基于 Hive 的淘宝用户行为数据分析

1. 项目背景及目的

以淘宝、京东为首的电商行业经过十几年的快速发展，吸引并获得了大量的用户。移动互联网时代，用户的需求决定了电商行业未来的发展方向，在这样的背景下，获取用户行为并进行分析，理解用户需求，从而指导产品营销和设计，对于电商的未来发展具有重要意义。本项目使用大数据分析工具-数据仓库 Hive 对淘宝用户行为数据进行分析，通过数据清洗、建模分析及可视化等过程对业务问题进行深层分析，通过用户的行为习惯及偏好为商家决策提供相应建议。

2. 数据集介绍

本项目采用阿里云天池提供的淘宝用户行为数据集。该数据集共有 100,150,807 行记录与 5 个属性，数据格式如下表所示：

数据字段	描述
userid	用户id
itemid	商品id
categoryid	商品类目ID
type	行为类型
timestamp	时间戳

数据集介绍及下载地址：淘宝用户购物行为数据集_数据集-阿里云天池 (aliyun.com)，或百度网盘：https://pan.baidu.com/share/init?surl=5Ss-nDMA120EHhuwpzYm0g，（提取码：5ipq）

3. 开发工具

Hive：用于数据清洗与分析
mySQL：数据存储
可视化：echarts
Sqoop：数据迁移
具体使用方法参考：https://blog.csdn.net/m0_61466268/article/details/124718809

4. 项目开发内容（使用 Hive SQL 编程实现）

（1）实验环境准备

- 安装 Linux 操作系统
- 安装关系型数据库 MySQL
- 安装大数据处理框架 Hadoop
- 安装数据仓库 Hive
- 安装 Sqoop
- 安装 Eclipse

安装指南：https://dblab.xmu.edu.cn/blog/1362/

（2）数据导入

将数据加载到 hive，示例代码：

```
-- 建表
drop table if exists user_behavior;
create table user_behavior (
`user_id` string comment '用户ID',
`item_id` string comment '商品ID',
`category_id` string comment '商品类目ID',
`behavior_type` string  comment '行为类型，枚举类型，包括(pv, buy, cart, fav)',
`timestamp` int comment '行为时间戳',
`datetime` string comment '行为时间')
row format delimited
fields terminated by ','
lines terminated by '\n';

-- 加载数据
LOAD DATA LOCAL INPATH '/home/getway/UserBehavior.csv'
OVERWRITE INTO TABLE user_behavior ;
```

（3）数据清洗

通过 hive 对数据进行清洗，包括删除重复着，时间戳格式化，及删除异常值。参考代码：

```
--数据清洗，去掉完全重复的数据
insert overwrite table user_behavior
select user_id, item_id, category_id, behavior_type, timestamp, datetime
from user_behavior
group by user_id, item_id, category_id, behavior_type, timestamp, datetime;

--数据清洗，时间戳格式化成 datetime
insert overwrite table user_behavior
select user_id, item_id, category_id, behavior_type, timestamp, from_unixtime(timestamp, 'yyyy-MM-dd HH:mm:ss')
from user_behavior;

--查看时间是否有异常值
select date(datetime) as day from user_behavior group by date(datetime) order by day;

--数据清洗，去掉时间异常的数据
insert overwrite table user_behavior
select user_id, item_id, category_id, behavior_type, timestamp, datetime
from user_behavior
where cast(datetime as date) between '2017-11-25' and '2017-12-03';

--查看 behavior_type 是否有异常值
select behavior_type from user_behavior group by behavior_type;
```

（4）数据分析及可视化

（Echarts 可视化操作可参考：https://dblab.xmu.edu.cn/blog/1369/）

1）用户流量及购物情况

```sql
--总访问量PV，总用户量UV
select sum(case when behavior_type = 'pv' then 1 else 0 end) as pv,
       count(distinct user_id) as uv
from user_behavior;
```

