# TalkingData 广告流量欺诈数据分析及预测

TalkingData AdTracking Fraud Detection Challenge


2018 届　　软 件 学 院

专　　业　　软 件 工 程

学　　号　　SA18225058

学生姓名　　邓洋杰

指导教师　　汪　炀


**完成日期　　2017 年 5 月 21 日**

# 数据集介绍

选自Kaggle赛题：TalkingData AdTracking Fraud Detection Challenge。

链接：https://www.kaggle.com/c/talkingdata-adtracking-fraud-detection

目标：本题选自kaggle在完成最终目标欺诈预测的同时，也从这些海量数据中学得一系列知识，用来辅助用户决策。

赛题介绍：欺诈风险无处不在，但对于那些在网上投放广告的公司而言，欺诈点击可能会以压倒性的数量发生，产生误导性的点击数据，并会浪费的资金。通过广告渠道点击广告即可影响成本。中国每月有超过10亿的智能移动设备正在使用，因此中国是世界上最大的移动市场，遭受大量的欺诈性流量。

TalkingData是中国最大的独立大数据服务平台，覆盖全国70％以上的有源移动设备。他们每天处理30亿次点击，其中90％可能是欺诈性的。他们目前为应用程序开发人员防止点击欺诈的方法是衡量用户点击其投资组合的过程，并标记产生大量点击却永远不会最终安装应用程序的IP地址。有了这些信息，他们就建立了一个IP黑名单和设备黑名单。

虽然成功，但他们希望始终比欺诈者领先一步，并向Kaggle社区寻求帮助，以进一步开发他们的解决方案。您面临的挑战是构建一种算法，用于预测手机用户在点击移动应用广告后是否会下载应用。为了支持您的建模，他们提供了一个慷慨的数据集，涵盖了4天内大约2亿次点击！

结题时间：2018年5月7日

训练数据集大小为7.54GB，共有184903891条。

测试数据集大小为863.3MB，共有18790469条。

# 目录

# 第 1 章　数据集展示

## 1.1　载入数据

本次数据集选取，样本抽样方法使用分层抽样。采用 pandas 的 read_csv 函数 skiprows 参数，保证样本具有代表性。在训练模型中使用分层抽样抽取 7500 0000 条数据，在数据分析中使用分层抽样抽取 100 0000 条数据。

载入数据类型设置：dtypes = { ‘ip’ : ‘uint32’ , ‘app’ : ‘uint16’ , ‘device’:‘uint16’,‘os’:‘uint16’,‘channel’:‘uint16’,‘is_attributed’: ‘uint8’ , ‘click_id’ : ‘uint32’ }，设置读入列：[ ‘ip’ , ‘app’ , ‘device’ , ‘os’ , ‘channel’ , ‘click_time’ , ‘is_attributed’ ]来降低内存消耗。

## 1.2　数据展示

打印读入数据集的条数及列数。此数据集一共有 100 0000 条记录，7 列属性。

```
In [4]: print("train -  rows:",trainset.shape[0]," columns:", trainset.shape[1])
        train -  rows: 1000000  columns: 7
```

图表 1-1

简单地查看读入的数据表。

```
In [5]: trainset.head()
```

Out[5]:

| | ip | app | device | os | channel | click_time | is_attributed |
|---|---|---|---|---|---|---|---|
| **85978** | 102897 | 28 | 1 | 19 | 135 | 2017-11-08 12:35:03 | 0 |
| **318367** | 106200 | 12 | 1 | 15 | 245 | 2017-11-08 12:39:06 | 0 |
| **189684** | 59962 | 14 | 1 | 13 | 379 | 2017-11-08 12:36:49 | 0 |
| **102739** | 275930 | 18 | 1 | 19 | 121 | 2017-11-08 12:35:20 | 0 |
| **142699** | 88053 | 3 | 1 | 13 | 130 | 2017-11-08 12:36:01 | 0 |

图表 1-2

## 1.3 数据集类型

打印读入数据集的条数及列数。此数据集一共有 100 0000 条记录，7 列属性。

1.3.1 记录数据：每个记录包含固定的数据字段（属性）集。记录之间或数据字段之间没有明显的联系，并且每个记录（对象）具有相同的属性集。

1.3.2 维度：数据集维度是数据集中的对象具有的属性数目。值得注意地是，分析高维数据有时会陷入所谓的维灾难，而此数据集含有 7 个维度。

数据属性（特征）解读：

ip：为每次点击提交的 ip 地址（32 位无符号整数离散特征）；

app：营销应用的编号（16 位无符号整数离散特征）；

device：用户手机的设备类型（16 位无符号整数离散特征）；

os：手机用户的操作系统（16 位无符号整数离散特征）；

channel：移动广告商发布的渠道（16 位无符号整数离散特征）；

click_time：手机用户点击广告的时间（时间类型）；

is_attribute:点击广告的手机用户是否下载应用（8 位无符号整数二元特征）

1.3.3 稀疏性：有些数据集，如具有非对称特征的数据集，一个对象的大部分属性上的值都为 0；在许多情况下，非零值还不到 1%。只存储非零值有助于节省大量的计算时间和存储空间。且有很多的数据挖掘算法适合处理稀疏特征数据。本数据集中，只有 is_attributed 具有大量零值，具有很高稀疏性。

## 1.4 数据汇总统计

期望数据完美是不现实的。由于人的错误，测量设备的限制或数据收集过程的漏洞都可能导致问题。数据的值乃至整个数据对象都可能丢失。所以在真实场景下，我们获得的数据集可能出现包括噪声、伪像、偏倚、精度和准确率等测量误差和收集错误，同时还包括离群点、遗漏和不一致的值、重复数据等问题。此时，对数据集进行缺失值，离群点，重复数据等数据质量问题进行探索。

