尚硅谷电商用户画像 之一

张晨

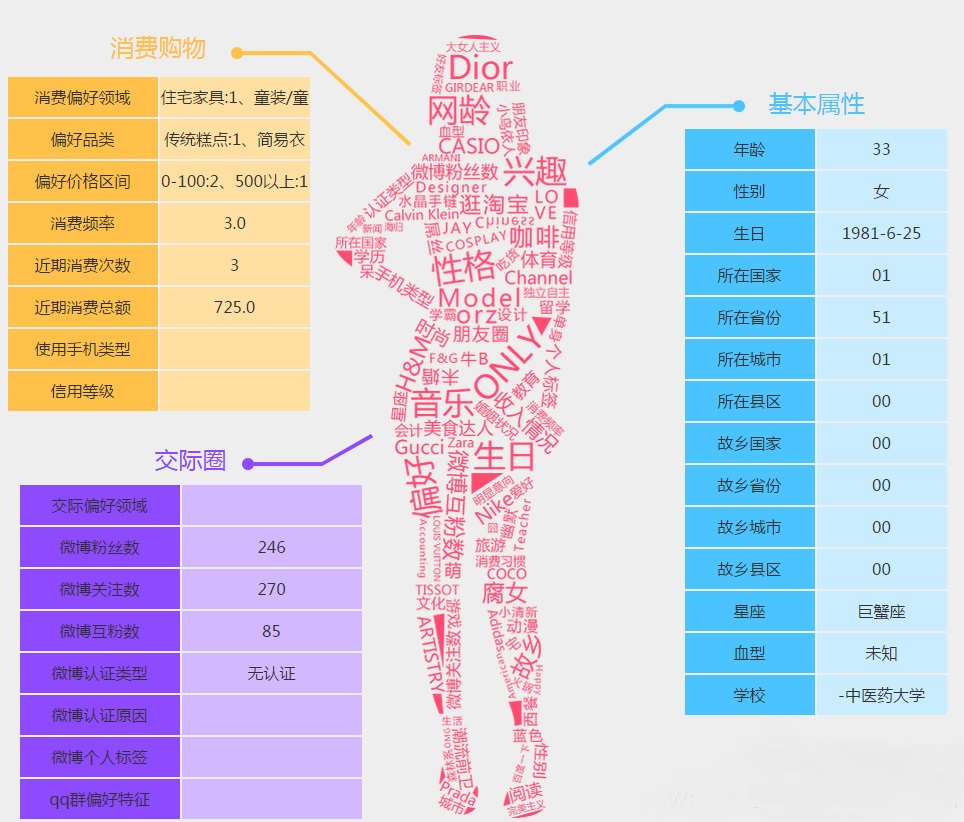
# 用户画像简介

## 什么是用户画像

用户画像，英文: **User Profile**

( 也有少数称： User Portrait 或User Persona)

**一句话概念：将用户信息标签化**



通过收集用户的社会属性、消费习惯、偏好特征等各个维度的数据，进而对用户或者产品特征属性进行刻画，并对这些特征进行分析、统计，挖掘潜在价值信息，从而抽象出用户的信息全貌**。**

## 定位与应用

### 1.2.1 定位：

相对于数据仓库而言，用户画像属于“上层建筑”，以数据仓库沉淀的数据为基础，提炼出更有价值的信息。

同时用户画像也是一种数据服务，在它之上还有“更高的建筑”，比如推荐系统，营销系统、风控系统、用于广告投放的DMP系统等等。这些系统往往需要对用户进行识别定位，那么用户画像就是最重要的数据来源。

