机器人鉴别程序说明文档

0 Preface

Q. 列出你从日志流数据中找到的机器人列表?

A. 见2.2

Q. 解释你如何判断得出每种恶意机器人，他们的显著特征是什么?

A. 逻辑设计见1.3，代码设计见robot.py文件与2.1

Q. kafka流式数据相关?

A. 使用方法见readme，数据库表设计见schema.sql

1 逻辑设计

1.1 背景概述 📺

在原本的kafka流式数据中，存在着一定比例的异常用户，他们是各式各样的自动化机器人，包括但不限于

- Credential Stuffing Attacking Robtot
    - 译作撞库机器人，或者是嗅探机器人，此类机器人通常用于
        - 验证某个账号有没有在一个站点中注册过
        - 通过暴力试探的方式盗取账号密码
- Order Grabing Robot
    - 即抢单机器人/抢票机器人，此类机器人通常用于
        - 以低价的方式获取商家为促销提供的特价商品
        - 获取商家为饥饿营销等商业策略提供的商品或是产量较小的稀有商品
- Scapling Robot
    - 又作刷单机器人，此类机器人通常用于
        - 通过大量无效订单营造虚假销量和虚假好评数
        - 通过大量无效订单进行恶性竞争，诋毁竞争对手
- Crawler
    - 这个即是爬虫机器人，此类机器人通常用途极广，在本业务场景下主要用于
        - 自动化大批量获取商品详细信息

各式各样的机器人在整个数据集中的占比很小，但是即便较少的点也会对一些特征值造成极大的影响，例如平均值，因此有必要在对用户行为进行分析前提前清理掉这类数据

本次数据依据URL格式的不同共分流到5张表上，分别为

- /item/favor 喜欢--->favor表
- /item/getDetail 详细信息--->getDetail表
- /item/cart--->cart表
- /item/buy 购买--->buy表
- /user/login 登录--->login表

1.2 异常特征 ❗

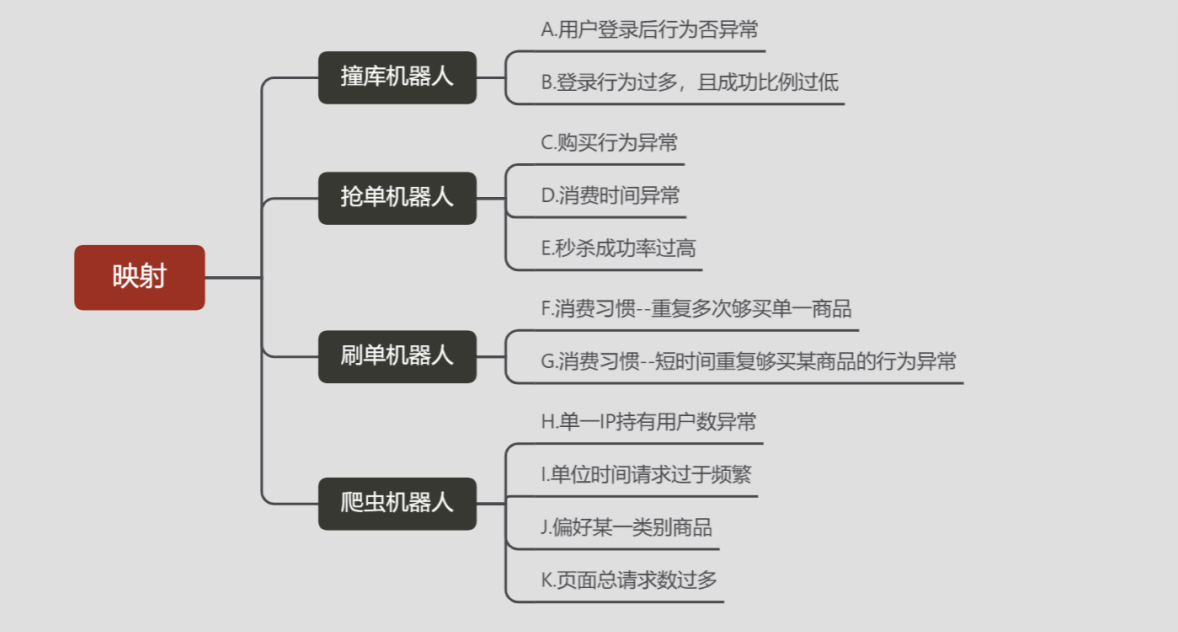
本次总共确定了11种用户特征，如下表

编号	特征描述
A	url 分布比例，隐式表示用户的行为分布比，用于检测用户的行为是否异常
B	登录成功比例，即某IP下登录行为成功（isSuccess=1）的比例
C	购买习惯特征，即加入购物车-购买：直接购买的比例
D	消费行为的时间分布, 即偏好在整点消费（整点+-1min内）占全部购买时间点的比例
E	秒杀成功率，即isSecondKill=1的比例，旨在秒杀成功率过高的ID
F	购买单一商品的最大数量，属于消费习惯范畴
G	短时间重复够买某商品的行为统计，属于消费习惯范畴
H	单一IP下登录用户的数目，旨在识别持有过多ID的异常IP
I	单位时间请求数目统计，旨在识别请求商品信息过于频繁的ID
J	偏好某一类别商品的异常ID
K	总请求数目统计，旨在识别浏览过过多商品信息的ID

1.3 逻辑映射

本部分主要介绍如何把上部分找到的特征应用到各类机器人上

1.3.1 映射



1.3.2 实现逻辑

- 总体逻辑
  - 概率学意义上的异常有很多中解释，一种定义是认为在$mean \pm 2 * std$ （即偏离平均值2倍标准差以上）外的数值都是异常数值，默认情况下，以下均以此作为判断标准
  - meta--operationn
    - 针对每一种特征设计SQL语句，提取出（UserId/IP, 待检验特征值）
    - 计算待检验特征值的标准差和平均值

