# 2018科大讯飞AI营销算法大赛总结

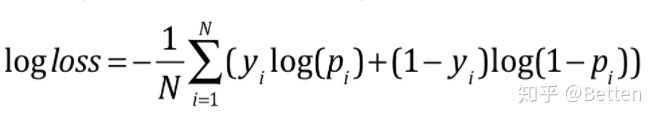
## 1、赛题分析

本次大赛提供了讯飞AI营销云的海量广告投放数据，参赛选手通过人工智能技术构建预测模型预估用户的广告点击概率，即给定广告点击相关的广告、媒体、用户、上下文内容等信息的条件下预测广告点击概率。

这又是一道关于CTR的问题，对于CTR问题而言，广告是否被点击的主导因素是用户，其次是广告信息。所以我们要做的是充分挖掘用户及用户行为信息，然后才是广告主、广告等信息。

赛题特征：广告信息、媒体信息、用户信息、上下文信息

提供数据：共1001650初赛数据 和 1998350条复赛数据（复赛训练数据为初赛数据和复赛数据）

评价指标：通过logloss评价模型效果（越小越好）

其中N表示测试集样本数量，y\_i 表示测试集中第i个样本的真实标签， p\_i 表示第i个样本的预估转化率。这类评估函数常用logloss和AUC,简单的说logloss更关注和观察数据的吻合程度，AUC更关注rank order。如果是按照概率期望来进行收费投放的话就用logloss，如果定投一定量就用AUC，主要还是和业务相关。

## 2、探索性分析

1、这一部分将会对部分数据进行分析，另外获取每个类别特征的转化率分布情况判断特征效果，看分布可以有一个很好的初步验证作用。

不同时刻的曝光量和点击率变化，将一天分成4个时段的曝光量和点击率情况。

2、训练集正负样本比例，大约为1:4，应该经过了降采样。

3、广告的长宽是很重要的特征，正负样本中关于这两个特征的分布存在较为明显的区别。

4、在训练集中的部分热门广告，adid为1537089的广告在训练集中曝光次数超过了12万次。

5、OPPO和 vivo的用户最多，而这两种机型的用户点击率也高于其他手机的用户

### 3、数据预处理

**由于数据噪音比较多，所以细致的预处理能够是模型更具泛化性，同时挖掘更多特征。**

1、初复赛训练数据合并后去重（7361条）

2、提取广告投放时间信息，日期、小时以及早中晚时间段

0-6>--1 | 7-12>--2 | 13-18>--3 | 19-24>--4

3. 细分广告主行业与媒体广告位，去除只有一个取值的字段

102400\_102401>--102400 102401

4. 清洗手机品牌和机型字段，对同类进行合并

iphone>--apple | redmi>--xiaomi | honor>--huawei

5. 对操作系统及其版本、名称进行更细粒度的刻画

5.1.1>--5 1 1 | 6.0.1>--6 0 1

6. 构造虚拟用户组别，对用户标签和其他类别特征进行编码

7. 对city特征进行切分，如

框内为身份证前六位，51代表广东省，04代表广州市，10代表白云区

8. 缺失值填充，对不同类型的数据填充不同类型的值

## 4、特征工程

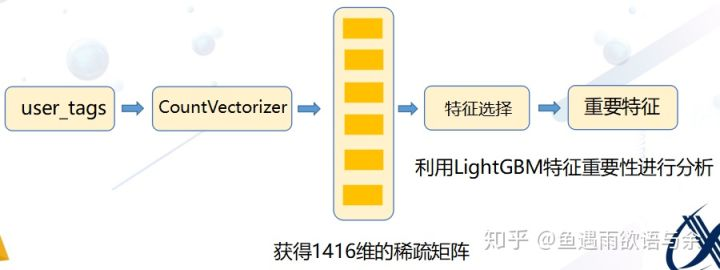
**4.1 特征构造**

**1. 基础特征：**原始特征（广告信息 媒体信息 用户信息 上下文信息）

**2. One-hot：**将类别特征离散化

由于最后融合三套代码的结果，所以有的代码进行了one-hot，有的没有这样做，而是直接labelencoder。

**3. user\_tags多值特征**：因为包含用户的属性信息，所以完美的表达user\_tags至关重要，提取有效属性，减少冗余。



**4. 统计特征：**

**统计特征我们用的都是常规操作，如count、ratio、nunique和ctr相关特征。**

**count：一维+二维count计数特征（如广告主id共计投放次数）**

**ratio：类别偏好的ratio比例特征(如广告主id的某个广告id投放比例)**

**mean：用户标签与其他字段的组合mean特征（如广告id对用户性别的投放比例）**

**nunique： 类别变量的nunique特征（如广告主id有多少个不同的广告id）**

**点击率：这里使用的历史点击率，来挖掘历史点击信息，同时防止过拟合**

### 4.2特征选择

**这里我们主要用了卡方检验和特征重要性，由于三套代码，所有使用的方法并不相同。**

**user\_tags特征我们分别用了卡方检验和特征重要性。**

train\_new **=** pd**.**DataFrame()