通过对数据集进行计数，平均值，标准差和四分位数计算，可以对数据对象的取值范围，散度，相关性做一个大致了解。

| | ip | app | device | os | channel | is_attributed |
|---|---|---|---|---|---|---|
| count | 1000000.000000 | 1000000.000000 | 1000000.000000 | 1000000.000000 | 1000000.000000 | 1000000.000000 |
| mean | 90438.079843 | 12.100961 | 17.448929 | 22.264450 | 273.023056 | 0.002432 |
| std | 68681.029471 | 16.325403 | 237.742996 | 53.345877 | 126.174892 | 0.049255 |
| min | 9.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| 25% | 39711.000000 | 3.000000 | 1.000000 | 13.000000 | 153.000000 | 0.000000 |
| 50% | 78120.000000 | 12.000000 | 1.000000 | 18.000000 | 265.000000 | 0.000000 |
| 75% | 116715.000000 | 15.000000 | 1.000000 | 19.000000 | 379.000000 | 0.000000 |
| max | 287533.000000 | 704.000000 | 3836.000000 | 777.000000 | 498.000000 | 1.000000 |

图表 1-3

如上表所示，本次数据分析使用分层抽样方法取 100 0000 条数据，我们可以观察到，虽然 ip 的取值范围很大，但是 app，device 和 channel 只有一个很小的离散类型取值范围. 我们将其分为 is_attributed==1/0 来继续观察。

| | ip | app | device | os | channel | is_attributed |
|---|---|---|---|---|---|---|
| count | 997568.000000 | 997568.000000 | 997568.000000 | 997568.000000 | 997568.000000 | 997568.0 |
| mean | 90231.728098 | 12.057656 | 17.415991 | 22.256853 | 273.145461 | 0.0 |
| std | 68477.908033 | 16.249001 | 237.871871 | 53.332551 | 126.203595 | 0.0 |
| min | 9.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.0 |
| 25% | 39611.000000 | 3.000000 | 1.000000 | 13.000000 | 153.000000 | 0.0 |
| 50% | 77949.000000 | 12.000000 | 1.000000 | 18.000000 | 265.000000 | 0.0 |
| 75% | 116501.750000 | 15.000000 | 1.000000 | 19.000000 | 379.000000 | 0.0 |
| max | 287533.000000 | 704.000000 | 3836.000000 | 777.000000 | 498.000000 | 0.0 |

图表 1-4

| | ip | app | device | os | channel | is_attributed |
|---|---|---|---|---|---|---|
| count | 2432.000000 | 2432.000000 | 2432.000000 | 2432.000000 | 2432.000000 | 2432.0 |
| mean | 175080.307566 | 29.863898 | 30.959704 | 25.380757 | 222.814556 | 1.0 |
| std | 94692.587099 | 31.168469 | 176.661832 | 58.485434 | 102.107938 | 0.0 |
| min | 27.000000 | 0.000000 | 0.000000 | 0.000000 | 3.000000 | 1.0 |
| 25% | 79131.750000 | 19.000000 | 0.000000 | 13.000000 | 115.750000 | 1.0 |
| 50% | 222751.000000 | 19.000000 | 1.000000 | 19.000000 | 213.000000 | 1.0 |
| 75% | 255693.250000 | 35.000000 | 1.000000 | 27.000000 | 274.000000 | 1.0 |
| max | 287466.000000 | 481.000000 | 3014.000000 | 748.000000 | 498.000000 | 1.0 |

图表 1-5

我们我们可以观察到在两个数据集中，ip，os 和 channel 在 is_attrinbuted== (0/1)中有相同的取值范围. 但是在 is_attributed=0 中 app 的最大值将近是 is_attributed=1 的两倍而且 is_attributed=0 的最大值将近是 is_attributed=1 的四倍大。

## 1.5 缺失值统计

定义统计函数 miss_data( )函数，来统计每个属性列中的缺失值。

In [6]:
```
def missing_data(data):
    total = data.isnull().sum().sort_values(ascending = False)
    percent = (data.isnull().sum()/data.isnull().count()*100).sort_values(ascending = False)
    return pd.concat([total, percent], axis=1, keys=['Total', 'Percent'])
missing_data(trainset)
```

Out[6]:

| | Total | Percent |
|---|---|---|
| is_attributed | 0 | 0.0 |
| click_time | 0 | 0.0 |
| channel | 0 | 0.0 |
| os | 0 | 0.0 |
| device | 0 | 0.0 |
| app | 0 | 0.0 |
| ip | 0 | 0.0 |

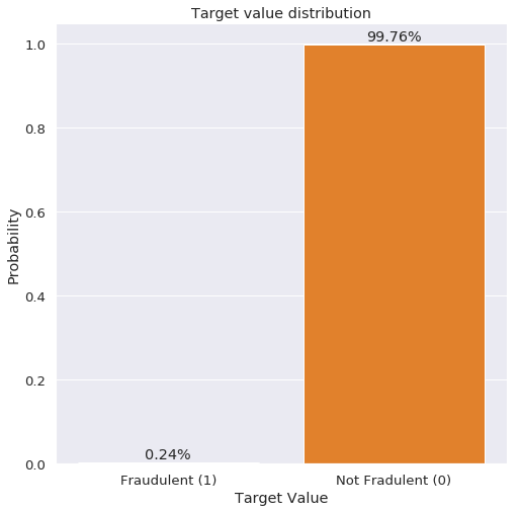
可以观察到上图所示，本次选取 100 0000 条数据集的 7 个维度没有出现缺失值。本数据集没有出现缺失值，收集数据完整。

