第7章 项目案例-广告检测的流量作弊识别

(源自:https://biglab.site)

(版本:Ver1.0-20231014)

任务背景

与传统的电视广告、户外广告采买相比，虚假流量一直被看作互联网广告特有的弊病。互联网广告虚假流量，是指通过特殊的方式，模仿人类浏览行为而生成的访问流量。

如通过设置程序，每分钟访问一次某网站的主页，这样的流量即属于虚假流量。

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，以促进相关产品的销售。而媒体的责任则是尽可能引导更多的用户浏览这些信息。同等条件下，流量大的网站收取的广告费用更高。

因此，部分网站受利益的驱使，会通过作弊方式产生虚假广告流量。

虚假广告流量的问题在数字营销行业中一直存在，给广告主带来了严重的损失。但互联网不是法外之地，利用大数据及人工智能技术构筑网络安全屏障，一方面能够保障国家安全；一方面能够提高市场效率，消除网络垃圾，惩治网络违法行为。目前，广告监测行为数据被越来越多地用于建模和做决策，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，甚至是在用户毫无感知的情况下被控制访问等非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。

本章将通过Spark大数据技术实现广告检测的流量作弊识别，使读者可以更加熟悉Spark相关技术，并灵活应用相关技术解决相应的大数据问题。

任务7.1分析需求

任务描述

任何解决方案都需要从需求入手，分析实现目标所需要进行的步骤。本节的任务如下。了解在互联网中常见的广告流量作弊方式；结合广告检测的流量作弊识别案例的目标，分析案例的需求。

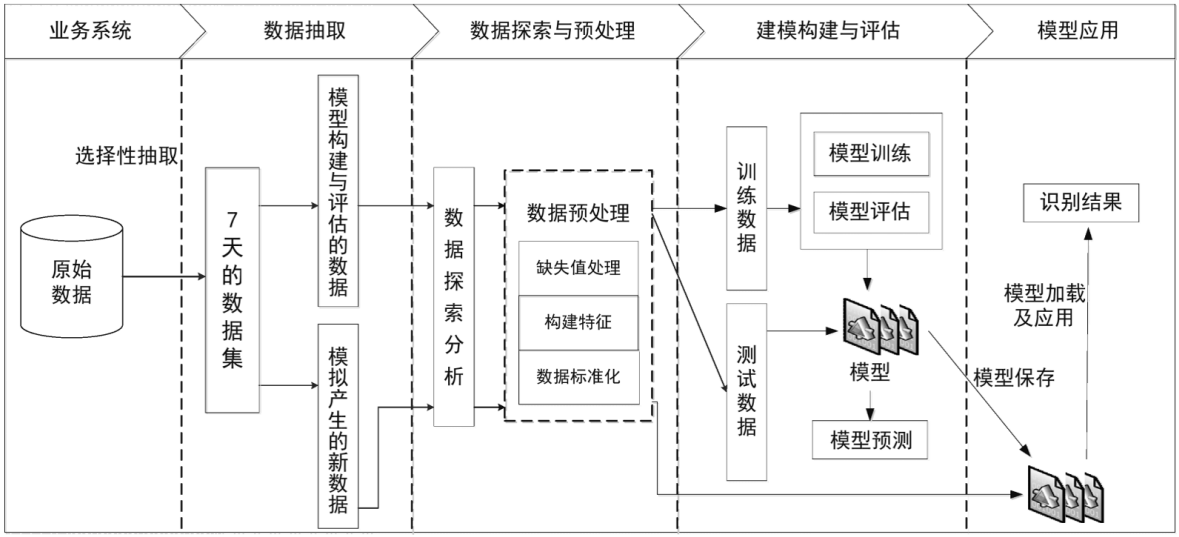
互联网时代的核心之一是流量，更多的流量意味着更多的关注和可能的更高的收入。广告主在互联网投放广告时往往会依据流量信息来设计投放方案，广告流量作弊不仅仅会使广告主选择错误的广告投放方案，造成浪费，也会使后期根据用户浏览信息对现有广告进行修改时出现偏差。这些问题常常会引发“蝴蝶效应”，造成不可估量的损失。因此，对广告流量进行作弊检测进而加以防范是非常有必要的。广告的浏览信息数据量往往十分庞大，人工对其进行筛选很不现实，所以一般会通过算法对海量浏览信息进行自动化筛选。

常见的流量作弊方式

广告流量作弊方式	说明
脚本刷量	设定程序，使计算机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计算机访问目标网站
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，强制修改用户计算机的访问位置，使用户原本访问的网站被修改为目标网站

分析需求

根据目标对广告检测的流量作弊识别的整体实现流程进行拆分，如下图。



广告检测的流量作弊识别实现流程的步骤如下。

1. 对广告检测获得的历史流量数据进行选择性抽取和数据划分。
2. 对第（1）步中形成的数据集进行数据探索分析，包括缺失值、冗余字段的基础探索和流量作弊的行为特征的业务探索。
3. 根据探索分析结果得出的清洗规则，对数据进行相应的预处理，并构建建模时需要的特征，形成建模样本数据。
4. 建立不同的虚假流量识别模型，并对模型进行评估及对比。
5. 保存效果较好的模型，模拟新数据产生，加载保存好的模型以进行应用。