### 1.2.2 应用：

画像数据的主要应用类型：

* + 运营决策

了解用户群体，聚焦目标用户，定位产品方向。

* + 精准营销

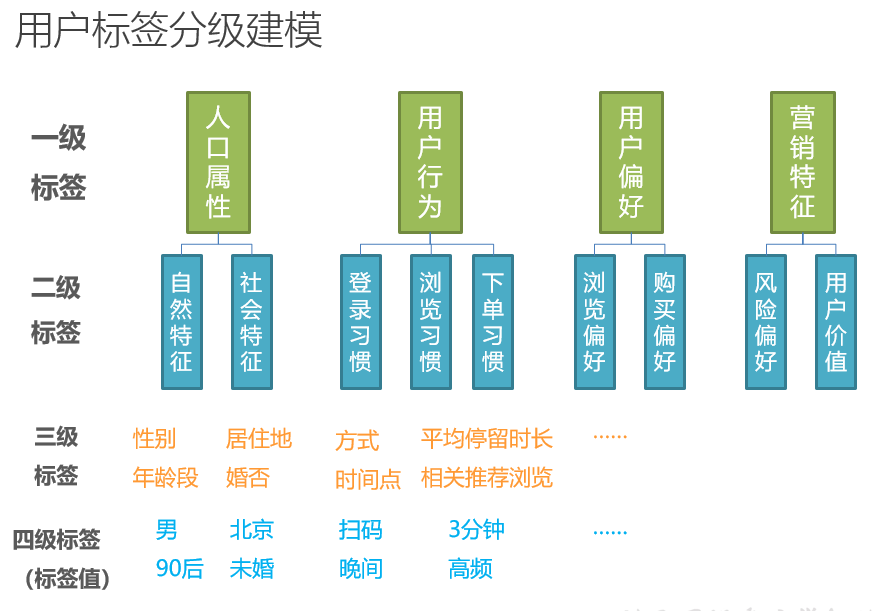
营销活动推送、广告投放、个性化推荐。

* + 用户分群

寻找高价值用户，挽留待流失用户，提升用户活跃。

## 1.3 用户标签

### 1.3.1 标签分级



### 1.3.2 标签分类

* **统计类标签**

直接提取的标签，又叫事实标签。

比如：性别，年龄，最近一次登录时间，月均消费。

有非常通用且明确的定义，是最为常见的标签。

* **规则类标签**

需要自定义规则。

比如：高价值用户、意见领袖、电子产品爱好者、黄牛党。

需要运营、产品、业务人员，根据企业自身的业务特征，设计适合自身的规则定义。往往同一个名称的标签，在不同企业的规则不同。

* **挖掘类标签**

一般通过机器学习算法进行预测的标签。又叫预测类标签。

比如：预测性别、预测年龄、潜在流失用户。

通常是很难根据某一个规则得到的标签。需要机器学习通过系统现有的数据，反复迭代获得一个模型算法，再根据算法得到标签。

开发周期长，难度大，准确度不能保证。但是往往也是最有价值的标签，因为从数据得到的数据，有时往往比定死的规则更反映真实情况。

# 第二章 用户画像的架构



数仓阶段：sql

画像流程处理：shell+azkaban

用户画像管理平台：分群管理会生成人群包、人群圈选(人圈)，一般白天即时处理。

用户标签明细及分析：比较大clickhouse，中等规模ES（没有位图），存储标签表、人群表

## 2.1 用户画像管理平台：

提供可视化页面，对标签及标签产生的规则进行定义，甚至直接提供可视化开发页面。

提供后台调度系统，根据标签定义的规则，从数仓中抽取计算。

计算后的用户画像标签也由平台管理，通过标签的组合，把用户分成不同的群体。为其他业务系统提供支持。

**技术实现：**

**用户画像系统**本质上是一个内部的管理系统，方便用户画像开发团队，搭建标签管理任务的。基于标准的Web应用的技术。

**Vue.js**负责前端页面。

**Springboot** 负责后台应用，数据保存在Mysql数据库中，相关的技术框架还包括MybatisPlus、StringTask。

因为还需要把spark程序任务提交到**Yarn**，所以还用到SparkLauncher插件。

## 2.2  画像处理流程

**画像处理流程**主要是根据标签及整个流程的规则**计算标签**，把数据仓库中的数据进行重组。

一般统计类和规则类标签使用**spark-sql**即可，复杂的规则类标签和挖掘类标签可以使用**spark-core**和**spark-mllib**完成。

## 2.3 画像标签数据应用：用户分群

提供了分群操作所以要操作支持即席查询的OLAP，对标签及人群进行操作。

根据实际需要一般选择性能较好，支持即席查询的OLAP数据库。用于组合和多个条件来筛选用户，比如**Clickhouse**或者**Elasticsearch** .同时也会使用K-V数据库用于精确查询用户和人群,比如**Redis**、**Hbase** 、**Pika**。

