# 2018 阿里妈妈广告算法大赛调研方案

2018-11-08

# 目录

# 1.赛题描述

以阿里电商广告为研究对象，给定广告点击相关的用户（user）、广告商品（ad）、检索词（query）、上下文内容（context）、商店（shop）等信息的条件下预测广告产生购买行为的概率（pCVR）。形式化定义为：pCVR=P(conversion=1 | query, user, ad, context, shop)。属于 2 分类问题。

初赛和决赛分别为：

（1）日常的转化率预估；

（2）特殊日期的转化率预估；

# 2.评价指标（Logloss）

$$\log loss = -\frac{1}{N}\sum_{i=1}^{N}(y_i \log(p_i)+(1-y_i)\log(1-p_i))$$

# 3.数据格式

| 字段 | 字段解释 |
|---|---|
| instance_id | 序列编号 |
| item_id | 广告商品编号，Long 类型 |
| item_category_list | 广告商品的的类目列表，String 类型；从根类目（最粗略的一级类目）向叶子类目（最精细的类目）依次排列，数据拼接格式为 "category_0;category_1;category_2"，其中 category_1 是 category_0 的子类目，category_2 是 category_1 的子类目 |
| item_property_list | 广告商品的属性列表，String 类型；数据拼接格式为 "property_0;property_1;property_2"，各个属性没有从属关系 |
| item_brand_id | 广告商品的品牌编号，Long 类型 |
| item_city_id | 广告商品的城市编号，Long 类型 |
| item_price_level | 广告商品的价格等级，Int 类型；取值从 0 开始，数值越大表示价格越高 |
| item_sales_level | 广告商品的销量等级，Int 类型；取值从 0 开始，数值越大表示销量越大 |
| item_collected_level | 广告商品被收藏次数的等级，Int 类型；取值从 0 开始，数值越大表示被收藏次数越大 |
| item_pv_level | 广告商品被展示次数的等级，Int 类型；取值从 0 开始，数值越大表示被展示次数越大 |
| user_id | 用户的编号，Long 类型 |
| user_gender_id | 用户的预测性别编号，Int 类型；0 表示女性用户，1 表示男性用户，2 表示家庭用户 |
| user_age_level | 用户的预测年龄等级，Int 类型；数值越大表示年龄越大 |
| user_occupation_id | 用户的预测职业编号，Int 类型 |
| user_star_level | 用户的星级编号，Int 类型；数值越大表示用户的星级越高 |

| | |
|---|---|
| context_id | 上下文信息的编号，Long 类型 |
| context_timestamp | 广告商品的展示时间，Long 类型；取值是以秒为单位的 Unix 时间戳，以 1 天为单位对时间戳进行了偏移 |
| context_page_id | 广告商品的展示页面编号，Int 类型；取值从 1 开始，依次增加；在一次搜索的展示结果中第一屏的编号为 1，第二屏的编号为 2 |
| predict_category_property | 根据查询词预测的类目属性列表，String 类型；数据拼接格式为“category_A:property_A_1,property_A_2,property_A_3;category_B:-1;category_C:property_C_1,property_C_2”，其中 category_A、category_B、category_C 是预测的三个类目；property_B 取值为-1，表示预测的第二个类目 category_B 没有对应的预测属性 |
| shop_id | 店铺的编号，Long 类型 |
| shop_review_num_level | 店铺的评价数量等级，Int 类型；取值从 0 开始，数值越大表示评价数量越多 |
| shop_review_positive_rate | 店铺的好评率，Double 类型；取值在 0 到 1 之间，数值越大表示好评率越高 |
| shop_star_level | 店铺的星级编号，Int 类型；取值从 0 开始，数值越大表示店铺的星级越高 |
| shop_score_service | 店铺的服务态度评分，Double 类型；取值在 0 到 1 之间，数值越大表示评分越高 |
| shop_score_delivery | 店铺的物流服务评分，Double 类型；取值在 0 到 1 之间，数值越大表示评分越高 |
| shop_score_description | 店铺的描述相符评分，Double 类型；取值在 0 到 1 之间，数值越大表示评分越高 |