任务7.2.探索分析广告流量数据

任务描述

不同的作弊行为产生的数据特征不同，对数据进行探索分析并合理归纳虚假流量的数据特征，为后期有针对性地对数据进行预处理、构建相应的指标提供可靠依据，可有效提高模型分类的准确率。

本节的任务如下。

1. 对广告流量数据进行探索分析，包括对数据说明、数据记录数、日访问流量等的基础探索；

2. 对作弊浏览的特征探索。

数据准备

通过百度云下载数据文件

链接：https://pan.baidu.com/s/1FF7HHQi9y2kVBXom_Omd_g
提取码：8ft4

数据列表：

软件	版本	安装包称
Spark_data_max	3.2.1	spark_data_max.tar.gz

将spark_data_max.tar.gz下载到d:/spark目录，然后将文件解压到当前路径，解压后就有文件：

```
D:\spark\spark_data_max\case_data_new.csv
```

数据说明

广告流量数据,如文件:D:\spark\spark_data_max\case_data_new.csv

字段名称	说明
rank	记录序号
dt	相对日期，单位为天
cookie	cookie值
ip	IP地址，已脱敏
idfa	IDFA值，可用于识别iOS用户
imei	IMEI值，可用于识别Android用户
android	Android值，可用于识别Android用户
openudid	OpenUDID值，可用于识别iOS用户
mac	MAC值，可用于识别不同硬件设备
timestamps	时间戳
camp	项目ID
creativeid	创意ID
mobile_os	设备OS版本信息，该值为原始值
mobile_type	机型
app_key_md5	App Key信息
app_name_md5	App Name信息
placementid	广告位信息
useragent	浏览器信息
mediaid	媒体ID信息
os_type	OS类型标记
born_time	cookie生成时间
label	作弊标签，1表示作弊，0表示正常

基础探索数据

可以探索记数数、日流量、字段缺失值、冗余数据

探索作弊流量的数据特征

如脚本刷新网页作弊、定期清除cookie，刷新网页作弊、ADSL重新拨号后刷新网页作弊

初始化创建库ad_traffic

```
package chap07

import org.apache.spark.sql.SparkSession

object s1_InitDb {
  def main(args: Array[String]): Unit = {
    val spark = SparkSession.builder().enableHiveSupport()
      .master("local[4]").appName("Explore").getOrCreate()

    spark.sql("create database ad_traffic;")
  }

}

```

探索数据源码参考

```
package chap07

import org.apache.spark.sql.functions._
import org.apache.spark.sql.{DataFrame, SaveMode, SparkSession}
object s2_Explore {
  def main(args: Array[String]): Unit = {
    val spark = SparkSession.builder().enableHiveSupport()
      .master("local[4]").appName("Explore").getOrCreate()

    spark.sparkContext.setLogLevel("WARN") //WARN,ERROR
    // 读取数据
    val rawData =
spark.read.option("header","true").csv("D:\\spark\\spark_data_max\\case_data
_new.csv")
    // 统计记录数
    println("原始数据集行数为：" + rawData.count())

    // 统计日流量
    rawData.groupBy("dt").count().selectExpr("dt","count as
dayCount").sort("dt").show()

    // 获取列名并存为List中
    val columnName = rawData.columns.toList
    // 计算数据字段缺失值
    for (i <- columnName){
      MissingCount(rawData,i)
    }

    def MissingCount(data:DataFrame,columnName:String): Unit ={
      if (columnName != "creativeid") {
        val missingRate = data.select(
          columnName).na.drop().count().toDouble / data.count()
        println(columnName+" 缺少值比率：" + (1-missingRate)*100 + "%")
```

```
      }
      else{
        val creativeidMissing = data.select(columnName).filter(
          "creativeid ==0").count() / data.count().toDouble
        println(columnName+" 缺少值比率：" + creativeidMissing*100+"%")
      }
    }

    // 脚本刷新网页作弊
    // 统计cookie和ip相同的流量记录数
    val cookie_ip_distribute = rawData.groupBy(
      "ip","cookie").count().withColumn("ip_cookie_count_precent", col(
      "count") / rawData.count()*100).orderBy(desc("count"))
    cookie_ip_distribute.show(false)

    // 统计同一个ip和cookie的浏览次数超过100的记录数
    val click_gt_100 = cookie_ip_distribute.filter("count > 100").count()
    println("同ip、cookie出现超过100次以上的记录数：" + click_gt_100)

    // 定期清除cookie，刷新网页作弊
    // 统计每个ip对应的不同cookie次数的分布情况
    val ip_distribute = rawData.groupBy("ip").agg(
      countDistinct("cookie") as "ip_count").groupBy("ip_count").agg(count(
      "ip_count") as "ip_count_count", count(
      "ip_count") / rawData.count()*100 as
"ip_count_count_precent").orderBy(
      desc("ip_count"))
    ip_distribute.show(false)

    // ADSL重新拨号后刷新网页作弊
    // 统计ip前两段相同的记录数的分布情况
    val ip_two = rawData.withColumn("ip_two",substring_index(
      col("ip"), ".", 2)).groupBy("ip_two").agg(
      count("ip_two") as "ip_two_count").orderBy(desc("ip_two_count"))

    ip_two.show(false)


    // 统计ip前3段相同的记录数的分布情况
    val ip_three = rawData.withColumn(
      "ip_three",substring_index(col("ip"), ".",
3)).groupBy("ip_three").agg(
      count("ip_three") as "ip_three_count").orderBy(desc("ip_three_count"))

    ip_three.show(false)


    // 删除缺失字符
    val data_new = rawData.drop("mac").drop("creativeid").drop(
      "mobile_os").drop("mobile_type").drop("app_key_md5").drop(
      "app_name_md5").drop("os_type")

    data_new.write.mode(SaveMode.Overwrite).saveAsTable("ad_traffic.AdData")
  }