## 1.5 样本平衡性

在数据分析中，所谓的类别不平衡问题指的是数据集中各个类别的样本数量极不均衡。以二分类问题为例，假设正类的样本数量远大于负类的样本数量，通常情况下把样本类别比例超过 4:1（也有说 3:1）的数据就可以称为不平衡数据。占比 percent 表达式如下：

$$percent = \frac{N1}{total}$$

percent为每个属性值的数据对象占比，N1为每个属性值的数据对象个数，total为数据对象总数。

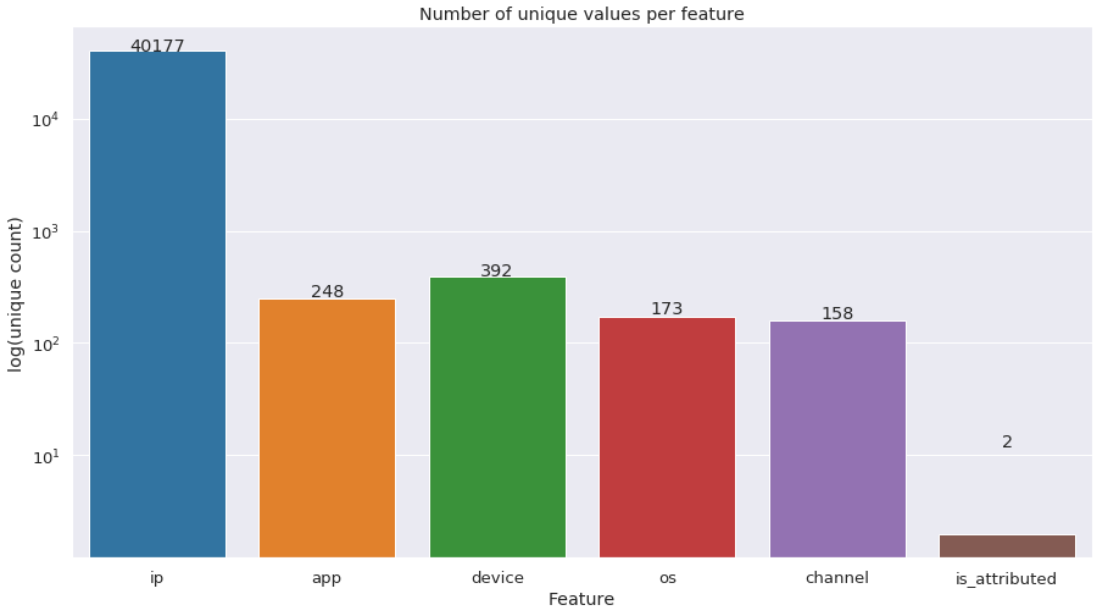


通过对数据集的样本标签is_attributed进行可视化，如上图，可以看到正样本（is_attributed=1）数量仅占0.24%。简单来讲，样本不平衡会使得我们的分类模型存在很严重的偏向性。如果正负样本比例为100:1,那岂不是把全部样本都判定为正样本就有99%+的分类准确率了。具体解决，样本不平衡方法，将在模型训练部分进行讲解。

## 1.6 重复数据检测

数据集可能包含重复或几乎重复的数据对象，由于同一用户可能在不同时间多次点击广告，以及不同用户可能使用相同的 app，device，channel 进行点击。所以在不同的维度上可能存在较高的数据重复率。

此时，我们对 ip，app，device，os，channel，is_attribute7 个维度进行重复数据检测，统计每个维度的唯一值。




读入100 0000条数据，统计此数据集中每个维度数据的维一值个数。可以发现：ip地址的唯一值占比4%，表示点击群体较为固定，而其中也包括大量的欺诈性点击ip。app，device，os，channel与用户喜好，广告商选择有关，具有较低的唯一值。is_attributed为数据对象标签为二分类属性。

# 第 2 章　数据分析

## 2.1　关联性分析

关联性分析用于发现隐藏在大型数据集中的有意义的联系。所发现的规则可以用关联规则或频繁集的形式表示。在关联分析中，关联规则的强度可以用它的支持度和置信度来度量，关联规则是形如 的蕴含表达式，其中 X 和 Y 是不相交的项集。

支持度需要用到支持度计数，即包含特定项集的事务个数，其支持度计数 表达式如下：

$$\sigma(X) = |\{t_i | X \subseteq t_i, t_i \in T\}|$$

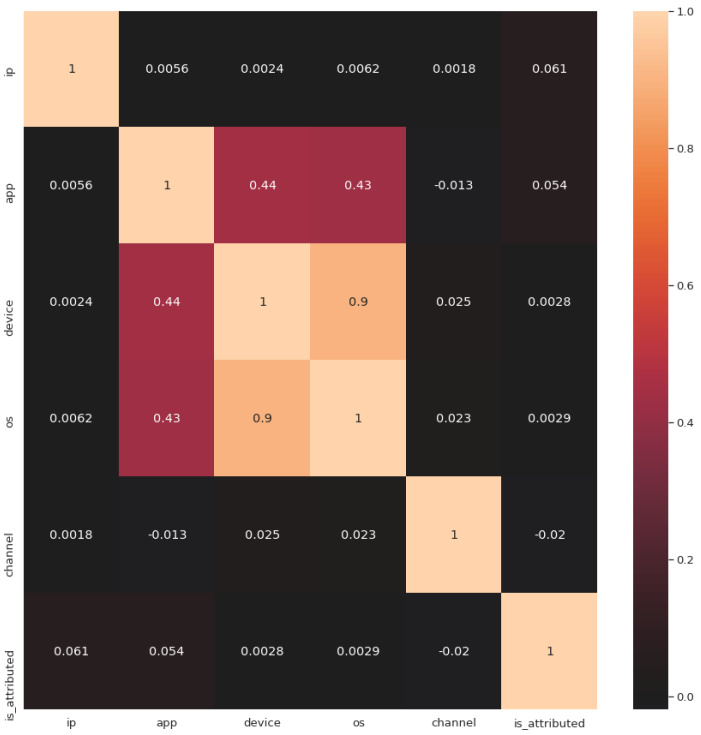
其中，符号|.|表示集合中元素的个数。支持度确定规则可以用于给定数据集的频繁程度，而置信度确定 Y 在包含 X 的事务中出现的频繁程度。支持度（s）和置信度（c）这两种度量的形式定义如下：