# 4.输入

见“数据格式”；

# 5.输出

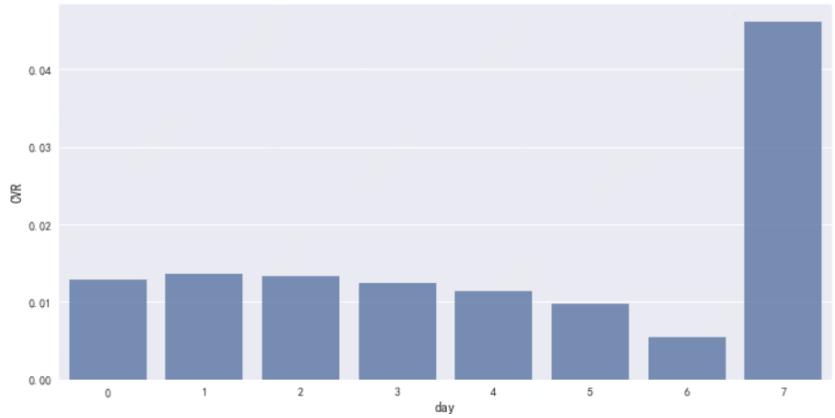
转化概率；

示例：

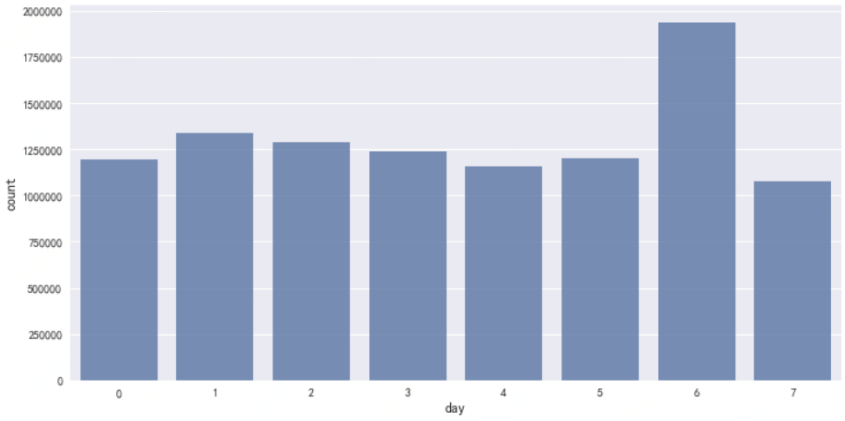
| instance_id | predicted_score |
|---|---|
| 2475218615076601065 | 0.9 |

# 6.数据分析

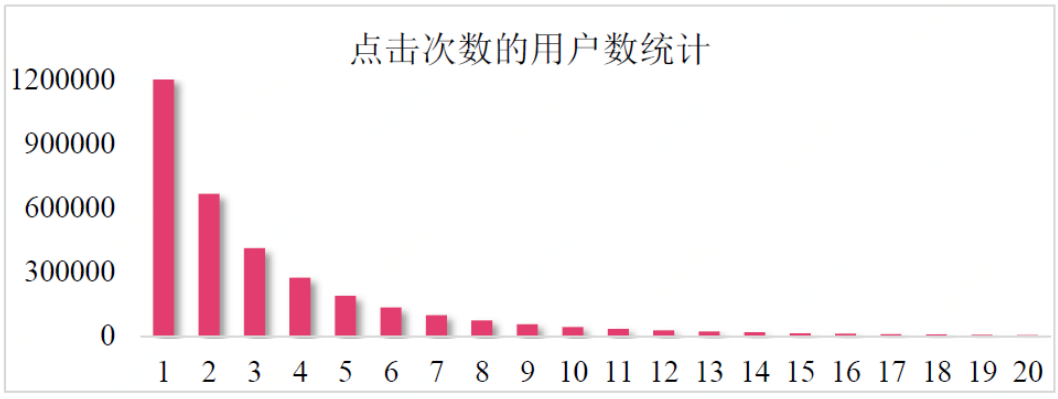
1. 正负样本比例：1:70（151,210 / 10,432,037）；
2. User: 2,958,506；
3. Item: 84,678；
4. 每天的转化率情况；