Result | Statistics

select sum(case when behavior_type = 'pv' then 1 输入一个 SQL 表达式来过滤结果 (使用

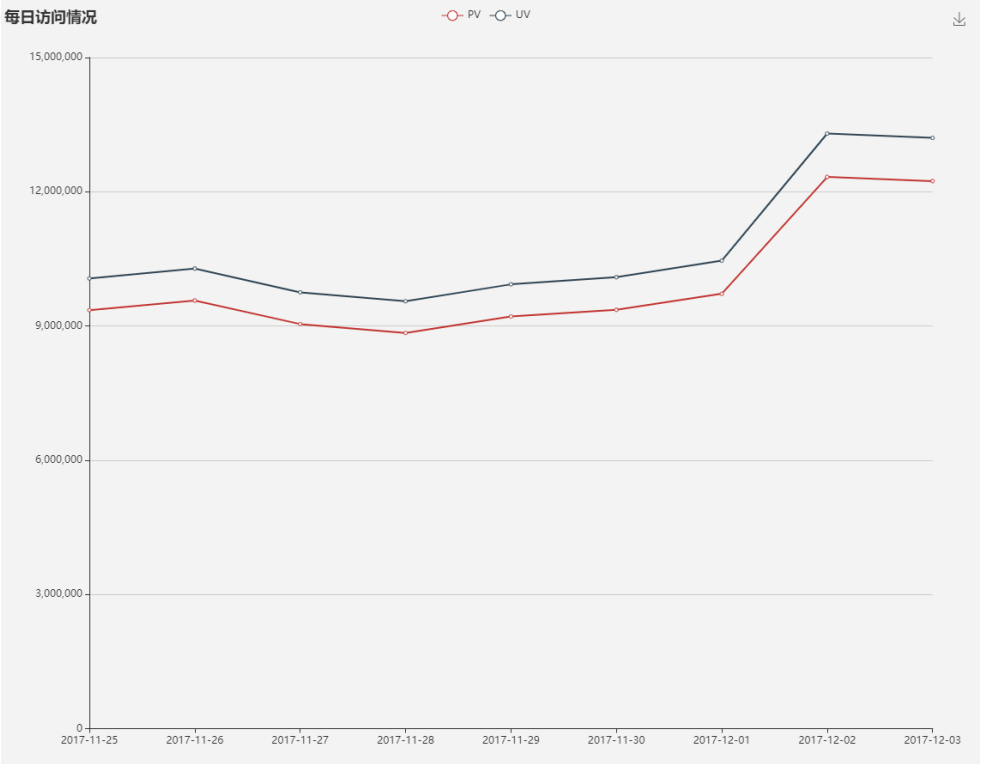
	pv	uv
1	89,660,671	987,991

```sql
--日均访问量，日均用户量
select date(datetime) as day,
       sum(case when behavior_type = 'pv' then 1 else 0 end) as pv,
       count(distinct user_id) as uv
from user_behavior
group by date(datetime)
order by day;
```

Result | Statistics

select date(datetime) as day, sum(case when beha 输入一个 SQL 表达式来过滤结果 (使用 Ctrl+S

	day	pv	uv
1	2017-11-25	9,353,416	706,641
2	2017-11-26	9,567,422	715,516
3	2017-11-27	9,041,186	710,094
4	2017-11-28	8,842,932	709,257
5	2017-11-29	9,210,820	718,922
6	2017-11-30	9,358,998	730,597
7	2017-12-01	9,718,956	740,139
8	2017-12-02	12,329,641	970,401
9	2017-12-03	12,237,300	966,977




```
--每个用户的购物情况，加工到 user_behavior_count
create table user_behavior_count as
select user_id,
       sum(case when behavior_type = 'pv' then 1 else 0 end) as pv,   --点击数
       sum(case when behavior_type = 'fav' then 1 else 0 end) as fav,  --收藏数
       sum(case when behavior_type = 'cart' then 1 else 0 end) as cart,  --加购物车数
       sum(case when behavior_type = 'buy' then 1 else 0 end) as buy  --购买数
from user_behavior
group by user_id;

--复购率：产生两次或两次以上购买的用户占购买用户的比例
select sum(case when buy > 1 then 1 else 0 end) / sum(case when buy > 0 then 1 else 0 end)
from user_behavior_count;
```

结论分析：2017-11-25 到 2017-12-03 这段时间，PV 总数为 89,660,671 ，UV 总数为 987,991。从日均访问量趋势来看，进入 12 月份之后有一个比较明显的增长，猜测可能是因为临近双 12 ，电商活动引流产生，另外，2017-12-02 和 2017-12-03 刚好是周末，也可能是周末的用户活跃度本来就比平常高。总体的复购率为 66.01%，说明用户的忠诚度比较高。

