广告检测中的流量作弊识别

互联网虚假流量，是指通过特殊的方式，模仿人类浏览行为生成的访问流量。例如通过设置程序，每分钟访问一次某网站的主页，这样的流量就属于虚假流量。广告主寻找媒体投放广告的目的是将信息传达给目标受众，以此促进销售量。而媒体的责任则是尽可能引导更多的用户浏览这些信息。一般而言，浏览量的增加，的确能促进销售量的增加，因此多数广告主是基于浏览支付广告费的。换句话说：同等条件下，流量大的网站收取的广告费用更高。部分网站受利益的驱使，通过作弊方式产生流量。

本案例将使用广告检测流量数据，分别采用决策树算法和随机森林算法建立分类模型，识别出流量数据中的作弊流量，减少投放广告的客户损失。

# 了解广告流量作弊现状与作弊方式

## 广告流量作弊背景

与传统的电视广告、户外广告采买相比，流量作弊一直以来被看作互联网广告特有的弊病。2016年第一季度Facebook发文称，其Atlas DSP平台半年的流量质量测试结果显示，由机器人模拟和黑IP等手段导致的恶意流量高达75%。仅2016上半年，AdMaster反作弊解决方案认定平均每天能有高达28%的作弊流量。低质量虚假流量的问题一直存在，这也是过去十年间数字营销行业一直在博弈的问题。基于AdMaster海量监测数据，50%以上的项目均存在作弊嫌疑；不同项目中，作弊流量占广告投放5%到95%不等；其中垂直类（集中在某些特定的领域或某种特定的需求）和网盟类（网站的广告联盟，是精准投放广告的一种）媒体的作弊流量占比最高；PC端作弊流量比例显著高于移动端和智能电视平台。

虚假流量的存在给广告主带来了严重的损失。一方面虚假流量提高了广告费用，直接损害了广告主的利益。另一方面，广告监测行为数据被越来越多地用于建模和做决策，例如绘制用户画像，跨设备识别对应用户等。作弊行为，恶意曝光，甚至是在用户完全无感知的情况下被控制访问等产生的不由用户主观发出的行为给数据带来了巨大的噪声，给模型训练造成了很大影响。

本案例基于海量广告流量检测数据，对原始数据采用无放回随机抽样法抽取7天的流量记录作为原始建模数据。

本次数据挖掘建模目标如下：

建立互联网虚假流量识别模型，精准识别虚假作弊流量记录。

## 常见广告流量作弊方式

常见的几种流量作弊方式如表 1‑1所示。

表 1‑1流量作弊方式说明

|  |  |
| --- | --- |
| **流量作弊方式** | **说明** |
| 脚本刷量 | 通过设定程序，使电脑按一定的规则访问目标网站。 |
| 控制肉鸡访问 | 利用互联网上受病毒感染的电脑访问目标网站。 |
| 页面代码修改 | 通过病毒感染或其他方式，在媒体网站插入隐藏代码，在其页面加载肉眼不可见的指向目标网站的小页面。 |
| DNS劫持 | 通过篡改DNS服务器上的数据，强制修改用户电脑的访问位置，使原本访问网站被动修改为目标网站。 |

流量作弊者通过各项技术，不断模拟人的行为，增大识别作弊流量的难度。例如：控制分时间段的IP访问量，使用正常的UA（User-Agent，用户代理商），控制在页面曝光的时间，及访问的路径等。访问流量通过上述手段的处理，虽然识别难度增大了，但并不意味着作弊流量是不可识别的。机器模拟的流量是通过软件实现，必定与人类的点击流量存在一定的差异。

## 了解流量作弊识别的步骤与流程

本次数据挖掘建模的总体流程如图 1‑1所示。广告检测中的流量作弊识别主要包括以下步骤：

1. 对广告检测中获得的历史流量数据进行选择性抽取；
2. 对（1）形成的数据集进行数据探索分析和预处理，包括探索流量作弊的行为特征、冗余属性识别、流量数据的缺失值、数据变换等，并进行相应地预处理，根据建模的需要进行属性构造等；
3. 在（2）得到的数据基础上，构建建模样本；
4. 建立虚假流量识别模型，虚假流量识别模型进行模型分析评价。
5. 对（4）形成的模型结果应用。