5. 每天的点击次数；




6. 用户点击行为次数分布；




7. 用户购买行为次数分布；

8. 数据情况总结：

（1） 发现每个人的平均点击及购买次数较少，所以和通常的根据用户的历史数据预测未来是否购买问题是不一样的，没必要进行滑窗统计 user_id 的特征；

（2） 是一个低频诉求的场景，及具有长尾分布形式；

# 7.Top1&2 团队算法

第一名：lightgbm；

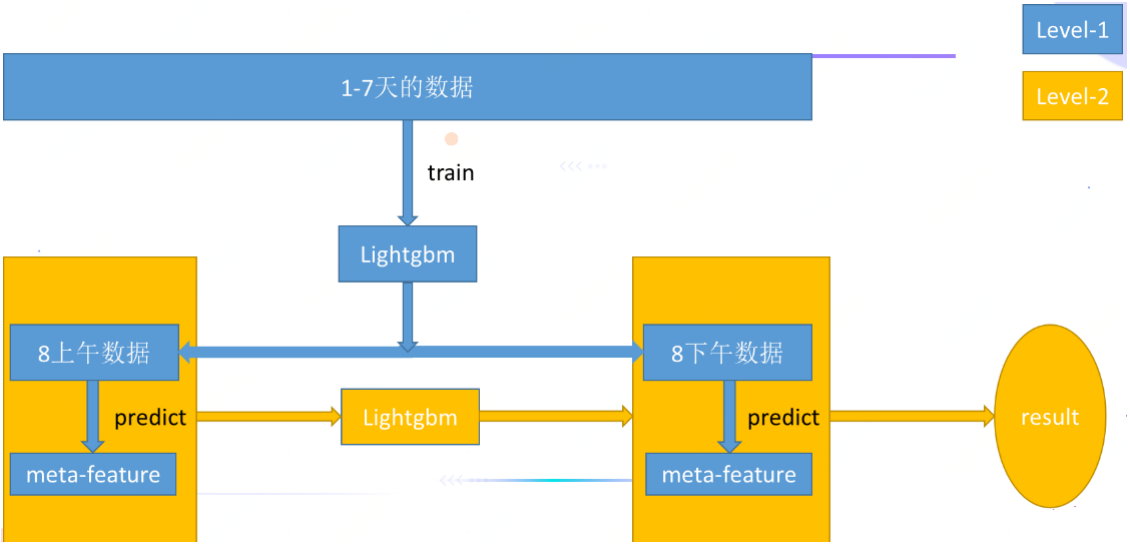
第二名：lightgbm；

第三名：

Lightgbm
Xgboost
Catboost
GBDT+LR
NN (DeepFFM,DeepFM,FNN)

# 8.Top1 团队模型&特征&主代码介绍

## 8.1 Top1 模型设计




（1） Level-1 将使用预热期的所有数据对这一时期的购物行为建模，并将它在购物节当天的预测作为第二个模型的输入；

（2） 单模型：lightgbm；

## 8.2 Top1 特征简单介绍

### 8.2.1 Embedding 特征

根据原数据提供的“item_property_list”和“predict_category_property”字段来分析。根据原始数据提供的 item 特征及搜索 item 特征，描述 user 属性偏好，然后统计有这些属性偏好的 user 和所搜索并点击的 item 之间的关系。即：用属性来表征 user_id，用 user_id 来表征 item_id；

**（1）Sample Embedding**

sample_emb_x=[x1,x2,x3,x4,...,xn]

解释：xn 为第 n 个 property 在不在 predict_category_property 中；

sample_emb_y=[y1,y2,y3,y4,...,yn]