## 2.4 系统演示

## 2.5 项目教学模块

1. 理解用户画像系统的设计思路，以及标签的设计流程与应用。
2. 用户画像管理平台的搭建及使用。
3. 掌握用户画像平台的数据库表，包括标签表、任务表、进度表等全部表含义。
4. 使用spark开发标签计算、重组、导出等操作，完成标签计算的业务处理流程。
5. 利用clickhouse实现对画像数据的存储及多个标签的组合筛选。
6. 学习用Springboot、Mybatis等框架，完成用户分群功能。
7. 入门机器学习，用sparkmllib中的算法完成挖掘类标签的开发。

# 第三章 搭建用户画像管理平台

## 3.1 系统物料：

1. 平台主程序代码包 ( 因为还需要调试开发， 直接在idea运行本地运行 )

Gitee仓库地址：https://gitee.com/windyzj/user\_profile\_manager\_班号.git

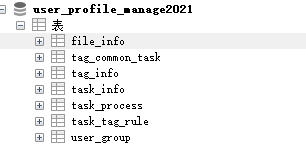
1. spark远程提交服务Jar包，放在有spark环境的服务器。
2. 管理平台，Mysql建表脚本

本地程序如何打包上传：

1. 搭建本地库：点击IDEA上方VSC-Import into version control-create git repository
2. 在git创建代码库，复制链接
3. 在IDEA git-上传

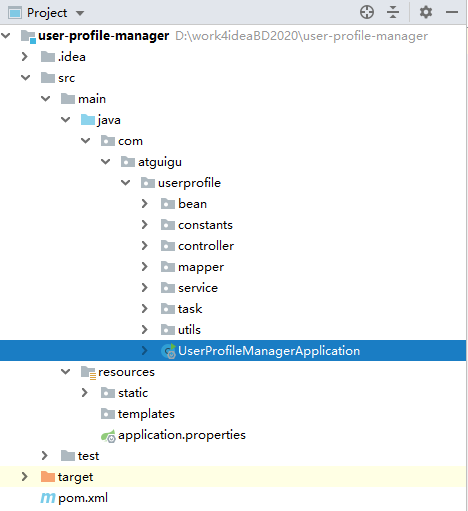
## 3.2 搭建用户画像管理平台的数据库

在MySQL中导入数据脚本



## 3.3 平台主程序项目

### 3.3.1 把主程序导入idea工程



### 3.3.2 修改相关的配置

application.properties

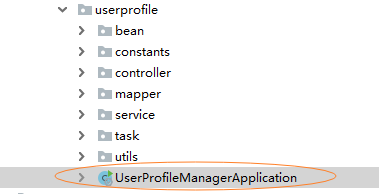
|  |
| --- |
| ***#后台服务端口号* server.port**=**80**  ***#用于连接clickhouse 保存标签计算结果 和 人群包 如果这里没装clickhouse 这里可以先注掉 #spring.datasource.dynamic.datasource.clickhouse.url=jdbc:clickhouse://bigdata01:8123/user\_profile2021 #spring.datasource.dynamic.datasource.clickhouse.driver-class-name=ru.yandex.clickhouse.ClickHouseDriver  #画像平台上传到hdfs的文件地址和路径* hdfs.url**=**hdfs://bigdata01:8020 hdfs.username**=**bigdata hdfs.filedir**=**/user\_profile\_manage  *#上传文件大小的限制，如果有超大jar包可以调整* spring.servlet.multipart.max-file-size**=**100MB spring.servlet.multipart.max-request-size**=**100MB  *#mybaits-plus空值处理* mybatis-plus.global-config.db-config.field-strategy**=**not\_null  *#提交远程服务的 之后部署远程spark提交器的地址* spark.rest.submitter.url**=**http://bigdata01:8266/spark-submit  *#日志级别* logging.level.root**=**error** |