$$s(X \rightarrow Y) = \frac{\sigma(X \cup Y)}{N}$$

$$c(X \rightarrow Y) = \frac{\sigma(X \cup Y)}{\sigma(X)}$$

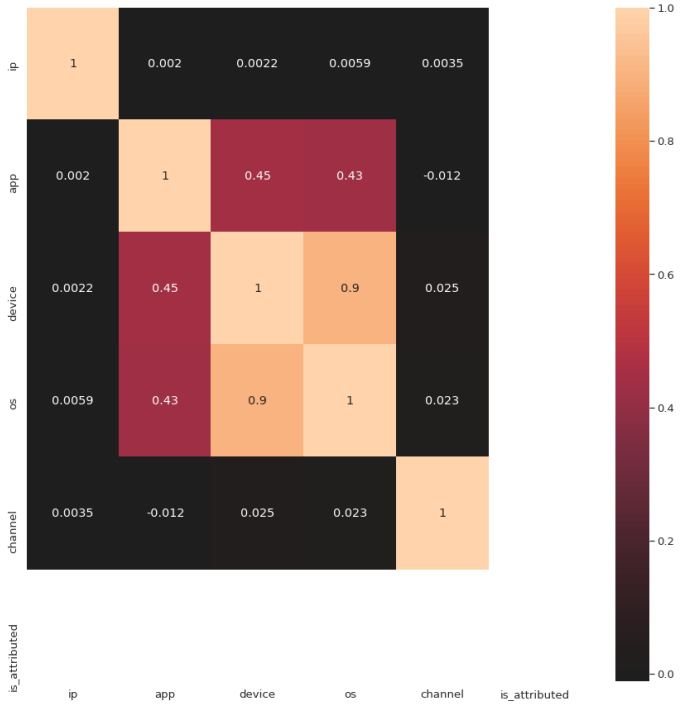
支持度是一种重要的度量，因为支持度很低的规则可能只是偶然出现。对于给定的规则 ，置信度越高，Y 在包含 X 的事务中出现的可能性就越大。置信度也可以估计 Y 在给定 X 下的条件概率。我们应当小心解释关联分析的结果。由关联规则作出的推论并不必然蕴含因果关系。它只表示规则前件和后件中的项明显地同时出现。另一方面，因果关系需要关于数据中原因和结果属性的知识，并且通常涉及长期出现的联系。

这里对 100 0000 条数据的 trainset 进行 2-项集的关联规则发现。对关联度标准化后可视化如下：

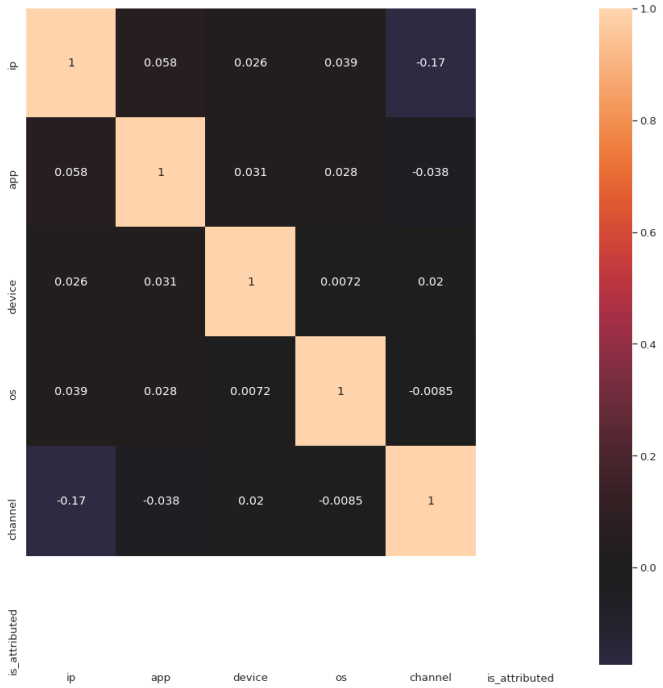



可以发现{app，device}，{app，os}，{device，os}具有较高的关联度，分别为0.44，0.43，0.90。这些特征属性从数据分布上来看具有较高的关联性。

这些关联度是从整个数据集来看。当我们训练训练模型时，关注点主要放在不同类别数据集的关联度，接着查看在负样本（is_attributed=0）中，各维度之间的关联性，如下：