解释：yn 为第 n 个 property 在不在 item_property_list 中；

**（2）User Embedding**

user_emb_x=mean([sample_emb_x_1,sample_emb_x_2,...,sample_emb_x_k])

解释：sample_emb_x_k 为该 user 的第 k 条样本的 sample_emb；

user_emb_y=mean([sample_emb_y_1,sample_emb_y_2,...,sample_emb_y_k])

解释：sample_emb_y_k 为该 user 的第 k 条样本的 sample_emb；

通过这种对所有样本的 sample_emb 做 mean 操作来对 user 做 embedding。

**（3）Item Embedding**

item_emb_x=mean([user_emb_x_1,user_emb_x_2,...,user_emb_x_k])

解释：user_emb_x_k 为该 item 的第 k 条样本的 user_emb；

item_emb_y=mean([user_emb_y_1,user_emb_y_2,...,user_emb_y_k])

解释：user_emb_y_k 为该 item 的第 k 条样本的 user_emb；
通过这种对所有样本的 use_emb 做 mean 操作来对 item 做 embedding。

该部分得到了 6*n 个特征，n 的大小视情况而定，这里取 count_top100 的 property 来做 embedding，所以总共 6*100 个特征。【示例可参考“参考链接（2）”】

### 8.2.2 统计特征

| |
|---|
| user 点击 item 个数 |
| user 最后一次搜索时间 |
| user 浏览展示页面的最大页数 |
| user 搜索的小时平均 |
| user 和 item 最后一次交互时间 |
| user 点击 item 所属品类的个数 |
| … |

### 8.2.3 时差特征

| |
|---|
| user 距离上次时长，距离下次时长 |
| user 与商品 item 交互距离上次时长，距离下次时长 |
| user 与商品品类 item_category 交互距离上次时长，距离下次时长 |
| user 与商品品牌 item_brand_id 交互距离上次时长，距离下次时长 |
| … |

### 8.2.4 排序特征

| |
|---|
| User 的第几次交互，倒数第几次交互 |
| User 与商品 item 的第几次交互，倒数第几次交互 |
| user 与商品品类 item_category 第几次交互，倒数第几次交互 |
| user 与商品品牌 item_brand_id 第几次交互，倒数第几次交互 |
| … |