图 1‑1虚假流量识别流程

# 探索分析广告流量检测数据

## 数据说明与存储

在广告检测中，每一秒都会采集一条或多条状态数据。由于采集频率较高，数据总量非常大。本案例对原始数据采用无放回随机抽样法抽取7天的流量记录作为原始建模数据。

广告检测中的流量数据包含以下22个属性：rank、dt、cookie、ip、idfa、imei、android、openudid、mac、timestamps、camp、creativeid、mobile\_os、mobile\_type、app\_key\_md5、app\_name\_md5、placementid、useragent、mediaid、os\_type、born\_time、label。数据示例及说明如表 2‑1所示。

表 2‑1流量数据监测表

|  |  |  |  |
| --- | --- | --- | --- |
| **属性名称** | **中文名称** | **示例** | **备注** |
| rank | 记录序号 | 5（第5条记录） |  |
| dt | 相对日期 | 3（第3天） | 单位为天 |
| cookie | cookie值 | 7083a0cba2acd512767737c65d5800c8 |  |
| ip | IP地址 | 101.52.165.247 | 经过脱敏 |
| idfa | idfa值 | bc50cc5fb39336cf39e3c9fe1b16bf48 | 可用于识别IOS用户 |
| imei | imei值 | 990de8af5ed0f3744b61770173794555， | 可用于识别Android用户 |
| android | android值 | 7730a40b70cf9b023d23e332da846bfb | 可用于识别Android用户 |
| openudid | openudid值 | 7aaeb5d6af25f9fe918ec39b0f79a2c8 | 可用于识别IOS用户 |
| mac | mac值 | 6ed9fcefd06a2ab5f901e601a3a53a2d | 可用于识别不同硬件设备 |
| timestamps | 时间戳 | 0（记录于数据区间的初始时间点） |  |
| camp | 项目ID | 61520 |  |
| creativeid | 创意ID | 0 |  |
| mobile\_os | 设备OS版本信息 | 5.0.2 | 该值为原始值 |
| mobile\_type | 机型 | 'Redmi+Note+3'（设备为红米Note3） |  |
| app\_key\_md5 | app key信息 | ffe435bdb6ce18dd4758c0005c4787db |  |
| app\_name\_md5 | app name信息 | 6f569b4fa576d25fb98e60bda9c97426 |  |
| placementid | 广告位信息 | 72ee620530c7c8cd4b423d4b4502b45b |  |
| useragent | 浏览器信息 | "Mozilla%2f5.0%20%28compatible%3b%20MSIE%209.0%3b%20Windows%20NT%206.1%3b%20Trident%2f5.0%29%20Fengxing%2f3.0.3.77%20MZ%2f75B00973C5D899C8BA4858F5E4FAA59B" |  |
| mediaid | 媒体id信息 | 1118 |  |
| os\_type | OS类型标记 | 0（采集到的OS类型标记为0） |  |
| born\_time | cookie生成时间 | 160807（第160807日） |  |
| label | 作弊标签 | 0 | 1为作弊 |

项目所提供的建模数据已经标记了相关流量数据是否作弊的标签，然而目标网站在收集流量数据的时候是没有类别标签的，所以应该处理一份没有标签的数据，与原始的网站数据一致，以便在后期用于模型应用，更加贴合实际生产环境。

项目所获取的数据是业务人员从网站MySQL数据库导出的CSV格式的数据，总共有22个字段的值，命名为case\_data\_new.csv。项目主要采用Spark技术进行数据探索、数据处理、建模与分析的过程，因为Spark在大数据处理效率方面具有很大的优势，另一个方面就是Spark拥有非常丰富的机器学习算法库，可以简化复杂的建模实现过程。并且通过Spark可以读取MySQL、Hive、HBase等多种存储系统的数据。目前案例的数据相对较少，可以直接将数据存储到MySQL，通过Spark读取进行处理，但是如果检测数据量比较多的情况下，将数据传输到基于Hadoop的大数据仓库Hive进行存储更加合适。