```

```
}

```

在IDEA控制台运行结果：

```
"C:\Program Files\Java\jdk1.8.0_281\bin\java.exe" "-javaagent:D:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2023.1\lib\idea_rt.jar=55018:D:\Program Files\JetBrains\IntelliJ IDEA Community......
原始数据集行数为：1704154
+---+--------+
| dt|dayCount|
+---+--------+
|  1|  222665|
|  2|  323512|
|  3|  273309|
|  4|  233976|
|  5|  244887|
|  6|  251982|
|  7|  153823|
+---+--------+

rank 缺少值比率：0.0%
dt 缺少值比率：0.0%
cookie 缺少值比率：0.0%
ip 缺少值比率：0.0%
idfa 缺少值比率：92.19213756503227%
imei 缺少值比率：79.83116549325942%
android 缺少值比率：80.78859070248346%
openudid 缺少值比率：84.14450806675923%
mac 缺少值比率：78.82843921382691%
23/10/14 10:33:05 WARN ProcfsMetricsGetter: Exception when trying to compute pagesize, as a result reporting of ProcessTree metrics is stopped
timestamps 缺少值比率：0.0%
camp 缺少值比率：0.0%
creativeid 缺少值比率：98.38905404089067%
mobile_os 缺少值比率：80.23365259243003%
mobile_type 缺少值比率：77.39617428941281%
app_key_md5 缺少值比率：79.96577774074409%
app_name_md5 缺少值比率：80.5729411778513%
placementid 缺少值比率：0.0%
useragent 缺少值比率：4.350839184721567%
mediaid 缺少值比率：0.0%
os_type 缺少值比率：67.29080822507825%
born_time 缺少值比率：0.0%
label 缺少值比率：0.0%
+---------------+--------------------------------+-----+--------------------+
|ip             |cookie                          |count|ip_cookie_count_precent|
+---------------+--------------------------------+-----+--------------------+
|44.75.99.61    |646d9cd31ae2a674d1ed6d68acc6e019|1208 |0.07088561245051797 |
```

```
|24.147.126.192 |8ce10846ee66427f8527780107aa200f|1109 |0.06507627831756989
   |
|167.229.153.163|07de748dce9b1368c5b5130955ce7409|934  |0.05480725333508592
   |
|236.197.41.7   |32eefef058200011e763d5e4fc29a8ec|632  |0.03708585022245642
   |
|225.238.2.77   |a153835d4e80dd2669683f674ca708be|406
|0.023824137959362827   |
|2.90.65.111    |869b7e061094fad43425ebe6cbd70dde|369  |0.02165297267735193
   |
|61.222.188.99  |8d3a4618fb892ac933e96cfe963aa337|354  |0.02077277053599616
   |
|228.145.228.70 |4cba9c30eefdecf92543e19bd1caa052|350
|0.020538049964967955   |
|212.192.121.67 |9016201fad855101dd6c496dd1ccb9bc|349  |0.0204793698222109
    |
|228.77.242.25  |cf8d9f33570ea11c9cecc64194e81da6|336
|0.019716527966369236   |
|97.200.72.4    |c9cea0a377a252f5fcf57f5028c852b9|335
|0.019657847823612185   |
|38.24.247.161  |8920d2e178764afece5a7b0537772a81|327
|0.019188406681555775   |
|43.138.126.148 |d27ca52e9a76c2df1d4de6f87bfb594a|321
|0.018836325825013468   |
|228.65.20.94   |062bda023791ebb237d4b74594eef1af|321
|0.018836325825013468   |
|206.173.197.159|efcec1dac79ccd5ef196a0a72c6af866|303
|0.017780083255386544   |
|190.21.41.53   |ea55bf69c4053b57d50684cd28e5fafc|302
|0.017721403112629493   |
|101.188.77.160 |1dde6f28fa24f9aec73378030e766fa0|290  |0.01701724139954488
   |
|218.47.16.196  |92dbed09e2bb1b424d8fac3e1ebd7a4c|271
|0.015902318687160903   |
|218.13.60.59   |68a08e3a107c79ed4d96414eb7e7fd03|266
|0.015608917973375646   |
|38.69.182.243  |14d3a71fdd2ed6d81afab16fb620ed1e|252
|0.014787395974776926   |
+---------------+--------------------------------+-----+-------------------
----+
only showing top 20 rows

同ip、cookie出现超过100次以上的记录数：104
+--------+--------------+----------------------+
|ip_count|ip_count_count|ip_count_count_precent|
+--------+--------------+----------------------+
|10046   |1             |5.8680142757051305E-5 |
|6173    |1             |5.8680142757051305E-5 |