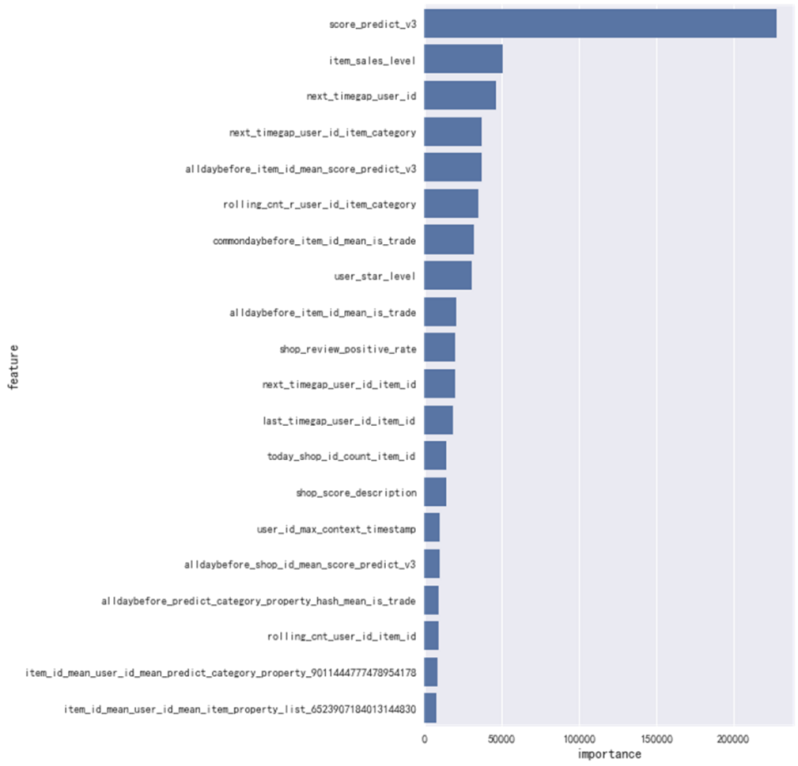
注：
（1）代码中只展示了 8.2.1 的示例 demo；

## 8.3 Top1 主代码&特征重要度

### 8.3.1 主代码展示

```
    # 个数特征
    train = get_cnt_feature(train, ["user_id", "item_category"], "item_price_level", True)
    train = get_cnt_feature(train, ["user_id"], "item_id", False)
    # 时差特征
    train = get_timegaps(train, ["user_id"], time_col="context_timestamp")
    train = get_timegaps(train, ["user_id", "item_category"], time_col="context_timestamp")
    train = get_timegaps(train, ["user_id", "item_brand_id"], time_col="context_timestamp")
    train = get_timegaps(train, ["user_id", "item_id"], time_col="context_timestamp")
    # 排序特征
    train = get_rolling_count(train, ["user_id"])
    train = get_rolling_count(train, ["user_id", "item_category"])
    train = get_rolling_count(train, ["user_id", "item_id"])
    train = get_rolling_count(train, ["user_id", "predict_category_property_hash"])
    train = get_rolling_count(train, ["user_id", "dayhour"])
    train = get_rolling_count(train, ["user_id", "item_category", "dayhour"])

    # 统计特征
    train = get_type_feature(train, ["user_id"], "item_id", "nunique")
    train = get_type_feature(train, ["item_category"], "item_id", "nunique")

    train = get_type_feature(train, ["shop_id", "day"], "instance_id", "count")

    train = get_type_feature(train, ["user_id"], "context_timestamp", "max")
    train = get_type_feature(train, ["user_id"], "context_page_id", "max")
    train = get_type_feature(train, ["user_id", "item_id"], "context_timestamp", "max")
    train = get_type_feature(train, ["user_id", "predict_category_property_hash"], "context_page_id", "max")
    train = get_type_feature(train, ["item_category"], "category_pre_index", "mean")
    train = get_type_feature(train, ["user_id"], "category_pre_index", "mean")
    train = get_type_feature(train, ["user_id"], "hour", "mean")

#表征特征
num = 0
for i in list(property_list.property)[:100]:
    num += 1
    if i != "-1":
        print(i)
        if num <= 100:
            train = get_cat_feature(train, "item_property_list", i)
            train = get_cat_feature(train, "predict_category_property", i)

        if num <= 100:
            train = get_type_feature(train, ["user_id"], "predict_category_property" + "_" + str(i), "mean")
            train = get_type_feature(train, ["user_id"], "item_property_list" + "_" + str(i), "mean")

        if num <= 100:
            train = get_type_feature(train, ["item_id"], "user_id_mean_predict_category_property" + "_" + str(i),"mean")
            train = get_type_feature(train, ["item_id"], "user_id_mean_item_property_list" + "_" + str(i), "mean")
```

