试图对它们进行合理地匹配，Lookalike的原理是通过以往广告的营销结果，找到潜在的和点击该广告的用户相似的用户，从而进行广告的持续曝光与点击，所以重点是要放在寻找用户之间的相似性上面，尤其是在各个维度上的联合相似度，遗憾的是，机器学习受限于特征工程，并不能取得最好的效果，而深度学习神经网络能在文本字段上进行嵌套组合和非线性函数拟合，因此本次竞赛神经网络模型表现较佳。

#### 15.6.3 延伸学习

##### **贝叶斯平滑**

本书介绍得点击率统计方式极为简单，借鉴了五折思想进行交叉统计防止leak的发生，但这样仍旧存在许多问题，比如统计方法相对粗糙，具有极大的不确定性容易过拟合训练集等。

##### **深度学习**

利用神经网络进行embedding实现对文本特征的处理，是本次竞赛top队伍的共有方案，其中尤其数nffm和xdeepffm模型效果最好，有兴趣的参赛者可参考guoday开源

<https://github.com/guoday/Tencent2018_Lookalike_Rank7th>