在负样本中不同维度之间的关联性与整个数据集关联度大体一致。最后，查看在正样本（is_attributed=1）中，各维度之间的关联性，如下：



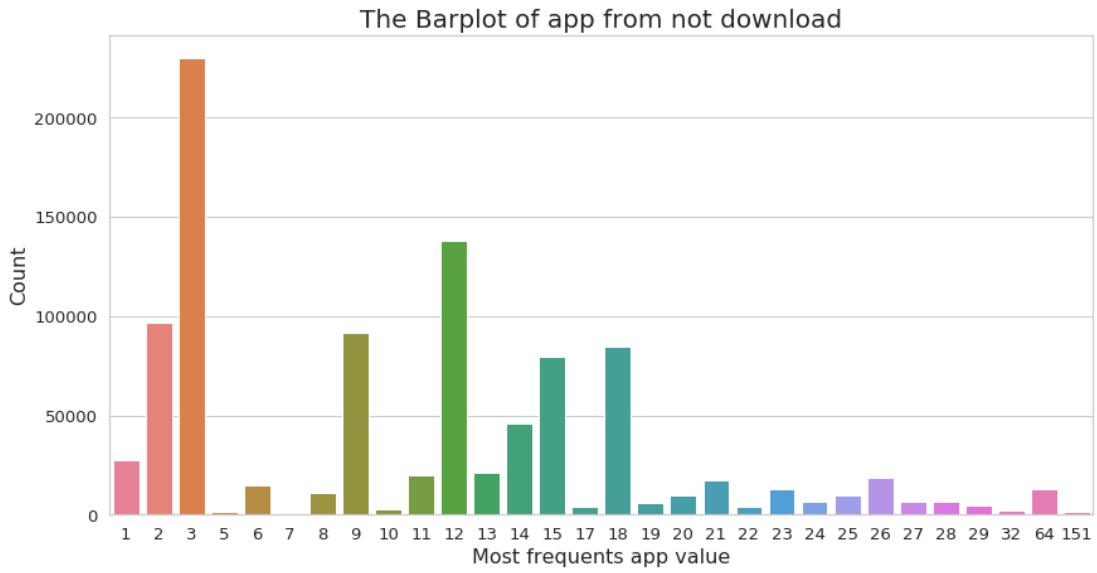

由于正样本数量较少，维度之间的关联性不太明显，但与负样本中有显著不同的是正样本各维度之间没有较强的关联性。此时，对于这条关联规则的解释是：用户点击是否是欺诈性的与用户所用设备，点击渠道等并没有特殊关系。而欺诈性点击则常常为某些平台通过脚本等机器恶意点击来骗取广告商的广告费用。由于这是极不平衡的数据集，负样本占比过高，从而导致负样本的特征关联与整个数据集具有相似性。在欺诈点击检测中，我们需要更多地关注 app，device，os 这样的高关联特征的数据对象。

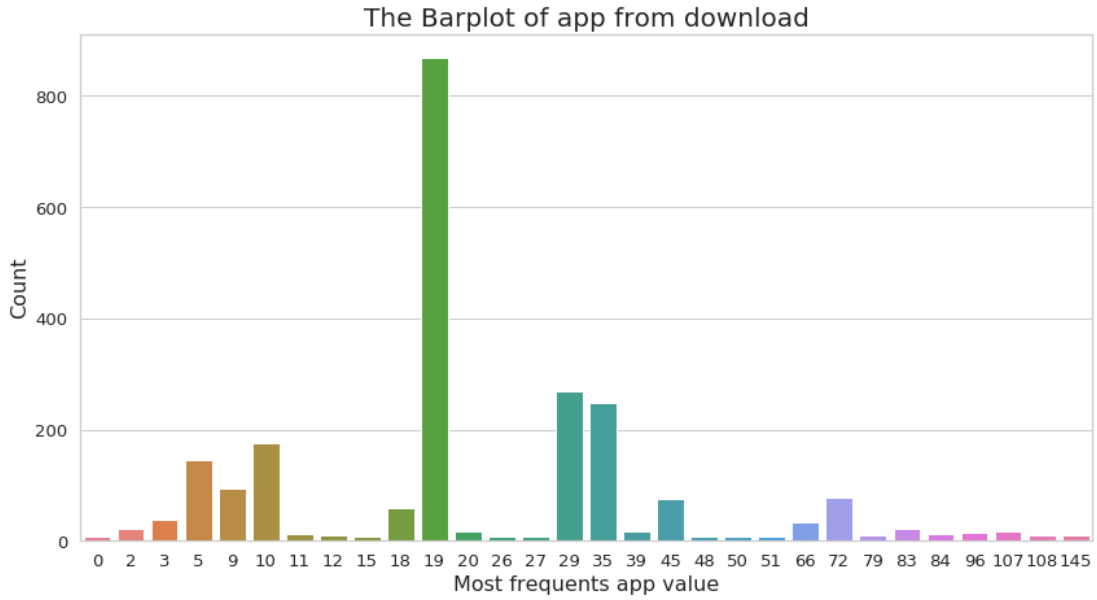
此外也需注意到对大型数据集进行关联分析时，需要处理两个关键的问题：第一，从数据集中发现模式可能在计算上要付出很高的代价；第二，所发现的某些模式可能是虚假的，因为它们可能是偶然发生的。

## 2.2 特征分析

通过对以上数据集的分析，我们不难发现 app，channel 为广告商的战略投放选择。我们可以通过对数据进行分析，来分别找出 app，channel 中选择这些哪一个 app，channel 投放会获得较好的投资回报，即具有较高的下载率。

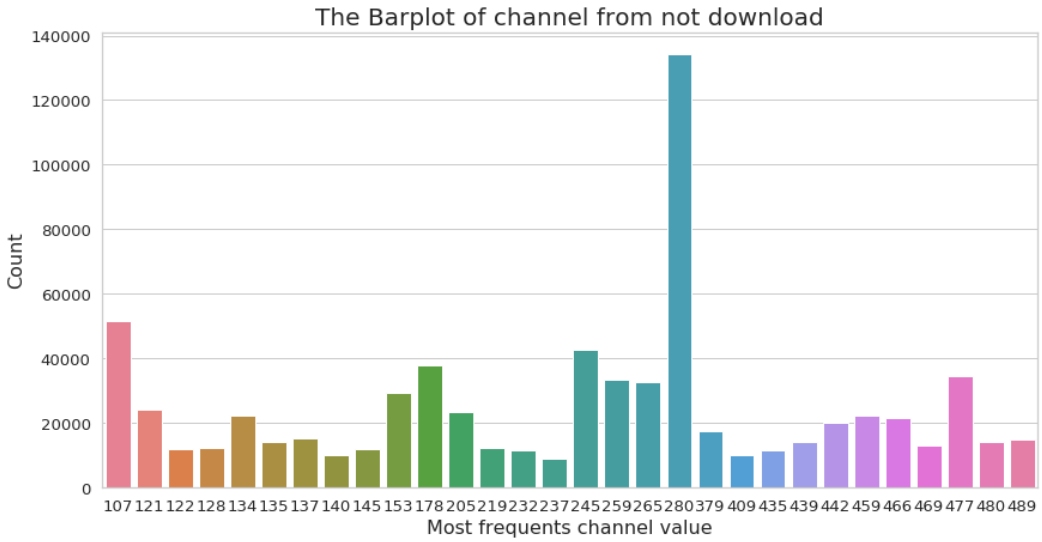
对 100 0000 条数据中对 app，channel 分别进行正/负样本进行属性值点击排序，来挖掘可以用于辅助广告商决策的结果。app 进行正负样本的点击统计前 30 名分布如下：