|5442    |1             |5.8680142757051305E-5 |
|5135    |1             |5.8680142757051305E-5 |
|4256    |1             |5.8680142757051305E-5 |
|4233    |1             |5.8680142757051305E-5 |
|4220    |1             |5.8680142757051305E-5 |
|4157    |1             |5.8680142757051305E-5 |
|4127    |1             |5.8680142757051305E-5 |
```

```
|4066    |1             |5.8680142757051305E-5 |
|4027    |1             |5.8680142757051305E-5 |
|3942    |1             |5.8680142757051305E-5 |
|3488    |1             |5.8680142757051305E-5 |
|3377    |1             |5.8680142757051305E-5 |
|3356    |1             |5.8680142757051305E-5 |
|3291    |1             |5.8680142757051305E-5 |
|3290    |1             |5.8680142757051305E-5 |
|3260    |1             |5.8680142757051305E-5 |
|3251    |1             |5.8680142757051305E-5 |
|3236    |1             |5.8680142757051305E-5 |
+--------+--------------+----------------------+
only showing top 20 rows

+-------+------------+
|ip_two |ip_two_count|
+-------+------------+
|190.21 |161154      |
|220.19 |117661      |
|38.24  |103438      |
|24.228 |76513       |
|246.124|75024       |
|38.69  |71825       |
|236.217|61955       |
|61.197 |48460       |
|166.115|44679       |
|212.37 |36773       |
|2.90   |35052       |
|220.251|30445       |
|97.200 |28189       |
|246.188|27218       |
|61.131 |26718       |
|197.10 |24791       |
|225.112|22799       |
|222.149|20901       |
|218.13 |18649       |
|38.239 |18313       |
+-------+------------+
only showing top 20 rows

+-----------+--------------+
|ip_three   |ip_three_count|
+-----------+--------------+
|38.24.247  |99894         |
|190.21.238 |82338         |
|24.228.119 |57354         |
|246.124.11 |41771         |
|212.37.105 |34408         |
|61.197.48  |32434         |
|190.21.41  |32032         |
|38.69.59   |29771         |
|190.21.130 |26933         |
|2.90.198   |25022         |
|220.251.245|20805         |
|97.200.72  |19264         |
```

```
|24.228.179 |18873         |
|166.115.204|18390         |
|166.115.71 |17977         |
|166.6.39   |16879         |
|236.217.34 |16788         |
|38.69.182  |16390         |
|190.21.123 |16377         |
|218.13.60  |16059         |
+-----------+--------------+
only showing top 20 rows

23/10/14 10:33:26 WARN HiveConf: HiveConf of name hive.stats.jdbc.timeout does not exist
23/10/14 10:33:26 WARN HiveConf: HiveConf of name hive.stats.retries.wait does not exist
23/10/14 10:33:27 WARN ObjectStore: Version information not found in metastore. hive.metastore.schema.verification is not enabled so recording the schema version 2.3.0
23/10/14 10:33:27 WARN ObjectStore: setMetaStoreSchemaVersion called but recording version is disabled: version = 2.3.0, comment = Set by MetaStore UNKNOWN@192.168.31.206
23/10/14 10:33:32 WARN SessionState: METASTORE_FILTER_HOOK will be ignored, since hive.security.authorization.manager is set to instance of HiveAuthorizerFactory.
23/10/14 10:33:32 WARN HiveConf: HiveConf of name hive.internal.ss.authz.settings.applied.marker does not exist
23/10/14 10:33:32 WARN HiveConf: HiveConf of name hive.stats.jdbc.timeout does not exist
23/10/14 10:33:32 WARN HiveConf: HiveConf of name hive.stats.retries.wait does not exist
23/10/14 10:33:33 WARN ObjectStore: Failed to get database global_temp, returning NoSuchObjectException

Process finished with exit code 0

