# 广电大数据用户画像数据处理模型（设计）

# 用户信息分析精炼的特征标识的研究

作　　者：姚　　圆
学　　院：信息工程学院
专　　业：数据科学与大数据技术
年　　级：2020
学　　号：2020145128
任课教师：李　京　建
课程名称：Spark 编程基础
完成日期：2023.06.30

# 目录

# 引言

用户画像是指根据用户属性、用户偏好、生活习惯、用户行为等信息抽像出来的标签化用户模型。就是给用户打标签，而标签通过对用户信息分析而来的高度精炼的特征标识。通过打标签可以利用一些高度概括、容易理解的特征来描述用户，可以让人更容易理解用户，并且可以方便计算机处理。

在互联网、电商领域用户画像常用来作为精准营销、推荐系统的基础性工作，其作用总体包括：

（1）精准营销：根据历史用户特征，分析产品的潜在用户和用户的潜在需求，针对特定群体，利用短信、邮件等方式进行营销。

（2）用户统计：根据用户的属性、行为特征对用户进行分类后，统计不同特征下的用户数量、分布；分析不同用户画像群体的分布特征。

（3）数据挖掘：以用户画像为基础构建推荐系统、搜索引擎、广告投放系统，提升服务精准度。

（4）服务产品：对产品进行用户画像，对产品进行受众分析，更透彻地理解用户使用产品的心理动机和行为习惯，完善产品运营，提升服务质量。

（5）行业报告&用户研究：通过用户画像分析可以了解行业动态，比如人群消费习惯、消费偏好分析、不同地域品类消费差异分析

# 摘要

随着互联网技术的快速发展和应用扩展，国家正式推进三网融合政策，三网融合是指电信网、广播电视网、互联网在向宽带通信网、数字电视网、下一代互联网演进过程中，三大网络通过技术改造，其技术功能趋于一致，业务范围趋于相同，网络互联互通、资源共享，能为用户提供语音、数据和广播电视等多种服务。 新媒体业务的飞速发展对传统媒体造成了巨大冲击，广电行业依靠资源稀缺形成的垄断优势已经失去。

项目目标:

1. 挖掘分析用户相关数据，对用户数据进行标签化，建立一个用户画像模型，可提供标签的增加和删除。

2. 利用 SVM 算法建立分类模型，预测用户是否值得挽留，并将预测结果作为用户画像的一个标签。

3. 提出的系统架构能适应用户数据大量增长时不需要调整系统架构的情况，即支持动态横向扩展。

关键词：标签化　SVM 算法　用户画像　分类

# Abstract

With the rapid development and application expansion of Internet technology, the state officially promotes the policy of triple play convergence. Triple play convergence means that in the evolution process of telecommunication network, broadcast television network and Internet to broadband communication network, digital television network and next generation Internet, the three networks through technical transformation, their technical functions tend to be consistent and their business scope tends to be the same. Network interconnection, resource sharing, can provide users with voice, data, radio and television and other services. The rapid development of new media business has caused a huge impact on traditional media, and the radio and television industry has lost the monopoly advantage formed by resource scarcity. Project objectives: 1. Mining and analyzing user related data, labeling user data, establishing a user portrait model, which can provide the addition and deletion of labels. 2. SVM algorithm is used to build a classification model to predict whether users are worth retaining, and the prediction result is used as a label of user portrait. 3. The proposed system architecture can adapt to the large growth of user data without the need to adjust the system architecture, that is, support dynamic horizontal expansion.

Key words: Label SVM algorithm user profile classification

# 第一章、 业务需求

## 1.1 需求分析

随着我国有线数字电视网络由单向广播电视网向双向化的下一代广播电视网的演进发展，传统广电运营商正由原来的单一网络运营商向综合信息服务运营商转变，在这个转变过程中，及时获取用户的兴趣偏好、收视习惯和消费特征成为了一个关键因素，同时，随着大数据技术的普及，广电运营商逐步开始利用大数据平台进行海量数据的存储和分析，通过分析海量数据的特性，挖掘数据价值，构建基于广电运营大数据的用户画像系统，将极大的方便广电运营商掌握用户行为特征和节目资源特性，从而更加及时准确的服务用户，进而极大的改善用户体验、引导用户消费、提升用户黏性，为广电运营商的个性化服务和智能运营决策提供辅助支持。

本文分析了广电运营场景下的用户画像应用，针对广电业务的特点和需求，设计并实现了一套用户画像构建系统，完成了用户画像和知识库的构建，同时提供交互式分析和可视化功能，方便分析人员进行运营分析，最后进行了总结和展望。

# 第二章、 实现理论依据

## 2.1 选用平台

Spark+Hadoop 大数据框架：

1、Spark 的计算模式也属于 MapReduce，但不局限于 Map 和 Reduce 操作，还提供多种数据集操作类型，编程模型比 Hadoop MapReduce 更灵活。