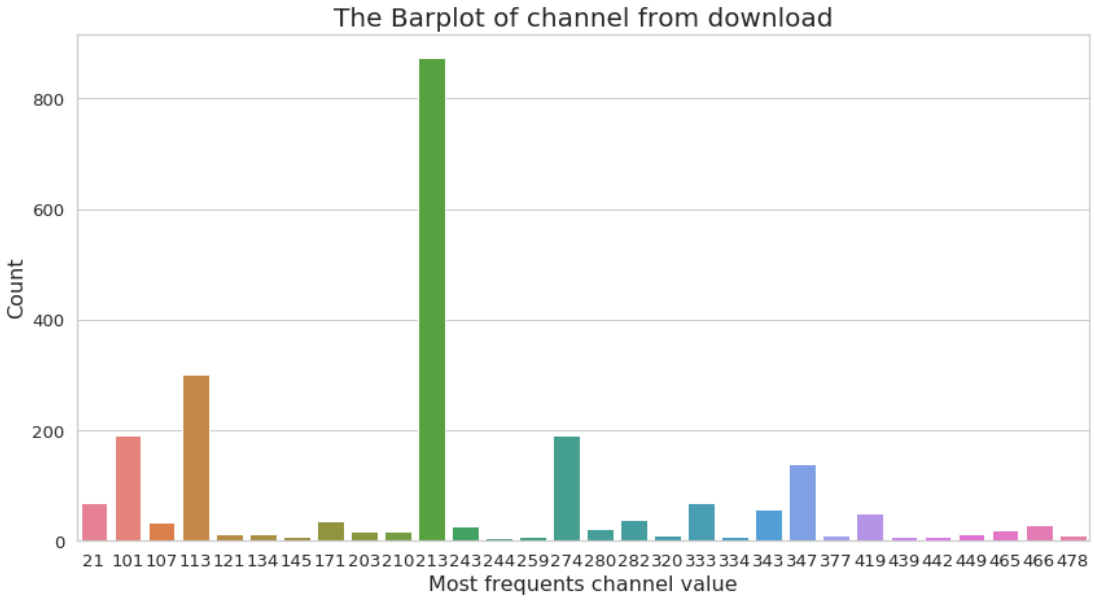



以上为 is_attributed=0/1 的 app 最高点量的前 30 名分布。可以观察到在欺诈点击（is_attributed=0）中 app=3 占有最高点击量。在真实下载点击 is_attributed=1 中，app=19 占有最高点击量。由此，可以向广告商建议：增加对编号为 3 的 app 的广告投入，再对编号 19 的 app 进行提高欺诈检测频率。

接着再对 channel 进行正负样本的点击量统计前 30 名，如下：



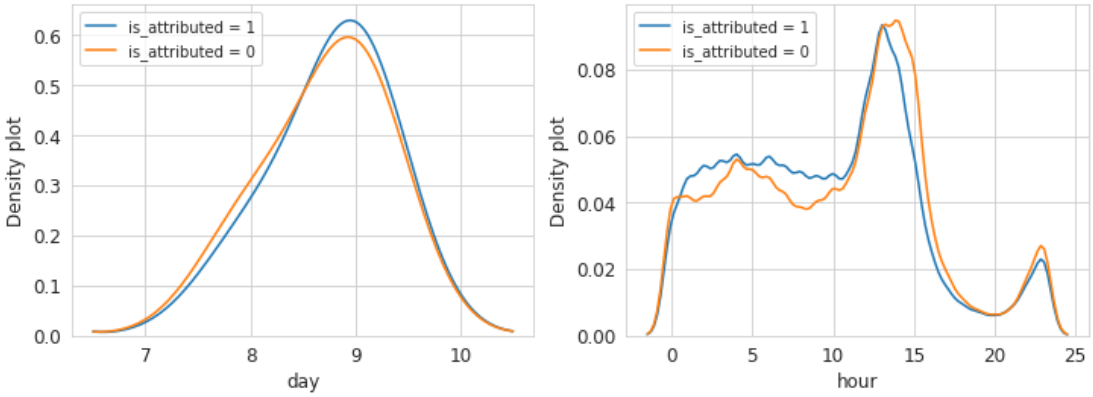

从 channel 的正负样本的前 30 名的分布中可以发现，负样本中编号为 280 的 channel 具有最高点击量，在正样本中编号为 213 的 channel 具有最高点击量。由此，可以向广告商提出建议：增加对编号 213 的 channel 的广告投放量，同时提高对编号为 280 的 channel 的欺诈检测频率。

## 2.3 时间分析

我们采用分层抽样的方法抽取100 0000条数据，将其分成is_attributed=0/1样本集来分别观察，并对其分别从day，hour两个角度统计其的核密度分布图。核密度估计（Kernel density estimation），是一种用于估计概率密度函数的非参数方法，$x_1, x_2, \ldots x_n$为独立同分布F的n个样本点，设其概率密度函数为f，核密度估计表达式如下：

$$\widehat{f_h}(x) = \frac{1}{n}\sum_{i=1}^{n} K_h(x - x_i) = \frac{1}{nh}\sum_{i=1}^{n} K(\frac{x - x_i}{h})$$