        - 计算偏离点，部分特征因为特殊的业务逻辑（例如K）只取单边异常点
        - 合并某类机器人的各类异常特征，求交集，取出可疑点
        - 可选的将结果写回数据库
- 特征实现
    - A. 按照UserId 得到每个用户的5类行为的分布比例
    - B. 按照IP 获得每个IP的登录成功比例
    - C. 按照UserId 得到每个用户的（加入购物车再购买÷直接购买）比值
    - D. 按照UserId 得到每个用户的（整点± 1min ÷ 全部购买时间点）比值
    - E. 按照UserId 得到每个用户的（Success=1 ÷ isSecondKill）比值
    - F. 按照UserId 得到每个用户在某个商品上的下单次数
    - G. 按照UserId 得到每个用户在某个商品上的重复购买次数
    - H. 按照IP 获得每个IP下登录成功的用户数目和持有用户数
    - I. 按照UserId 得到每个用户单位时间（min）下请求页面的次数
    - J. 按照UserId 得到每个用户访问特定Category的次数
    - K. 按照UserId 得到每个用户总的请求页面的次数

2 API与实现说明

2.1 数据表操作说明📝

- 数据表的定义在KafkaConsumer文档中已有介绍，这里不再赘述
- 各类特征SQL操作
    - A.废弃 由于5表连接统计用户行为并不现实，本特征分析转由用户行为分析同学在该部分实现
    - B.登录正确率过低，且尝试次数过多(超过10次)
        - ```
          getIP_LoginSuccessRateTooLow()
          ```
        - ```
          SELECT COUNT(*) FROM getdetail;
          select ipAddr,sum(if(success=1,1,0))/count(success) as rate
          from login
          group by ipAddr;
          ```
    - C. 废弃 该特征值几乎为0，在本数据集下不具备实际意义
        - 直接购买的比例过高 统计每一个用户直接购买的数目和有购物车行为的数目
        - ```
          getUser_directPerchase()
          ```
        - ```
          select buy.userId, count(cart.userId)  c1, count(buy.userId)  c2,
          count(cart.userId)/count(buy.userId)  as rate
          from buy left join cart on buy.userId=cart.userId and
          buy.itemId=cart.itemId and buy.date>cart.date
          group by buy.userId;
          ```
    - D. 购买的时间分布
        - ```
          getUser_puchaseTimeDistribute()
          ```
          ```

```
select userId, sum(if(DATE_FORMAT(date,'%i')=59 or
DATE_FORMAT(date,'%i')=60,1,0)) / count(*) as rate
from buy
GROUP BY userId;
```

- E. isSecondKill 准确率评估

```
getUser_isSecondKillSuccess()
```

```
select userId, sum(success)/count(*) as  rate
from buy
where isSecondKill=1
group by userId;
```

- F. 刷单行为-太过于偏好够买某类东西（多次复购该类商品）

```
getUser_purchaseTooMuch()
```

```
select userId,count(*) as totalType,max(num)
from (select userId,itemId,count(*) as num
    from buy
    group by userId,itemId
    ) as x
group by userId;
```

- G. 刷单行为-单位时间（1min） 内buy请求过高

```
getUser_buyTooFrequently()
```

```
select
    userId, date, count(*)
    from
        (select userId, DATE_FORMAT(date, '%Y-%m-%d %H:%i') as date
        from buy
        ) as x
    group by userId, date;
```

- H. 某个IP下用户数目过多

```
getIP_ToomanyUsersLoginAndSuccess()
```

```
select ipAddr,count(*)
from login
where success=1
group by ipAddr;
```

- I. 1min 内用户请求数（简化为getDetail的数目）

```
getUser_getDetailTooFrequently()
```

```
select userId,date,count(*)
from
    (select userId,DATE_FORMAT(date,'%Y-%m-%d %H:%i') as date
    from getDetail
    where id<={} and id>={}
    ) as x
group by userId ,date;
```

- J. 废弃 本特征在逻辑上存在问题，不能以此特征有效说明某个ID是机器人，舍去
- K. getDetail过多

```
getUser_getDetailTooMuch
```

```
select userId,count(*) as num
from getDetail
where id>={} and id<={}
group by userId;
```

2.2 基本功能🤘

- 获得所有潜在的机器人

机器人类型	函数名	返回值
撞库机器人	pwdRobot()	函数返回所有的可疑IP
抢单机器人	orderGrabingRobot()	返回可疑的UserId
刷单机器人	scaplingRobot()	返回可疑的UserId
爬虫机器人	crawlerRobot()	返回可疑的UserId，默认会回打tag

- 如何处理千万级数据量的getDetail表
  - 在表上建立索引
  - 优化了Sql语句，减少了在查询上的中间结果集
  - 通过8轮epoch，将单次涉及的数据量降为 $\frac{1}{8}$,该参数可配置

2.3 附加功能🚀

- 将可疑IP传入ip2userId(l) 函数，会获得该IP下访问过的所有用户
- 将可疑Ip传入tagRobot,会在buy和getDetail表中标注的机器人
- robot.py函数的最上方有, batch，epoch1，epoch为3个超参数，因为getDeta表过大了，不得不分步完成sql语句
  也许你希望限制单次查询的范围,例如batch = 1000000，epoch1=0，epoch=1 的组合即意味着会查询 (0*1000000,1*1000000)区间内的数据. 计算公式为为（epoch1*batch，epoch*batch）