### 3.3.3 修改相关的配置

|  |
| --- |
| 127.0.0.1 userprofile.gmall.com |

### 3.3.4 启动画像平台应用

执行工程中的UserProfileManagerApplication



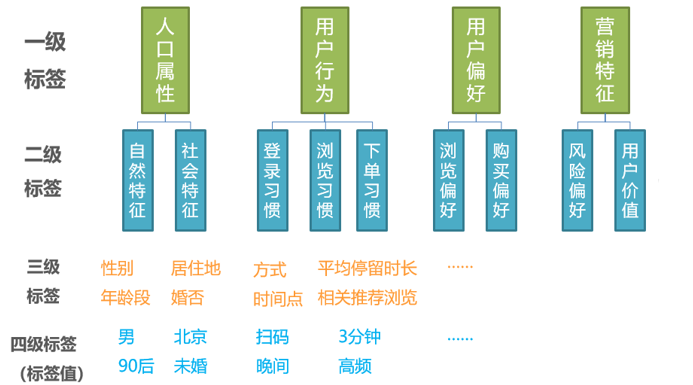
### 3.3.5 访问画像平台



# 第四章 标签管理

标签管理模块，是用户画像是开发的起点，所有的用户画像都起始于对标签的定义和规则的指定。

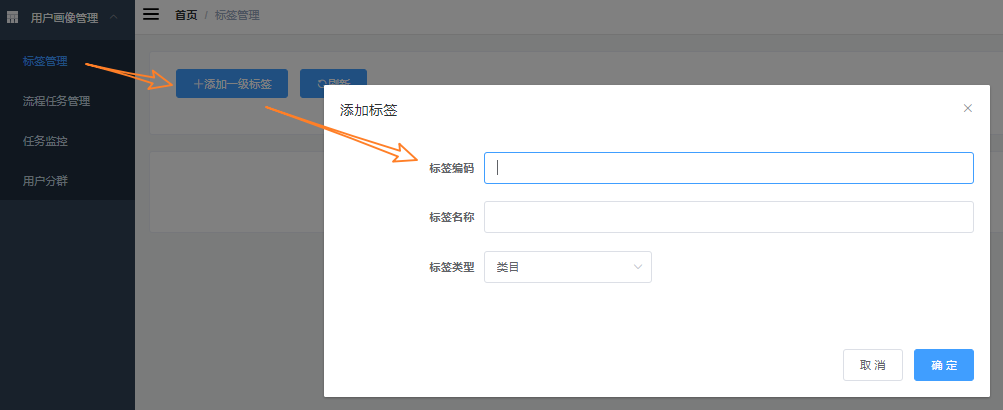
## 4.1 标签列表



根据之前对标签的规划，我们要在这里把标签创建为四个级别，其中**一二级**为标签的类目，第三级为主要的标签，第四级是具体的标签数值。

## 4.2 添加标签

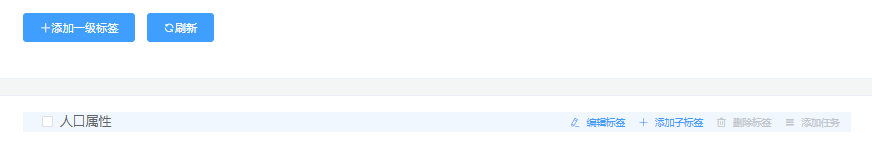
在页面的左上角点击，添加根节点



|  |  |
| --- | --- |
| 页面字段 | 说明 |
| 标签编码 | 根据规则填写编码(大写英文) |
| 标签名称 | 标签中文名称 |
| 标签类型 | 一级、二级选择类目，三四级选择:数值、浮点、文本、日期 |



填写内容，点击确认保存标签。



### 4.2.1 填写二级标签

在人口属性右侧点击【添加子标签】



注意编码要体现层级



### 4.2.2 填写三级标签

同样在【自然属性】右侧点击【添加子标签】



### 4.2.3 填写四级标签



### 4.2.4 数据库标签表字段说明

|  |  |  |
| --- | --- | --- |
| 数据库字段 | 字段说明 | 值说明 |
| id | 标签主键 |  |
| tag\_code | 标签编码 |  |
| tag\_name | 标签名称 |  |
| tag\_level | 标签层级 | 1、2、3、4 |
| parent\_tag\_id | 上级标签ID |  |
| tag\_type | 标签类型 | 1统计型、2规则型、3挖掘型 |
| tag\_value\_type | 标签值类型 | 1整数、2浮点、3文本、4日期 |
| tag\_task\_id | 标签任务编号 |  |
| tag\_comment | 标签说明 |  |
| update\_time | 更新时间 |  |
| create\_time | 创建时间 |  |