考虑到后期的数据情况，直接将CSV格式的文件数据导入到Hive数据库。首先需要创建相关Hive表，如代码 2‑1所示。

代码 2‑1 创建Hive数据库和表

|  |
| --- |
| create database ad\_traffic;  CREATE TABLE `case\_data\_sample\_tmp` (  `rank` int,  `dt` int,  `cookie` string,  `ip` string,  `idfa` string,  `imei` string,  `android` string,  `openudid` string,  `mac` string,  `timestamps` int,  `camp` int,  `creativeid` int,  `mobile\_os` int,  `mobile\_type` string,  `app\_key\_md5` string,  `app\_name\_md5` string,  `placementid` string,  `useragent` string,  `mediaid` string,  `os\_type` string,  `born\_time` int,  `label` int ) row format serde 'org.apache.hadoop.hive.serde2.OpenCSVSerde' with SERDEPROPERTIES  ("separatorChar"=",","quotechar"="\"")  STORED AS TEXTFILE;  CREATE TABLE `case\_data\_sample` (  `rank` int,  `dt` int,  `cookie` string,  `ip` string,  `idfa` string,  `imei` string,  `android` string,  `openudid` string,  `mac` string,  `timestamps` int,  `camp` int,  `creativeid` int,  `mobile\_os` int,  `mobile\_type` string,  `app\_key\_md5` string,  `app\_name\_md5` string,  `placementid` string,  `useragent` string,  `mediaid` string,  `os\_type` string,  `born\_time` int,  `label` int ) row format delimited fields terminated by ', '; |

代码 2‑1创建了Hive表case\_data\_sample和case\_data\_sample\_tmp，case\_data\_sample\_tmp用于导入CSV文件的数据，再将case\_data\_sample\_tmp表的数据复制到case\_data\_sample。之所以需要经过这个转储的过程，是因为case\_data\_sample\_tmp表导入CSV的数据后，字段类型会全部变成String类型，不利于后续数据挖掘。

接下来将CSV数据csv\_data\_new.csv上传到Linux节点，通过命令“sed -i 1d case\_data\_new.csv”删除文件首行的列名。将数据导入case\_data\_sample\_tmp表，如代码 2‑2所示。

代码 2‑2导入数据到case\_data\_sample\_tmp表

|  |
| --- |
| load data local inpath '/root/case\_data\_new.csv' into table case\_data\_sample\_tmp; |

接下来将case\_data\_sample\_tmp表的数据导入case\_data\_sample，如代码 2‑3所示。

代码 2‑3 导入数据到case\_data\_sample表

|  |
| --- |
| insert overwrite table case\_data\_sample select \* from case\_data\_sample\_tmp; |

按照之前的分析，需要生成一份未进行类别标识的样本数据，以案例样本数据为基础，生成没有类别标签的样本数据，如代码 2‑4所示。

代码 2‑4 创建表case\_data

|  |
| --- |
| create table case\_data as select rank,dt,cookie,ip,idfa,imei,android,openudid,mac,timestamps,  camp,creativeid,mobile\_os,mobile\_type,app\_key\_md5,app\_name\_md5,placementid,useragent,mediaid,  os\_type,born\_time from case\_data\_sample; |

可以在Hive中查看表的数据情况，如图 2‑1所示。

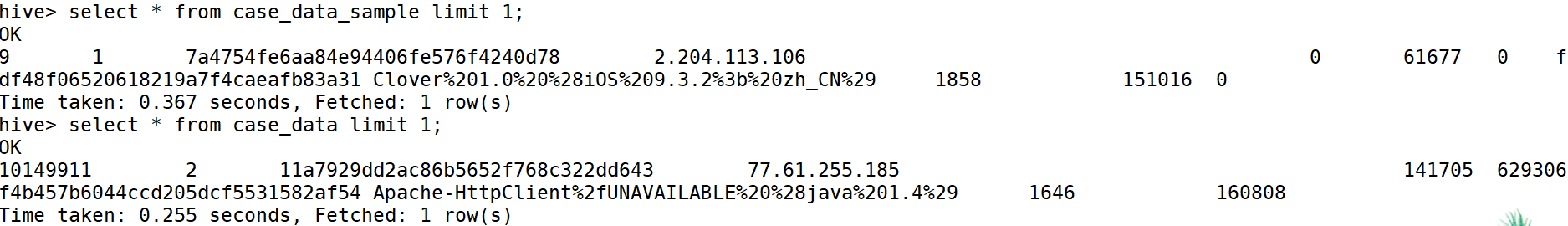


图 2‑1 流量数据样本

## 探索缺失值与冗余属性

本案例主要从基础数据探索、数据质量分析和特征数据分析三个方面进行数据探索分析。

### 基础数据探索

基础数据探索是数据挖掘的基础工作。在一开始拿到数据的时候，对于数据一般都处于知之甚少的状态或是从业务人员处得到一些关于数据的基本信息，然而这些基本信息也是未经验证的，具体数据是否与业务人员描述一致还未可知。因此在拿到项目数据的时候，需要对数据进行基本的探索工作，基本探索的内容一般就是数据的记录数、数据的时间分布等。