计算is_attributed=0/1的核密度分布如下：




可以观察到，与false（is_attributed = 0）相比，true（is_attributed = 1）点击的分布更加多样化（小时，分钟，秒）。这可以用两种方式解释：一种解释可能是由于true案例数量减少，分布不均匀。另一种解释可能是（通过大量案例验证）由于欺诈（false）点击的编程性质，它们的分布更均匀。一个观察结果与小时密度图相关：真假都有一定的小时轮廓，高原在1到16之间，鞍座在16到20之间，高峰在21到22之间。欺诈的高原（false）点击显示另一种模式。

## 2.4 特征组合

组合两个（或更多个）特征是用来学习非线性关系的一种常用做法。常采用的特征合成方法包括：将一个特征与本身或其他特征相乘、两个特征相除、对连续特征进行分桶等方式。且机器学习模型很少会组合连续特征。不过，机器学习模型却经常组合独热特征矢量，将独热特征矢量的特征组合视为逻辑连接，在本次研究中采用聚合的方式来进行逻辑链接。

本研究中采用的所有特征组合方法包括对某个聚集组合计算其个数，对剧集中的不相等值进行计数，计算每个聚集的累加个数，计算每个聚集组合出现次数的平均值，计算每个聚集组合的方差。此外，还将每个ip的下次点击next_click也作为一个特征，时间维度采用hour。生成如下27个特征：

```
Data columns (total 27 columns):
ip                          uint32
app                         uint16
device                      uint16
os                          uint16
channel                     uint16
is_attributed               uint8
hour                        uint8
X0                          uint8
X1                          uint32
X2                          uint8
X3                          uint16
X4                          uint8
X5                          uint16
X6                          uint8
X7                          uint32
X8                          uint8
A0                          uint32
A1                          uint32
A2                          uint32
ip_tcount                   uint16
ip_app_count                uint32
ip_app_os_count             uint16
ip_tchan_count              float32
ip_app_os_var               float32
ip_app_channel_var_day      float32
ip_app_channel_mean_hour    float32
nextClick                   float32
dtypes: float32(5), uint16(8), uint32(7), uint8(7)
memory usage: 5.5 GB
```

本数据集选取7500 0000条数据，使用27个特征进行输入。经过特征工程后的数据大小为5.5GB。

## 2.5 特征重要性

对于线性模型来说，特征的重要性由W的分量$W_i$体现出来，$|W_i|$越大，说明特征越重要（指得是相对于其他特征）。但非线性模型则困难得多。而随机森林可以用特殊的方法得出特征的重要性。基本原理是：如果特征j比较重要，如果往这个特征上塞入一些噪声，模型的表现肯定会变差。因此，可以对$\{x_{i,j}\}_{i=1}^{m}$做一个重新的排列，这样该特征的分布就不会发生变化。记重排列特征j上的值以后得到的数据集为$D^{(p)}$特征j上的重要性可以通过统计上称为排列测试 (permutation test)的方法进行估计:

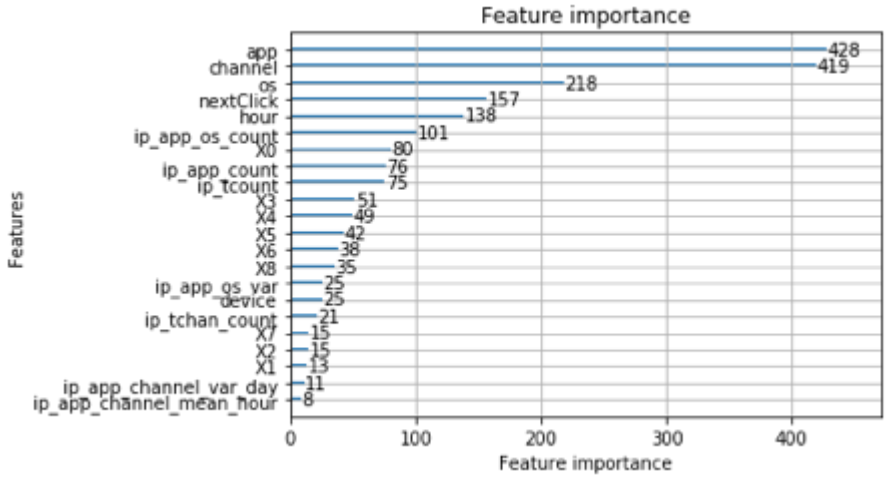
$$importance(j) = performs(D) - performs(D^{(p)})$$

这里，$performs(D^{(p)})$ 需要用重排列后的数据重新训练模型。 不过随机森林可以不用重新训练。前面提到过，可以用袋外估计$E_{oob}$直接估计模型的性能，因此随机森林可以用下面的公式进行估计：

$$E_{oob}(G) = \frac{1}{m}\sum_{i=1}^{m} err(y_i, G_i^{-}(x_i))$$

$$importance(j) = E_{oob}(G) - E_{oob}^{(p)}(G)$$

通过sklearn库函数计算得其特征重要性的前22位如下：


Feature importance
app
channel
os
nextClick
hour
ip_app_os_count
X0
ip_app_count
ip_tcount
X3
X4
X5
X6
X8
ip_app_os_var
device
ip_tchan_count
X7
X2
X1
ip_app_channel_var_day
ip_app_channel_mean_hour
428
419
218
157
138
101
80
76
75
51
49
42
38
35
25
25
21
15
15
13
11
8
Features
0
100
200
300
400
Feature importance