test\_new **=** pd**.**DataFrame()

cntv **=** CountVectorizer()

cntv**.**fit(data['user\_tags'])

train\_a **=** cntv**.**transform(train['user\_tags'])

test\_a **=** cntv**.**transform(test['user\_tags'])

train\_new **=** sparse**.**hstack((train\_new, train\_a), 'csr', 'bool')

test\_new **=** sparse**.**hstack((test\_new, test\_a), 'csr', 'bool')

*# 卡方检验*

SKB **=** SelectPercentile(chi2, percentile**=**95)**.**fit(train\_new, train\_y)

train\_new **=** SKB**.**transform(train\_new)

test\_new **=** SKB**.**transform(test\_new)

### 4.3stacking 特征

 **交叉统计特征太多内存不够怎么办？**

 **如何才能在减少特征维度的同时最大限度地保留所有特征的区分度信息？**

*# 通过stacking获取新的特征，减少内存的同时，又能保留完整特征的信息*

**def** **getStackFeature**(df\_,seed\_):

skf **=** StratifiedKFold(n\_splits**=**5,random\_state**=**seed\_,shuffle**=True**)

train **=** df\_**.**loc[train\_index]

test **=** df\_**.**loc[test\_index]

train\_user **=** pd**.**Series()

test\_user **=** pd**.**Series(0,index**=**list(range(test\_x**.**shape[0])))

**for** train\_part\_index,evals\_index **in** skf**.**split(train,train\_y):

EVAL\_RESULT **=** {}

train\_part **=** lgb**.**Dataset(train**.**loc[train\_part\_index],label**=**train\_y**.**loc[train\_part\_index])

evals **=** lgb**.**Dataset(train**.**loc[evals\_index],label**=**train\_y**.**loc[evals\_index])

bst **=** lgb**.**train(params\_initial,train\_part,

num\_boost\_round**=**NBR, valid\_sets**=**[train\_part,evals],

valid\_names**=**['train','evals'],early\_stopping\_rounds**=**ESR,

evals\_result**=**EVAL\_RESULT, verbose\_eval**=**VBE)

train\_user **=** train\_user**.**append(pd**.**Series(bst**.**predict(train**.**loc[evals\_index]),index**=**evals\_index))

test\_user **=** test\_user**+**pd**.**Series(bst**.**predict(test))

**return** train\_user,test\_user

## 5算法模型

**GBDT模型记忆性更强，记忆特征和标签相关特征组合能力强，因此在小数据集上有很好的结果 。**

**FFM和DeepFFM初期尝试并未得到很好的效果**

**最终我们选择了XGBoost和LightGBM，得到的结果并做了最终的加权融合。**

lgb\_clf **=** lgb**.**LGBMClassifier(boosting\_type**=**'gbdt', num\_leaves**=**48, max\_depth**=-**1, learning\_rate**=**0.02, n\_estimators**=**6000, max\_bin**=**425, subsample\_for\_bin**=**50000, objective**=**'binary', min\_split\_gain**=**0,min\_child\_weight**=**5, min\_child\_samples**=**10, subsample**=**0.8, subsample\_freq**=**1,colsample\_bytree**=**0.8, reg\_alpha**=**3, reg\_lambda**=**0.1, seed**=**1000, n\_jobs**=-**1, silent**=True**)

skf **=** list(StratifiedKFold(y\_loc\_train, n\_folds**=**5, shuffle**=True**, random\_state**=**1024))

baseloss **=** []

loss **=** 0

**for** i, (train\_index, test\_index) **in** enumerate(skf):

print("Fold", i)

lgb\_model **=** lgb\_clf**.**fit(X\_loc\_train[train\_index], y\_loc\_train[train\_index],