```

任务7.3预处理数据并构建特征

任务描述

在上一小节的数据探索分析中，了解到一些数据字段存在大量的缺失值，同时一些字段为说明性数据字段，不足以直接作为特征进行训练并构建模型。

本节的任务如下。

1. 根据上一小节的探索分析结果对数据进行处理，删除缺失率较高的数据字段；
2. 构建相应的新特征；对特征进行数据标准化。

删除缺失字段

为了减小缺失数据对模型产生的影响，删除缺失率过高的mac、creativeid、mobile_os、mobile_type、app_key_md5、app_name_md5、os_type等字段。删除缺失率高的字段。将处理后的数据保存至Hive中。先在Hive中创建数据库ad_traffic，使用saveAsTable()方法将处理后的数据保存至Hive的ad_traffic数据库中，表名为AdData，通过mode()方法设置保存模式为覆盖保存。（已含在上节源码：Explore.scala)

```
// 删除缺失字符
   val data_new = rawData.drop("mac").drop("creativeid").drop(
     "mobile_os").drop("mobile_type").drop("app_key_md5").drop(
     "app_name_md5").drop("os_type")

   data_new.write.mode(SaveMode.Overwrite).saveAsTable("ad_traffic.AdData")
```

构建广告流量作弊识别特征

特征	构建方法	说明
N	统计在5小时内，原始数据集中，同一ip、cookie的记录的出现次数	ip和cookie不变的情况下，出现的记录次数指标：N
N1	统计在5小时内，原始数据集中，同一个ip对应的不同cookie的数量	ip不变，对应的不同cookie出现的次数指标：N1
N2	统计在5小时内，原始数据集中，ip前2段相同的记录的出现次数	ip前2段相同的次数指标：N2
N3	统计在5小时内，原始数据集中，ip前3段相同的记录的出现次数	ip前3段相同的次数指标：N3

通过划分时间区间、构建特征、特征标准化得到模型训练数据和测试数据

构建特片源码参考

```
package chap07

import org.apache.spark.sql.{SaveMode, SparkSession}
import org.apache.spark.sql.functions._
import org.apache.spark.sql.types.DataTypes

object s3_Features {
  def main(args: Array[String]): Unit = {
    // 实例化SparkSession对象
    val spark = SparkSession.builder().enableHiveSupport()
      .master("local[4]").appName("Features").getOrCreate()
    // 设置日志输出等级为WARN
    spark.sparkContext.setLogLevel("WARN")

    // 定义特征的名称
```

```
    val timestamps = "timestamps"
    val cookie = "cookie"
    val ip = "ip"
    val N = "N"
    val N1 = "N1"
    val N2 = "N2"
    val N3 = "N3"
    val ranks = "rank"
    // 读取处理后的完整数据
    val data = spark.read.table("ad_traffic.AdData")

    // 取出时间戳最大最小值
    val max_min_timestamp =
data.select(max(col(timestamps).cast(DataTypes.IntegerType)) as "max_ts",
      min(col(timestamps).cast(DataTypes.IntegerType)) as
"min_ts").rdd.collect
    val max_ts = max_min_timestamp(0).getInt(0)
    val min_ts = max_min_timestamp(0).getInt(1)
    // 以18000秒切割时间段
    val times = List.range(min_ts, max_ts, 18000)
    println("时间分割点：" + times)

    for (i<- 0 to 4) {
      // 获取前25小时数据作为模型的构建与评估
      // N、N1、N2、N3特征构建
      val data_sub = data.filter("timestamps>=" + times(i) + " and
timestamps<" + times(i + 1))

      val data_N_sub = data_sub.groupBy(cookie, ip).agg(count(ip) as N)
        .join(data_sub, Seq(cookie, ip), "inner").select(ranks, N)

      val data_N1_sub = data_sub.groupBy(ip).agg(countDistinct(cookie) as
N1).join(
        data_sub, Seq(ip), "inner").select(ranks, N1)

      val data_ip_two = data_sub.withColumn("ip_two",
substring_index(col(ip), ".", 2))

      val data_ip_three = data_sub.withColumn("ip_three",
substring_index(col(ip), ".", 3))

      val data_N2_sub = data_ip_two.groupBy("ip_two").agg(count("ip_two") as
"N2").join(
        data_ip_two, Seq("ip_two"), "inner").select(ranks, N2)

      val data_N3_sub =
data_ip_three.groupBy("ip_three").agg(count("ip_three") as "N3").join(
        data_ip_three, Seq("ip_three"), "inner").select(ranks, N3)

      val data_model_N = data_N_sub.join(data_N1_sub,
ranks).join(data_N2_sub, ranks).join(data_N3_sub, ranks)


 data_model_N.write.mode(SaveMode.Append).saveAsTable("ad_traffic.TimeFeatur
es")
```

```
60
61        // 合并特征与标签
62        val
FeaturesData=spark.read.table("ad_traffic.TimeFeatures").join(data,ranks).se
lect(
63          col(ranks),col("dt"),col("N"),col("N1"),col("N2"),
64          col("N3"),col("label").cast("double"))
65
66
 FeaturesData.write.mode(SaveMode.Overwrite).saveAsTable("ad_traffic.Feature
sData")
67
68      }
69
70    }
71  }
72
```