# 第 3 章　模型部分

## 3.1　模型准备

模型选用 Light GBM，以 CART 树作为基分类器。LGBM 模型采用直方图算法，把数值型特征进行分桶（而这些桶称为 bin），同时，将特征根据其所在的 bin 进行梯度累加。这样，遍历一次数据后，直方图累积了需要的梯度信息，然后可以直接根据直方图，寻找最优的切分点。且对目标函数进行泰勒展开，累加了一阶和二阶梯度，同时还累加了梯度的和，还有个数。

LGBM 采用 leaf-wise tree growth，每次从当前所有叶子中，找到分裂增益最大的叶子分裂，leaf-wise 可以降低更多的误差，得到更好的精度。同时 LGBM 还具有特征并行、数据并行、投票并行等优点。其参数设置如下：

```
#设置lgb的参数
MAX_ROUNDS = 2000 #lgb iterations
EARLY_STOP = 50  #lgb early stop
OPT_ROUNDS = 650  #To be adjusted based on best validation rounds
params = {
        'boosting_type': 'gbdt',
        'objective': 'binary',
        'metric':'auc',
        'learning_rate': 0.2, # 【consider using 0.1】
        #'is_unbalance': 'true',  #because training data is unbalance (replaced with scale_pos_weight)
        'scale_pos_weight': 200, # because training data is extremely unbalanced
        'num_leaves': 7,  # we should let it be smaller than 2^(max_depth), default=31
        'max_depth': 3,  # -1 means no limit, default=-1
        'min_data_per_leaf': 100,  # alias=min_data_per_leaf , min_data, min_child_samples, default=20
        'max_bin': 100,  # Number of bucketed bin for feature values,default=255
        'subsample': 0.7,  # Subsample ratio of the training instance.default=1.0, alias=bagging_fraction
        'subsample_freq': 1,  # k means will perform bagging at every k iteration, <=0 means no enable,alias=bagging_fr
        'colsample_bytree': 0.9,  # Subsample ratio of columns when constructing each tree.alias:feature_fraction
        'min_child_weight': 0,  # Minimum sum of instance weight(hessian) needed in a child(leaf),default=1e-3,Like min_
        'subsample_for_bin': 200000,  # Number of samples for constructing bin
        'min_split_gain': 0,  # lambda_l1, lambda_l2 and min_gain_to_split to regularization
        'reg_alpha': 0,  # L1 regularization term on weights
        'reg_lambda': 0,  # L2 regularization term on weights
        'nthread': 4, # should be equal to REAL cores:http://xgboost.readthedocs.io/en/latest/how_to/external_memory.ht
        'verbose': 0
         }

target = 'is_attributed'
predictors = ['nextClick','app','device','os', 'channel', 'hour',
                  'ip_tcount', 'ip_tchan_count', 'ip_app_count',
                  'ip_app_os_count', 'ip_app_os_var',
                  'ip_app_channel_var_day','ip_app_channel_mean_hour',
                  'X0', 'X1', 'X2', 'X3', 'X4', 'X5', 'X6', 'X7', 'X8']
categorical = ['app', 'device', 'os', 'channel', 'hour']
```

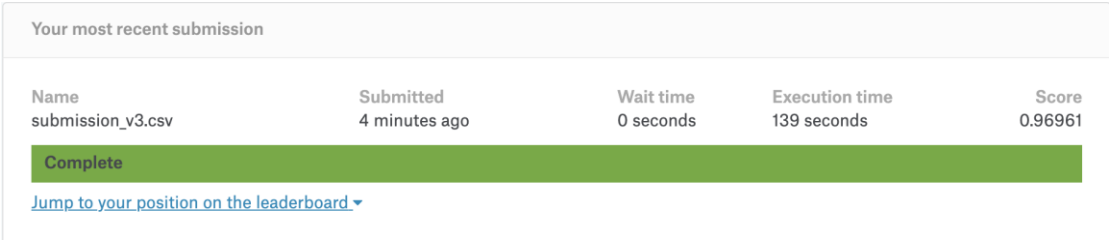
## 3.2　模型训练

模型采用 9:1 划分训练集和验证集，训练结果如下：

```
[50]    valid's auc: 0.966631
[60]    valid's auc: 0.967517
[70]    valid's auc: 0.967673
[80]    valid's auc: 0.967991
[90]    valid's auc: 0.968181
[100]   valid's auc: 0.968716
[110]   valid's auc: 0.968428
[120]   valid's auc: 0.97032
[130]   valid's auc: 0.970717
[140]   valid's auc: 0.970985
[150]   valid's auc: 0.971218
[160]   valid's auc: 0.971551
[170]   valid's auc: 0.971558
[180]   valid's auc: 0.971583
[190]   valid's auc: 0.971681
[200]   valid's auc: 0.971778
[210]   valid's auc: 0.971874
[220]   valid's auc: 0.971947
[230]   valid's auc: 0.971954
[240]   valid's auc: 0.971961
[250]   valid's auc: 0.971962
[260]   valid's auc: 0.972227
[270]   valid's auc: 0.972326
[280]   valid's auc: 0.972349
[290]   valid's auc: 0.972399
[300]   valid's auc: 0.972404
[310]   valid's auc: 0.972445
[320]   valid's auc: 0.972454
[330]   valid's auc: 0.972472
[340]   valid's auc: 0.972565
[350]   valid's auc: 0.972374
[360]   valid's auc: 0.972396
[370]   valid's auc: 0.972107
[380]   valid's auc: 0.971673
[390]   valid's auc: 0.971828
Early stopping, best iteration is:
[340]   valid's auc: 0.972565
[0:19:29.071222]: model training time
```

## 3.3 模型预测

对 test.csv 进行特征工程后，输入模型得到提交结果。结果在 kaggle 平台提交得分为：




【所有文本资料见个人 github：https://github.com/YeSei/TalkingData-AdTracking】