eval\_names**=**['train', 'valid'],

eval\_metric**=**'logloss',

eval\_set**=**[(X\_loc\_train[train\_index], y\_loc\_train[train\_index]),

(X\_loc\_train[test\_index], y\_loc\_train[test\_index])], early\_stopping\_rounds**=**100)

baseloss**.**append(lgb\_model**.**best\_score\_['valid']['binary\_logloss'])

loss **+=** lgb\_model**.**best\_score\_['valid']['binary\_logloss']

test\_pred **=** lgb\_model**.**predict\_proba(X\_loc\_test, num\_iteration**=**lgb\_model**.**best\_iteration\_)[:, 1]

print('test mean:', test\_pred**.**mean())

res['prob\_%s' **%** str(i)] **=** test\_pred

print('logloss:', baseloss, loss **/** 5)

## 6 思考总结

**由于本次比赛数据中缺乏用户id这一关键信息，用户画像难以得到清晰地建立，因此如何充分挖掘用户标签中所包含的信息至关重要。**

**即使是同样的业务场景，在不同的数据收集背景下，同样的特征完全可能会起到完全相反的效果，这也是一种数据陷阱。**

**匿名化数据需要对数据进行充分理解分析，甚至可以尝试根据业务理解进行反编码，这样能够为特征工程指明方向。**

**建模过程中充分考虑了用户标签与其他信息的交互作用，并采用Stacking抽取特征信息的方式减少维度与内存的使用，对广告与用户交互信息的充分挖掘，也使得模型在AB榜测试相对稳定。**

**模型缺乏差异性和创新性，最开始尝试过deepffm，由于效果一般而没有坚持改进，大部分精力放在了数据理解与特征挖掘上。**

**数据特点**

按类型： 数值型、单值型、多值型(user\_tags)

按业务类型： 广告主、媒体、上下文（设备环境）、用户

**模型方案选择**

初步使用常规的lgb\_model，速度快。效果优秀，方便测试特征。

多值特征CountVector处理，以及单值特征one-hot，这里使用csr矩阵来存储，大幅度优化内存。

验证集策略，这里使用的是5折，自己测试的时候使用过三种方案

5折交叉检验，验证特征是否符合整体数据集。

最后一天作为验证集，验证特征是否时序稳定（线上验证集是后一天）

最后一天作验证集，但是拆分四份，分别作统计特征以及分别验证（考虑到线上测试集采样比例降低了1/4）

**关键点**

**count特征：**

作为类别特征中最重要的属性之一，count特征能表示很多有趣的东西。比如你有1000个机型，但是其中有一两百个机型都是比较奇葩的名字， 但是出现次数很少，对于模型而言，几乎不可能学到出现次数很少的主体的信息（甚至无法满足叶子分裂条件），比如 某某的iphone、xx最酷的、土豪金xx， 但是他们的count特征就有趣了。因为你会发现count 都为1的数据，这些人，都是喜欢"个性化"的。

**ratio特征:**

ratio 特征表示选择。比如，某个广告主男生用户的受众占比比较高。诸如此类。 ratio 特征有效的前提是（广告主或者线上推荐系统，一定会朝着转化率高的属性加大投放。所以这里如果知道他们系统的应对策略，也会是一个很有趣方向，比如腾讯赛的lookalike机制造成的uidcount的泄露） 另一个前提是，ratio的双方，存在着真实的交互，比如，广告主有权利选择某个平台，某些类型的用户。比如推荐系统能选择用户等等。 所以事实上暴力的ratio效果会很差，这里我选择的是 媒体、广告主、上下文的交互。

**count\_day\_hour\_\*\***

基于对业务系统的思考。发现每天的每个小时的投放量和转化率有密切关联。因为系统发现转化率变高了，会提高投放量？响应周期定义为1个hour。 线下测试，确实也是很强的特征，但是线上效果不好，通过数据观察发现，测试集最后一天的数量大约采样降低了到了1/4，所以对最后一天使用了\* 4的操作（也可用用/当天的总量来替代） （这里其实告诉我们，请珍惜每一个毒特征，特征毒，说明模型学到了某些东西，而这些东西，测试集没有了。那么就要考虑，是真的没有了？还是要通过某些处理就可以找回来。找回来就是强特～）

**hint:**

这里有个好玩的思考方向，如果广告主选择了60%的男生，可是他们系统里男生的占比达到了70%，那么该广告真的是男生优先的么？这个特征就不开源啦～以上的分数也基本足够了。