运行结果：

```
1  Using Spark's default log4j profile: org/apache/spark/log4j-
   defaults.properties
2  23/10/14 10:54:12 INFO SparkContext: Running Spark version 3.2.1
3  23/10/14 10:54:12 WARN NativeCodeLoader: Unable to load native-hadoop
   library for your platform... using builtin-java classes where applicable
4  23/10/14 10:54:12 INFO ResourceUtils:
   ==============================================================
5  23/10/14 10:54:12 INFO ResourceUtils: No custom resources configured for
   spark.driver.
6  23/10/14 10:54:12 INFO ResourceUtils:
   ==============================================================
7  23/10/14 10:54:12 INFO SparkContext: Submitted application: Features
8  23/10/14 10:54:12 INFO ResourceProfile: Default ResourceProfile created,
   executor resources: Map(cores -> name: cores, amount: 1, script: , vendor: ,
   memory -> name: memory, amount: 1024, script: , vendor: , offHeap -> name:
   offHeap, amount: 0, script: , vendor: ), task resources: Map(cpus -> name:
   cpus, amount: 1.0)
9  23/10/14 10:54:12 INFO ResourceProfile: Limiting resource is cpu
10 23/10/14 10:54:12 INFO ResourceProfileManager: Added ResourceProfile id: 0
11 23/10/14 10:54:12 INFO SecurityManager: Changing view acls to: yuxm
12 23/10/14 10:54:12 INFO SecurityManager: Changing modify acls to: yuxm
13 23/10/14 10:54:12 INFO SecurityManager: Changing view acls groups to:
14 23/10/14 10:54:12 INFO SecurityManager: Changing modify acls groups to:
15 23/10/14 10:54:12 INFO SecurityManager: SecurityManager: authentication
   disabled; ui acls disabled; users  with view permissions: Set(yuxm); groups
   with view permissions: Set(); users  with modify permissions: Set(yuxm);
   groups with modify permissions: Set()
16 23/10/14 10:54:13 INFO Utils: Successfully started service 'sparkDriver' on
   port 53269.
17 23/10/14 10:54:13 INFO SparkEnv: Registering MapOutputTracker
18 23/10/14 10:54:13 INFO SparkEnv: Registering BlockManagerMaster
19 23/10/14 10:54:13 INFO BlockManagerMasterEndpoint: Using
   org.apache.spark.storage.DefaultTopologyMapper for getting topology
   information
```

```
23/10/14 10:54:13 INFO BlockManagerMasterEndpoint: BlockManagerMasterEndpoint up
23/10/14 10:54:13 INFO SparkEnv: Registering BlockManagerMasterHeartbeat
23/10/14 10:54:13 INFO DiskBlockManager: Created local directory at C:\Users\yuxm\AppData\Local\Temp\blockmgr-93f72e05-38fe-4406-8bce-c33476a134ca
23/10/14 10:54:13 INFO MemoryStore: MemoryStore started with capacity 1958.7 MiB
23/10/14 10:54:13 INFO SparkEnv: Registering OutputCommitCoordinator
23/10/14 10:54:13 INFO Utils: Successfully started service 'SparkUI' on port 4040.
23/10/14 10:54:13 INFO SparkUI: Bound SparkUI to 0.0.0.0, and started at http://YYMNEO:4040
23/10/14 10:54:13 INFO Executor: Starting executor ID driver on host YYMNEO
23/10/14 10:54:13 INFO Utils: Successfully started service 'org.apache.spark.network.netty.NettyBlockTransferService' on port 53304.
23/10/14 10:54:13 INFO NettyBlockTransferService: Server created on YYMNEO:53304
23/10/14 10:54:13 INFO BlockManager: Using org.apache.spark.storage.RandomBlockReplicationPolicy for block replication policy
23/10/14 10:54:13 INFO BlockManagerMaster: Registering BlockManager BlockManagerId(driver, YYMNEO, 53304, None)
23/10/14 10:54:13 INFO BlockManagerMasterEndpoint: Registering block manager YYMNEO:53304 with 1958.7 MiB RAM, BlockManagerId(driver, YYMNEO, 53304, None)
23/10/14 10:54:13 INFO BlockManagerMaster: Registered BlockManager BlockManagerId(driver, YYMNEO, 53304, None)
23/10/14 10:54:13 INFO BlockManager: Initialized BlockManager: BlockManagerId(driver, YYMNEO, 53304, None)
23/10/14 10:54:15 WARN HiveConf: HiveConf of name hive.stats.jdbc.timeout does not exist
23/10/14 10:54:15 WARN HiveConf: HiveConf of name hive.stats.retries.wait does not exist
23/10/14 10:54:17 WARN ObjectStore: Version information not found in metastore. hive.metastore.schema.verification is not enabled so recording the schema version 2.3.0
23/10/14 10:54:17 WARN ObjectStore: setMetaStoreSchemaVersion called but recording version is disabled: version = 2.3.0, comment = Set by MetaStore UNKNOWN@192.168.31.206
23/10/14 10:54:17 WARN ObjectStore: Failed to get database global_temp, returning NoSuchObjectException
时间分割点：List(0, 18000, 36000, 54000, 72000, 90000, 108000, 126000, 144000, 162000, 180000, 198000, 216000, 234000, 252000, 270000, 288000, 306000, 324000, 342000, 360000, 378000, 396000, 414000, 432000, 450000, 468000, 486000, 504000, 522000, 540000, 558000, 576000, 594000)
23/10/14 10:54:24 WARN SessionState: METASTORE_FILTER_HOOK will be ignored, since hive.security.authorization.manager is set to instance of HiveAuthorizerFactory.
23/10/14 10:54:24 WARN HiveConf: HiveConf of name hive.internal.ss.authz.settings.applied.marker does not exist
23/10/14 10:54:24 WARN HiveConf: HiveConf of name hive.stats.jdbc.timeout does not exist
23/10/14 10:54:24 WARN HiveConf: HiveConf of name hive.stats.retries.wait does not exist
```

```
23/10/14 10:54:30 WARN ProcfsMetricsGetter: Exception when trying to compute pagesize, as a result reporting of ProcessTree metrics is stopped
```