### 8.3.2 特征重要度结果展示

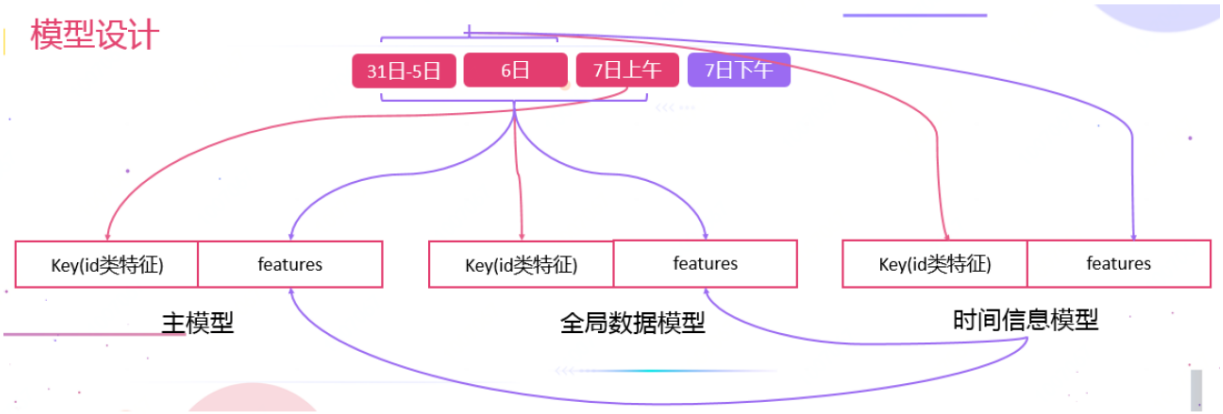


# 9.Top2 团队模型&特征介绍

## 9.1 Top2 模型设计

### 9.1.1 数据划分


### 9.1.2 模型设计




采用三种方式训练模型：
（1）主模型，使用 7 号上午的数据作为训练样本，对 31-5 号，6 号，7 号数据提取特征；
（2）全局数据模型，使用全部带标签的样本作为训练数据，使用全部数据提取特征；
（3）时间信息模型，使用 31-6 号的数据作为训练样本，对 31-6 号的数据提取特征。
时间信息模型对 7 号当天的样本进行预测，将预测结果作为新的特征添加到（1）和（2）模型中，弥补前面模型对时间刻画的缺失。

## 9.2 Top2 特征详细介绍

### 9.2.1 第一部分：基础特征群（3+14=17 维）

**（1） 转化时间戳，提取时间特征：天、小时；（3 维）**

（1.1）天；（1 维）

（1.2）小时维度序号_1h；（1 维）

（1.3）30min 维度序号_30min；（1 维）

**（2） 统计特征：item_category_list，item_property_list，predict_category_property 字段的类别与属性；（14 维）**

（2.1）item_category_list 与 predict_category_property 相同的 category 数量；（1 维）

（2.2）item_property_list 与 predict_category_property 相同的 property 数量；（1 维）

（2.3）item_category _list 中的第二个类别（下称‘cate’，id 类，可理解为 2 级品类。数据中的 item_category _list 字段第一个类别全部相同）；（1 维）

（2.4）item_property_list 字段，统计属性的数量；（1 维）

（2.5）predict_category_property 字段，统计类别的数量；（1 维）

（2.6）predict_category_property 字段，统计属性的数量；（1 维）

（2.7）predict_category_property 字段，提取第一个类别（下称‘query1’，id 类）；（1 维）

（2.8）predict_category_property 字段，提取全部类别（下称‘query’，以‘_’连接，形成 id 类特征）；（1 维）

（2.9）item_property_list 字段，提取 top1 属性；（解释：使用全部点击数据，统计 property 出现的次数，根据次数从大到小对当前样本 item_property_list 字段中的 property 排序，提取 top1 的 property， id 类）（1 维）

（2.10）item_property_list 字段，提取 top2 属性-top5 属性，top10 属性；（解释：使用全部点击数据，统计 property 出现的次数，根据次数从大到小对当前样本 item_property_list 字段中的 property 排序，取 top2/top3/top4/top5/top10 的 property，以‘_’连接，id 类）（5 维）

注：

（1）item_category_list，item_property_list，predict_category_property 信息见“数据格式表”；

（2）后续代码中使用 query 指代“查询操作后的预测类别”，有 3 种情况，分别为：

（1.1）query：predict_category_property 字段的全部类别，以‘_’连接，id 类特征；

（1.2）query1：predict_category_property 字段的第一个类别， id 类特征；

（1.3）predict_category_property：predict_category_property 字段本身，id 类特征；

（3）top1 示例：2636395404473730413；

（4）top2 示例：2636395404473730413_9148482949976129397；

（5）将 id 类特征转化为序号，使用的是 LabelEncoder().fit_transform 函数。

## 9.2.2 第二部分：查询交互，用户交互，竞争特征（60+52+16=128 维）

**（1） 查询交互特征；【时间泄露】（同一 user，当前样本的时间戳，60 维）**

（1.1）当前样本之前（全部数据开始时间戳-当前时间戳） / 之后（当前时间戳-全部数据结束时间戳），该 user 点击相同的 query / query1 / predict_category_property 次数；（2*3=6 维）