#### 记录数探索

案例的原始数据是从海量的流量数据中抽取出的7天的数据量，提供的建模数据CSV文件大小为551M，数据已导入Hive表case\_data\_sample。首先第一步就是需要探索这7天的数据量，如代码 2‑5所示，数据探索的代码皆在Spark Shell中执行。

代码 2‑5 统计数据记录数

|  |
| --- |
| //统计记录数  scala> val num=spark.sqlContext.read.table("ad\_traffic.case\_data\_sample").count  num: Long = 1704154 |

7天总共有1704154条流量数据，数据量并不是很多，接下来查看每天的流量数据，是否存在异常的情况。

#### 日流量探索

广告检测流量数据记录总共有7天，在样本数据中，通过字段dt记录了流量数据提取的相对时间。字段的值为1~7,1表示提取的7天流量数据的第一天数据，以此类推。对每天的数据流量数进行统计，查看是否有异常现象，如代码 2‑6所示。

代码 2‑6 统计数据日流量

|  |
| --- |
| //统计日流量  scala>val dt="dt"  scala>val day\_num = spark.read.table("ad\_traffic.case\_data\_sample").groupBy(dt).agg(count(dt) as "daycount")  scala>day\_num.show(false)  +---+--------+  |dt |daycount|  +---+--------+  |1 |222665 |  |6 |251982 |  |3 |273309 |  |5 |244887 |  |4 |233976 |  |7 |153823 |  |2 |323512 |  +---+--------+ |

从代码 2‑6统计结果可以看出，数据的日流量差异不大，说明并没有存在数据异常增加或减少的情况，数据产生的环境相对稳定。

### 数据质量分析

数据质量分析是数据预处理的前提，也是数据挖掘分析结论有效性和准确性的基础，主要任务是检查原始数据中是否存在脏数据，脏数据一般是指不符合要求，以及不能直接进行分析的数据，包括：缺失值、不一致值等。

#### 数据类型分析

#### 缺失数据分析

在很多的数据样本中，经常会存在一些具有缺失值的字段。如果对挖掘过程没有影响的字段缺失值可以不考虑，但是如果在挖掘过程中，需要使用具有缺失值的字段且字段的缺失值会影响挖掘的过程，就需要对缺失值进行进一步的处理。对7天所有的广告检测中的流量数据进行缺失值探索，统计出各个属性的数量情况，如图 2‑2所示。

图 2‑2属性缺失率折线图

对cookie、ip、idfa等20个属性1704154条流量数据进行了统计，计算出了各属性的缺失率，结果如图 2‑2所示。从图中可以看出，mac、creativeid、mobile\_os、mobile\_type、app\_key\_md5、app\_name\_md5、os\_type、idfa、imei、android、openudid等属性的缺失率非常高，尤其是creativeid属性，高达98.39%，由于原始数据为字符型数据，无法进行插补，后续编写程序，对部分缺失过多的属性进行删除操作。

#### 冗余数据分析

## 探索作弊流量数据特征

在表 1‑1中列出了常见的几种流量作弊方式，不同作弊行为产生的数据特征不同，对虚假流量数据探索分析，将虚假流量根据作弊行为进行划分，为构建特征指标提供保障。

### 脚本刷新网页作弊

脚本刷新网页作弊是通过设定程序，使电脑按一定的规则访问目标网站。该作弊方式产生的数据记录中cookie与ip不变，且存在多条记录，如图 2‑3所示。



图 2‑3脚本刷新网页作弊行为数据（部分）

从图 2‑3可知，某用户（ip为44.75.99.61，cookie为646d9cd31ae2a674d1ed6d68acc6e019）在第1天利用同一浏览器（useragent：Mozilla）多次访问某一网址（mediaid：151）上的广告（placementid：e886beb3cc63365cf71e1ae19aae60ea），经过统计，产生了1206条cookie和ip不变的数据记录，该用户的行为属于利用脚本刷新网页的作弊行为，所产生的流量数据为虚假流量。