特征标准化源码参考

```scala
package chap07

import org.apache.spark.ml.feature.{MinMaxScaler, VectorAssembler}
import org.apache.spark.sql.SparkSession

object s4_Scaler {
  def main(args: Array[String]): Unit = {
    // 实例化SparkContext
    val spark = SparkSession.builder().master("local[4]")
      .appName("Scaler").enableHiveSupport().getOrCreate()
    spark.sparkContext.setLogLevel("WARN")
    // 读取表数据
    val data = spark.read.table("ad_traffic.FeaturesData")

    val VectorData = new VectorAssembler()

.setInputCols(Array("N","N1","N2","N3")).setOutputCol("VectorFeatures")
      .transform(data)

    // 最小值--最大值归一化
    val MaxMin = new
MinMaxScaler().setInputCol("VectorFeatures").setOutputCol("features").fit(VectorData)
    val dataScaler = MaxMin.transform(VectorData)

    val Array(modelData,testData) = dataScaler.randomSplit(Array(0.7,0.3))

 modelData.write.mode("overwrite").option("header","true").saveAsTable("ad_traffic.ModelData")

 testData.write.mode("overwrite").option("header","true").saveAsTable("ad_traffic.TestData")

    println("模型训练数据量："+modelData.count())
    println("模型后期加载数据量："+testData.count())

  }
}

```

运行结果：

```
模型训练数据量：163590
模型后期加载数据量：70455
```

任务7.4构建与评估分类模型

任务描述

本节的任务如下。

1. 使用逻辑回归算法和随机森林算法构建分类模型；
2. 进行模型预测与评估；
3. 经过对不同模型的效果对比，选择效果较好的模型并应用至实际的模型加载及预测中。

构建与评估逻辑回归模型

通过观察label字段可以看出，广告流量作弊识别为经典的二分类问题，即该广告访问记录是否为作弊访问记录。

逻辑回归模型是解决二分类问题的一个经典模型，而且逻辑回归的原理简单，对于二分类问题的预测准确率也较高。

在模型构建与评估中，编写的Spark程序将不以本地模式运行，而是对Spark程序进行编译打包，使用集群模式将程序上传至集群中运行，因此在IntelliJ IDEA中的SparkSession的实例化和部分参数的设置将会被调整。

构建逻辑回归模型参考

```
package chap07

import org.apache.spark.ml.classification.LogisticRegression
import org.apache.spark.ml.evaluation.MulticlassClassificationEvaluator
import org.apache.spark.sql.SparkSession

object s5_Logistic {
  def main(args: Array[String]): Unit = {
    // master设置为spark任务接收端口，任务名称设置为该任务所对应的名称，此任务为逻辑回归，设置为Logistic
    val spark = SparkSession.builder().enableHiveSupport()
//      .master("spark://192.168.137.100:7077")
      .master("local[4]")
      .appName("Logistic").enableHiveSupport().getOrCreate()
    spark.sparkContext.setLogLevel("WARN")

    // 设置数据位置和模型保存位置为自定义输入参数，在spark-submit提交程序时设置具体的值
    val inputTable = "ad_traffic.ModelData" //args(0)
    val output = "/user/myname/Logistic" //args(1)

    // 读取表数据并进行切分
    val ModelData = spark.read.table(inputTable)
    val Array(train,test) = ModelData.randomSplit(Array(0.7,0.3))
    // 构建并训练逻辑回归模型
    val model = new LogisticRegression()
      .setElasticNetParam(0.03)
      .setMaxIter(15)
      .fit(train)

    val pre = model.transform(test)
    pre.select("label","prediction").show()
    val evaluator = new MulticlassClassificationEvaluator()
```

```
      .setLabelCol("label")
      .setPredictionCol("prediction")
      .setMetricName("accuracy")
    println("Logistic Model Accuracy: " + evaluator.evaluate(pre))

    model.write.overwrite().save(output)
  }
}