2.Spark 提供了内存计算，可将中间结果放到内存中，对于迭代运算效率更高 3.Spark 基于 DAG 的任务调度执行机制，要优于 Hadoop MapReduce 的迭代执行机制。

```
# Spark 概述
SparkCore：最低层的组件
SparkSQL：查询计算
SparkStreaming：进行流计算
MLlib：机器学习算法库
```

```
GraphX：编写图计算算法
Hadoop 表达能力有限，磁盘开销大
运行速度快，内存计算
有向无环图的计算
容易使用 Java，Scala，python，R 语言
完整的软件栈，独立的集群模式
```

## 2.2 数据说明

### 2.2.1 用户信息数据：mediamatch_usermsg

```
字段 描述

terminal_no 客户地址编号

phone_no 客户编号

sm_name 品牌名称

run_name 状态名称

sm_code 品牌编号

owner_name 客户等级名称

owner_code 客户等级

run_time 状态变更时间

addressoj 完整地址

estate_name 街道或小区地址

force 宽带是否生效

open_time 开户时间
```

### 2.2.2 用户状态变更信息数据：mediamatch_userevent

```
字段 描述

run_name 状态名称

run_time 更改状态时间
```

```
6.
7.    owner_code 客户等级编号
8.
9.    owner_name 客户等级名称
10.
11.   sm_name 品牌名称
12.
13.   open_time 开户时间
14.
15.   phone_no 客户编号
```

### 2.2.3 账单信息数据：mmconsume_billevents

```
1.    字段 描述
2.
3.    fee_code 费用类型
4.
5.    phone_no 客户编号
6.
7.    owner_code 客户等级
8.
9.    owner_name 客户等级编号
10.
11.   sm_name 品牌名
12.
13.   year_month 账单时间
14.
15.   terminal_no 用户地址编号
16.
17.   favour_fee 优惠金额(+代表优惠，-代表额外费用)
18.
19.   should_pay 应收金额，单位:元
```

### 2.2.4 用户收视行为数据：media_index

```
1.    字段名 描述 terminal no 用户地址编号
2.
3.    phone no 用户编号
4.
5.    duration 观看时长，单位:毫秒
6.
7.    station name 直播频道名称
8.
9.    origin_time 观看行为开始时间
10.
```

```
end_time 观看行为结束时间

owner_code 客户等级

owner_name 客户等级名称

vod_cat_tags vod 节目包相关信息(nested object)按不同的节目包目录组织

resolution 点播节目的清晰度

audio Jlang 点播节目的语言类别

region 节目地区信息

res_name 设备名称

res_type 媒体节目类型 0 是直播，1 是点播或回看

vod_titlevod 节目名称

category_name 节目所属分类

program_title 直播节目名称

sm_name 用户品牌名称
```

### 2.2.5 订单信息数据 media_index

```
字段 描述

phone_no 用户编号

owner_name 客户等级名称

optdate 产品订购状态更新时间

Prodname 订购产品名称

sm_name 用户品牌名称

offerid 订购套餐编号

offername 订购套餐名称

business_name 订购业务状态
```

```
18.
19.   owner_code 客户等级
20.
21.   prodprcid 订购产品名称(带价格)的编号
22.
23.   prodprcname 订购产品名称(带价格)
24.
25.   effdate 产品生效时间
26.
27.   expdate 产品失效时间
28.
29.   orderdate 产品订购时间
30.
31.   cost 订购产品价格
32.
33.   mode_time 产品标识，辅助标识电视主、附销售品
34.
35.   prodstatus 订购产品状态
36.
37.   run_name 状态名
38.
39.   orderno 订单编号
```

# 第三章 数据存储-数据说明

## 3.1 数据存储准备

### 3.1.1 数据存储准备

使用 xftp 将数据上传到 opt 目录下 依次将上传的五个文件删除第一行数据，因为第一行数据为表结构

```
1.    sed -i 1d mediamatch_usermsg.csv
2.    sed -i 1d media_index.csv
3.    sed -i 1d mediamatch_userevent.csv
4.    sed -i 1d mmconsume_billevents.csv
5.    sed -i 1d order_index.csv
```

然后我们打开虚拟机启动 Hadoop 服务、Zookeeper 服务

### 3.1.2 启动 hive:

```
1.    hive --service metastore &
```

### 3.1.3 进入 hive:

```
hive
```

### 3.1.4 创建一个数据库 zjsm：

```
create database zjsm;
```

### 3.1.5 我们需要创建五个表:

```
mediamatch_usermsg   用户信息表
mediamatch_userevent  状态变更表
mmconsume_billevents  账单表
order_index   订单表
media_index    收视表
```