对cookie和ip不变的数据记录进行分组统计，探索cookie和ip不变的数据记录统计次数分布情况，如代码 2‑7所示。

代码 2‑7 统计cookie和ip相同的流量记录数分布

|  |
| --- |
| //统计cookie和ip不变的记录统计次数分布情况  scala> val cookie\_ip\_distribute = spark.read.table("ad\_traffic.case\_data\_sample").groupBy(col("ip")).agg(count("ip") as "cookie\_ip\_count").groupBy("cookie\_ip\_count").agg(count("cookie\_ip\_count") as "cookie\_ip\_count\_count",count("cookie\_ip\_count")/1704154\*100 as "cookie\_ip\_count\_count\_precent").orderBy(desc("cookie\_ip\_count"))  cookie\_ip\_distribute: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [cookie\_ip\_count: bigint, cookie\_ip\_count\_count: bigint ... 1 more field]  scala> cookie\_ip\_distribute.show(false)  +---------------+---------------------+-----------------------------+  |cookie\_ip\_count|cookie\_ip\_count\_count|cookie\_ip\_count\_count\_precent|  +---------------+---------------------+-----------------------------+  |11718 |1 |5.8680142757051305E-5 |  |6591 |1 |5.8680142757051305E-5 |  |5970 |1 |5.8680142757051305E-5 |  |5836 |1 |5.8680142757051305E-5 |  |5156 |1 |5.8680142757051305E-5 |  |5135 |1 |5.8680142757051305E-5 |  |5103 |1 |5.8680142757051305E-5 |  |4833 |1 |5.8680142757051305E-5 |  |4828 |1 |5.8680142757051305E-5 |  |4654 |1 |5.8680142757051305E-5 |  |4599 |1 |5.8680142757051305E-5 |  |4516 |1 |5.8680142757051305E-5 |  |4469 |1 |5.8680142757051305E-5 |  |4439 |2 |1.1736028551410261E-4 |  |4416 |1 |5.8680142757051305E-5 |  |4412 |1 |5.8680142757051305E-5 |  |4366 |1 |5.8680142757051305E-5 |  |4344 |1 |5.8680142757051305E-5 |  |4340 |1 |5.8680142757051305E-5 |  |4324 |1 |5.8680142757051305E-5 |  +---------------+---------------------+-----------------------------+  only showing top 20 rows |

从代码 2‑7统计结果可以看出，在7天的流量数据中，确实存在同一ip和cookie高频点击广告的情况，对点击频数超过1000次的ip和cookie数进行一个简单的统计，如代码 2‑8所示。

代码 2‑8 统计点击频数超过1000次的IP和Cookie数

|  |
| --- |
| scala> val click\_gt\_1000 = cookie\_ip\_distribute.filter("cookie\_ip\_count>1000").select(sum(col("cookie\_ip\_count\_count")))  click\_gt\_1000: org.apache.spark.sql.DataFrame = [sum(cookie\_ip\_count\_count): bigint]  scala> click\_gt\_1000.show(false)  +--------------------------+  |sum(cookie\_ip\_count\_count)|  +--------------------------+  |331 |  +--------------------------+ |

代码 2‑8统计结果显示，7天中同一个ip和cookie点击广告超过1000的就有331个，各自对应的记录数相加，将占据大量的流量记录，如果不加以识别，将会对广告主造成很大损失。

# 预处理广告流量检测数据

## 处理缺失数据

## 构建广告流量作弊识别模型关键特征

## 进行数据标准化

## 构建建模样本

# 使用决策树算法进行作弊流量识别

## 了解决策树算法

## 模型构建

## 模型评估

# 使用随机森林算法进行作弊流量识别

## 了解随机森林算法

## 模型构建

# 模型应用

经过模型评估，最终选择随机森林算法构建的分类器。接下来需要将分类器应用到虚假流量识别中。

# 小结

本案例根据案例需求与目标，对流量检测数据进行了数据探索、数据处理、模型构建、模型评估等。模型构建采取了两种算法（决策树和随机森林）构建虚假流量识别模型，通过模型评价可知，随机森林更加适合于该案例。因此本案例最终构建基于随机森林算法的虚假流量识别模型，该模型将应用到虚假流量的检测当中，减少通过虚假流量谋取利益的不当行为。