```

IDEA控制台运行结果：

```
+-----+----------+
|label|prediction|
+-----+----------+
|  0.0|       0.0|
|  0.0|       0.0|
|  0.0|       0.0|
|  0.0|       0.0|
|  1.0|       0.0|
|  0.0|       0.0|
|  0.0|       0.0|
|  0.0|       0.0|
|  1.0|       0.0|
|  1.0|       0.0|
|  1.0|       1.0|
|  1.0|       1.0|
|  1.0|       0.0|
|  1.0|       0.0|
|  1.0|       0.0|
|  1.0|       1.0|
|  1.0|       1.0|
|  1.0|       1.0|
|  0.0|       0.0|
|  0.0|       0.0|
+-----+----------+
only showing top 20 rows

Logistic Model Accuracy: 0.8583129584352078
```

构建与评估随机森林模型

```
package chap07

import org.apache.spark.ml.classification.RandomForestClassifier
import org.apache.spark.ml.evaluation.MulticlassClassificationEvaluator
import org.apache.spark.sql.SparkSession

object s6_RandomForest {
  def main(args: Array[String]): Unit = {
    // SparkSession实例化
    val spark = SparkSession.builder()
      .appName("RandomForest")
      // .master("spark://192.168.137.100:7077")
```

```
      .master("local[4]")
      .enableHiveSupport()
      .getOrCreate()
    spark.sparkContext.setLogLevel("WARN")
    val inputTabel =  "ad_traffic.ModelData" //args(0)
    val output = "/user/myname/RandomForest" //args(1)

    val ModelData = spark.read.table(inputTabel)
    val Array(traing,test) = ModelData.randomSplit(Array(0.7,0.3))

    // 当使用5棵决策树所构建的随机森林分类模型时准确度较高
    val rf = new
RandomForestClassifier().setFeaturesCol("features").setLabelCol("label").set
NumTrees(5)

    val model = rf.fit(traing)
    val pre = model.transform(test)
    pre.select("label","prediction").show()

    // 模型评估，设置评估标准
    val evaluator = new MulticlassClassificationEvaluator()

.setLabelCol("label").setPredictionCol("prediction").setMetricName("accuracy
")
    println("RandomForest Model Accuracy: " + evaluator.evaluate(pre))
    model.write.overwrite().save(output)

  }
}

```

IDEA控制台运行结果：

```
+-----+----------+
|label|prediction|
+-----+----------+
|  0.0|       0.0|
|  0.0|       0.0|
|  0.0|       0.0|
|  1.0|       1.0|
|  1.0|       1.0|
|  0.0|       0.0|
|  1.0|       1.0|
|  1.0|       1.0|
|  1.0|       1.0|
|  0.0|       0.0|
|  1.0|       1.0|
|  1.0|       0.0|
|  0.0|       0.0|
|  1.0|       1.0|
|  1.0|       1.0|
|  0.0|       0.0|
|  1.0|       1.0|
|  0.0|       0.0|
|  0.0|       0.0|
```

```
|  0.0|       0.0|
+-----+----------+
only showing top 20 rows

RandomForest Model Accuracy：0.9113461499421426
```

对比逻辑回归模型和随机森林模型，随机森林模型高出5.2%的准确率。

模型加载

```scala
package chap07

import org.apache.spark.ml.classification.{LogisticRegressionModel, RandomForestClassificationModel}
import org.apache.spark.ml.evaluation.MulticlassClassificationEvaluator
import org.apache.spark.sql.SparkSession

object s7_LoadModel {
  def main(args: Array[String]): Unit = {
    // 实例化SparkSession
    val spark = SparkSession.builder()
      .master("local[3]")
      .appName("LoadModel")
      .enableHiveSupport().getOrCreate()
    // 设置日志数据等级为Error
    spark.sparkContext.setLogLevel("Error")

    // 读取数据
    val TestData = spark.read.table("ad_traffic.TestData")

    // 加载逻辑回归模型并使用
    val LogisticModel = LogisticRegressionModel.load("/user/myname/Logistic")
    val LogisticPre = LogisticModel.transform(TestData)
    val LogisticAcc = new MulticlassClassificationEvaluator()

.setLabelCol("label").setPredictionCol("prediction").setMetricName("accuracy").evaluate(LogisticPre)
    println("逻辑回归模型后期数据准确率：" + LogisticAcc)

    // 加载随机森林模型并使用
    val RandomForest = RandomForestClassificationModel.load("/user/myname/RandomForest")
    val RandomForestPre = RandomForest.transform(TestData)
    val RandomForestAcc = new MulticlassClassificationEvaluator()

.setLabelCol("label").setPredictionCol("prediction").setMetricName("accuracy").evaluate(RandomForestPre)
    println("随机森林模型后期数据准确率：" + RandomForestAcc)

  }
}

```

IDEA控制台运行结果：

```
逻辑回归模型后期数据准确率：0.8583776878858846
随机森林模型后期数据准确率：0.9097438081044639
```

小结

从案例背景、实现目标、需求分析及系统架构设计展开介绍；接着介绍数据探索分析；然后介绍处理缺失值字段、构建特征、数据标准化；最后介绍模型构建、模型评估、模型加载与应用。