## 4.3 标签任务

### 4.3.1 添加标签

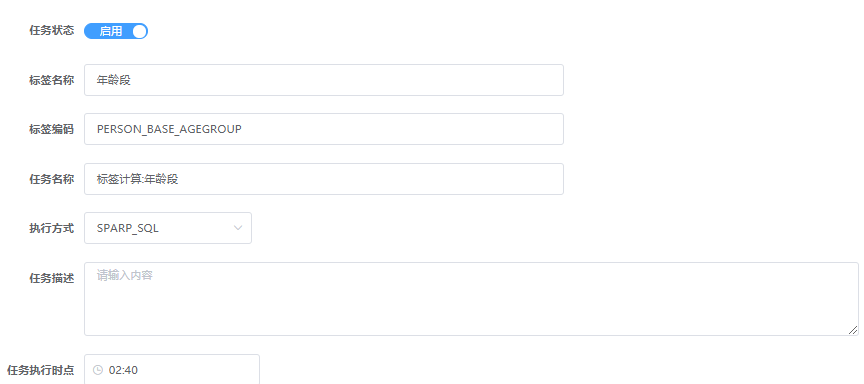


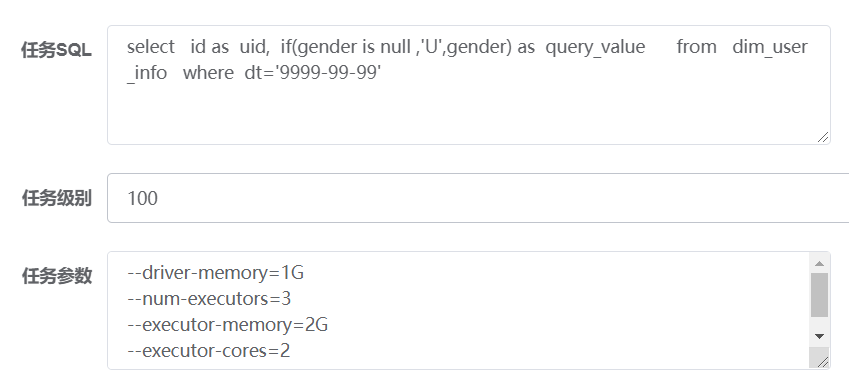


### 4.3.2 字段填写说明

|  |  |
| --- | --- |
| 页面字段 | 说明 |
| 任务状态 | 只有开启状态的任务平台才会进行定时调度 |
| 任务名称 | 任务的名称 |
| 执行方式 | SPARK\_SQL、SPARK\_JAR  复杂的规则类标签或者挖掘类标签使用SPARK\_JAR,一般统计和简单的规则使用SPARK\_SQL |
| 标签类型 | 一级、二级选择类目，三四级选择:数值、浮点、文本、日期 |
| 任务描述 |  |
| 运行时间 | 每天运行该标签任务的时间，一定晚于数仓0.5-1h |
| 任务SQL | 当选择SPARK\_SQL的执行方式时，  系统自动根据该栏位填写的SQL进行计算  sql的查询结果，必须是两列一列是uid ,一列是query\_value |
| 任务级别 | 根据级别决定任务的执行次序，级别数字越小越早执行。默认为100. |
| 任务参数 | 提交spark程序时提交的参数，master、deploy-mode、class 、执行的jar包不用写。  如：  --driver-memory=1G  --num-executors=3  --executor-memory=2G  --executor-cores=2  --conf spark.default.parallelism=12  注意参数与值之前要放等号，不要用空格 |
| 标签规则配置 | 根据sql语句产生的query\_value与四级标签（标签值）做映射。 |

### 4.3.3 填写示例

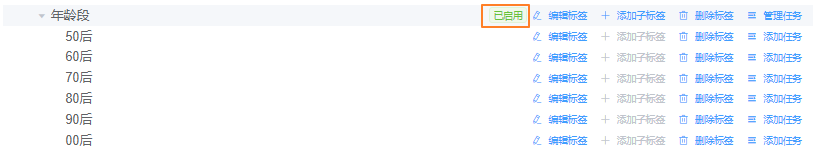






保存后

标签列表会显示已启用



到此为止，一个标签的简单定义完成了。