2）用户行为转换率

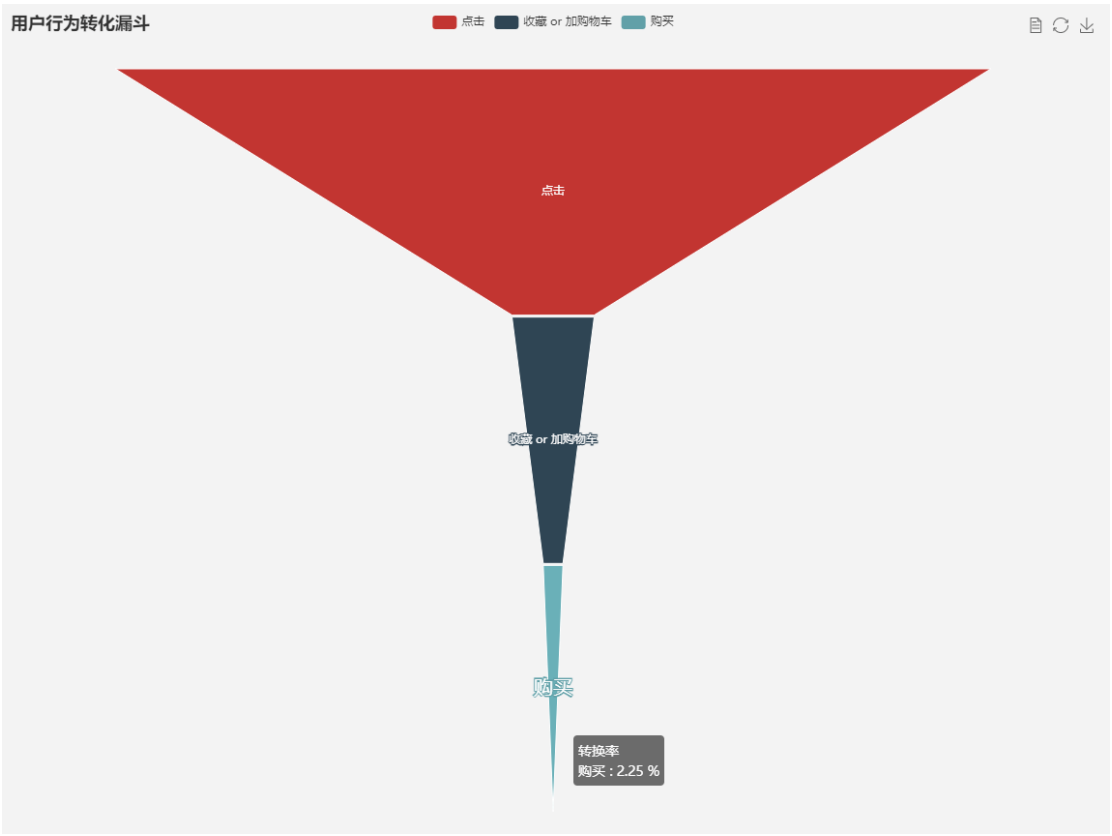
*<hive> Script-1

```
select a.pv,
       a.fav,
       a.cart,
       a.fav + a.cart as `fav+cart`,
       a.buy,
       round((a.fav + a.cart) / a.pv, 4) as pv2favcart,
       round(a.buy / (a.fav + a.cart), 4) as favcart2buy,
       round(a.buy / a.pv, 4) as pv2buy
from(
select sum(pv) as pv,    --点击数
       sum(fav) as fav,   --收藏数
       sum(cart) as cart,   --加购物车数
       sum(buy) as buy   --购买数
from user_behavior_count
) as a;
```

Result | Statistics

`select a.pv, a.fav, a.cart, a.fav + a.cart as `fav+cart`, a` 输入一个 SQL 表达式来过滤结果 (使用 Ctrl+Space)

	pv	fav	cart	fav+cart	buy	pv2favcart	favcart2buy	pv2buy
1	89,660,671	2,888,258	5,530,446	8,418,704	2,015,807	0.0939	0.2394	0.0225



结论: 2017-11-25 到 2017-12-03 这段时间，点击数为 89,660,671 ，收藏数为 2,888,258，加购物车数为 5,530,446，购买数为 2,015,807。总体的转化率为 2.25%，这个值可能是比较低的，从加到购物车数来看，有可能部分用户是准备等到电商节日活动才进行购买。所以合理推断：一般电商节前一段时间的转化率会比平常低。

3）用户行为习惯

```sql
-- 一天的活跃时段分布
select hour(datetime) as hour,
       sum(case when behavior_type = 'pv' then 1 else 0 end) as pv,   --点击数
       sum(case when behavior_type = 'fav' then 1 else 0 end) as fav,  --收藏数
       sum(case when behavior_type = 'cart' then 1 else 0 end) as cart,  --加购物车数
       sum(case when behavior_type = 'buy' then 1 else 0 end) as buy  --购买数
from user_behavior
group by hour(datetime)
order by hour;
```

可视化

```sql
--一周用户的活跃分布
select pmod(datediff(datetime, '1920-01-01') - 3, 7) as weekday,
       sum(case when behavior_type = 'pv' then 1 else 0 end) as pv,   --点击数
       sum(case when behavior_type = 'fav' then 1 else 0 end) as fav,  --收藏数
       sum(case when behavior_type = 'cart' then 1 else 0 end) as cart,  --加购物车数
       sum(case when behavior_type = 'buy' then 1 else 0 end) as buy  --购买数
from user_behavior
where date(datetime) between '2017-11-27' and '2017-12-03'
group by pmod(datediff(datetime, '1920-01-01') - 3, 7)
order by weekday;
```