## 3.2 执行数据存储脚本

### 3.2.1 在 Hive 中执行以下命令创建表并且导入数据到 Hive 中

账单表

创建 mmconsume_billevents 表

```
create table mmconsume_billevents(
terminal_no string,
phone_no string,
fee_code string,
year_month string,
owner_name string,
owner_code string,
sm_name string,
should_pay double,
favour_fee double)
row format delimited fields terminated by '\073';
```

### 3.2.2 将数据导入到 mmconsume_billevents 表

```
load data local inpath '/opt/mediamatch_usermsg.csv' overwrite into table mediamatch_usermsg;
```

### 3.2.3 订单表

创建 order_index 表

```
create table order_index(
phone_no string,
```

```
    owner_name string,
    optdate string,
    prodname string,
    sm_name string,
    offerid int,
    offername string,
    business_name string,
    owner_code string,
    prodprcid int,
    prodprcname string,
    effdate string,
    expdate string,
    orderdate string,
    cost string,
    mode_time string,
    prodstatus string,
    run_name string,
    orderno string,
    offertype string)
    row format delimited fields terminated by '\073';
```

### 3.2.4 将数据导入到 order_index 表中

```
load data local inpath '/opt/order_index.csv' overwrite into table order_index;
```

### 3.2.5 状态变更表

#### 3.2.5.1 创建 mediamatch_userevent 表

```
create table mediamatch_userevent(
phone_no string,
run_name string,
run_time string,
owner_name string,
owner_code string,
open_time string,
sm_name string)
row format delimited fields terminated by '\073';
```

### 3.2.6 将数据导入到 mediamatch_userevent 表中

```
load data local inpath '/opt/mediamatch_userevent.csv' overwrite into table mediamatch_userevent;
```

数据存储工作完成

# 第四章 数据预处理

## 4.1 实现思路

使用 Hive 支持来访问 Hive 中的表数据。通过循环遍历表名，并调用 Analyse() 函数对每个表进行分析，统计记录数和空值记录数，并打印输出。

代码实现：

```
1.  package code.anaylse
2.
3.  // 导入所需的库和模块：导入 org.apache.spark.sql.SparkSession 库，这是使用 Spark 进行数据分析的核心库。
4.  import org.apache.spark.sql.SparkSession
5.  object BasicAnaylse {
6.    val spark=SparkSession.builder().appName("BasicAnalyse")
7.      .master("local[*]")
8.      .enableHiveSupport()
9.      .getOrCreate()
10. // 创建 SparkSession：使用 SparkSession.builder()创建一个 SparkSession 实例。在 builder()方法中，设置应用程序的名称为"BasicAnalyse"，设置 Spark 的 master 节点为本地模式（.master("local[*]")），使用所有可用的本地线程进行运算。并启用 Hive 支持（.enableHiveSupport()），以便通过 Hive 访问数据。最后，通过.getOrCreate()获取或创建 SparkSession 实例。
11.   //
12. // 设置日志级别：使用 spark.sparkContext.setLogLevel("WARN")将 Spark 的日志级别设置为"WARN"，以减少日志输出。
13.   spark.sparkContext.setLogLevel("WARN")
14. // 创建 main()函数：主要的代码逻辑将放置在 main()函数中。
15.   def main(args: Array[String]): Unit = {
16.     //探索每个表中的重复记录表和空值记录数
17.     val tableName = Array("media_index","mediamatch_userevent","mediamatch_usermsg","mmconsume_billevents","order_index")
18.     var i = "";
19. // 使用 for 循环遍历 tableName 数组中的每个表名。
20.     for(i<-tableName){
21. // 在循环内部，调用 Analyse()函数，并将当前表名作为参数传递给它。
22.       Analyse(i)
23.     }
24.
25. // 可以选择性地注释掉一行代码，使用 spark.table()和 show()方法显示指定表的内容。
26. //    val mediamatch_userevent = spark.table("user_project.mediamatch_userevent")
```

```
//    mediamatch_userevent.show(false)
  }
//创建 Analyse()函数：该函数用于对指定表进行分析
  def Analyse(tableName:String): Unit ={
// 用 spark.table("user_project."+tableName)获取指定表的数据，前缀"user_project."用于指定数据库名称。
    val data = spark.table("user_project."+tableName)
// 使用 count()方法获取表中的记录数，并通过 print()函数打印出来。
    print(tableName+"表数据：  "+data.count())
//使用 data.select("phone_no").na.drop().count 选择"phone_no"字段，并使用 na.drop()方法去除该字段中的空值，再使用 count()方法获取非空值记录的数量。计算空值记录数，并通过 print()函数打印出来。
    print(tableName+"表 phone_no 字段为空数：  "+(data.count()-data.select("phone_no").na.drop().count))
  }
}
```