Result | Statistics

select pmod(datediff(datetime, '1920-01-01') - 3, 输入一个 SQL 表达式来过滤结果 (使用 Ctrl+Space)

	weekday	pv	fav	cart	buy
1	0	12,237,300	392,197	774,905	257,754
2	1	9,041,186	291,221	541,904	226,834
3	2	8,842,932	289,100	534,157	211,997
4	3	9,210,820	298,587	551,593	223,068
5	4	9,358,998	302,264	565,015	221,459
6	5	9,718,956	307,115	623,346	210,010
7	6	12,329,641	396,749	793,569	257,903

可视化:

结论：？

4）基于 RFM 模型找出有价值的用户

RFM 模型是衡量客户价值和客户创利能力的重要工具和手段，其中由 3 个要素构成了数据分析最好的指标，分别是:

- R-Recency（最近一次购买时间）
- F-Frequency（消费频率）
- M-Money（消费金额）

```sql
--R-Recency（最近一次购买时间），R值越高，一般说明用户比较活跃
select user_id,
       datediff('2017-12-04', max(datetime)) as R,
       dense_rank() over(order by datediff('2017-12-04', max(datetime))) as R_rank
from user_behavior
where behavior_type = 'buy'
group by user_id
limit 10;

--F-Frequency（消费频率），F值越高，说明用户越忠诚
select user_id,
       count(1) as F,
       dense_rank() over(order by count(1) desc) as F_rank
from user_behavior
where behavior_type = 'buy'
group by user_id
limit 10;

--M-Money（消费金额），数据集无金额，所以就不分析这一项
```

对有购买行为的用户按照排名进行分组，共划分为 5 组，前 - 1/5 的用户打 5 分 前 1/5 - 2/5 的用户打 4 分 前 2/5 - 3/5 的用户打 3 分 前 3/5 - 4/5 的用户打 2 分 前 4/5 - 的用户打 1 分 按照这个规则分别对用户时间间隔排名打分和购买频率排名打分，最后把两个分数合并在一起作为该名用户的最终评分：

```sql
with cte as(
select user_id,
       datediff('2017-12-04', max(datetime)) as R,
       dense_rank() over(order by datediff('2017-12-04', max(datetime))) as R_rank,
       count(1) as F,
       dense_rank() over(order by count(1) desc) as F_rank
from user_behavior
where behavior_type = 'buy'
group by user_id)

select user_id, R, R_rank, R_score, F, F_rank, F_score,  R_score + F_score AS score
from(
select *,
       case ntile(5) over(order by R_rank) when 1 then 5
                                           when 2 then 4
                                           when 3 then 3
                                           when 4 then 2
                                           when 5 then 1
       end as R_score,
       case ntile(5) over(order by F_rank) when 1 then 5
                                           when 2 then 4
                                           when 3 then 3
                                           when 4 then 2
                                           when 5 then 1
       end as F_score
from cte
) as a
order by score desc
limit 20;
```

*<hive> Script-1

```
        datediff('2017-12-04', max(datetime)) as R,
        dense_rank() over(order by datediff('2017-12-04', max(datetime))) as R_rank,
        count(1) as F,
        dense_rank() over(order by count(1) desc) as F_rank
from user_behavior
where behavior_type = 'buy'
group by user_id)

select user_id, R, R_rank, R_score, F, F_rank, F_score,  R_score + F_score AS score
from(
select *,
```

Result | Statistics

```
with cte as( select user_id, datediff('2017-12-04', n  输入一个 SQL 表达式来过滤结果 (使用 Ctrl+Space)
```

	user_id	r	r_rank	r_score	f	f_rank	f_score	score
1	263296	1	1	5	74	18	5	10
2	98084	1	1	5	78	17	5	10
3	949195	1	1	5	78	17	5	10
4	537150	1	1	5	109	8	5	10
5	681083	1	1	5	82	16	5	10
6	234304	1	1	5	84	15	5	10
7	294043	1	1	5	85	14	5	10
8	490508	1	1	5	87	13	5	10
9	960681	1	1	5	73	19	5	10
10	337305	1	1	5	93	11	5	10
11	885763	1	1	5	4	87	5	10
12	1003412	1	1	5	100	9	5	10
13	704589	1	1	5	4	87	5	10
14	702034	1	1	5	159	3	5	10
15	432739	1	1	5	112	6	5	10
16	1014116	1	1	5	118	5	5	10

结论：可以根据用户的价值得分，进行个性化的营销推荐

（5）数据迁移

使用 Sqoop 将数据从 Hive 导入 MySQL。具体步骤可参考：

操作指南：https://dblab.xmu.edu.cn/blog/1367/