（1.2）当前样本之前 / 之后，该 user 点击相同的 query / query1 / predict_category_property、相同 item / shop / brand / city / item_category_list / context_page_id 的次数；（2*3*6=36 维）

（1.3）当前样本之前 / 之后，该 user 点击了除当前 query / query1 / predict_category_property 之外的 unique(query) / unique(query1) / unique(predict_category_property) 数量；（2*3=6 维）

（1.4）查询该 user 在当前时间戳的 query / query1 / predict_category_property 在当天数据中第一次及最后一次出现的时间戳，统计 user 在最小时间戳之前 / 最大时间戳之后点击 unique(shop) / unique(item) 数量；（3*2*2=12 维）

注：

（1）item / shop / brand / city / item_category_list / context_page_id 信息见“数据格式表”；

**（2） 用户交互特征；【时间泄露】（同一 user，当天，52 维）**

（2.1）user 上下两次点击之间，最大时间间隔（当天，时间单位：秒）；（1 维）

（2.2）user 上下两次点击之间，最小时间间隔（当天，时间单位：秒）；（1 维）

（2.3）user 上下两次点击之间，平均时间间隔（当天，时间单位：秒）；（1 维）

（2.4）user 上下两次点击之间，中值时间间隔（当天，时间单位：秒）；（1 维）

（2.5）当前样本时间戳，距离当天第一次点击时间间隔（时间单位：秒）；（1 维）

（2.6）当前样本时间戳，距离当天最后一次点击时间间隔（时间单位：秒）；（1 维）

（2.7）当前样本时间戳，距离当天上次点击时间间隔（时间单位：秒）；（1 维）

（2.8）当前样本时间戳，距离当天下次点击时间间隔（时间单位：秒）；（1 维）

（2.9）当前样本时间戳，user 之前 / 之后点击 unique(query) / unique(query1) / unique(predict_category_property)的占比（前 / 后占比相加等于 1）；（2*3=6 维）

（2.10）当前样本时间戳，user 之前 / 之后点击 unique(item) / unique(shop) / unique(brand) / unique(city) 的占比（前 / 后占比相加等于 1）；（2*4=8 维）

（2.11）当前样本时间戳，user 之前 / 之后点击与该样本相同的 item / shop / brand / city / item_category_list 的 unique(query) / unique(query1) / unique(predict_category_property)的数量；（2*5*3=30 维）

**（3） 竞争特征；（同一 user，当天，16 维）**

（3.1）当前样本时间戳，user 之前 / 之后点击了价格更低的 unique(item)数量；（2 维）

（3.2）当前样本时间戳，user 之前 / 之后点击了销量更高的 unique(item)数量；（2 维）

（3.3）当前样本时间戳，user 之前 / 之后点击了评价数量更多的 unique(shop)数量；（2 维）

（3.4）当前样本时间戳，user 之前 / 之后点击了好评率更高的 unique(shop)数量；（2 维）

（3.5）当前样本时间戳，user 之前 / 之后点击了星级更高的 unique(shop)数量；（2 维）

（3.6）当前样本时间戳，user 之前 / 之后点击了服务态度更高的 unique(shop)数量；（2 维）

（3.7）当前样本时间戳，user 之前 / 之后点击了物流更好的 unique(shop)数量；（2 维）

（3.8）当前样本时间戳，user 之前 / 之后点击了描述平均分更高的 unique(shop)数量；（2 维）

## 9.2.3 第三部分：统计转化率特征，转化率排名特征（149+63=212 维）

**（1） 统计类特征；（10+36+2+10+36+10+45 = 149 维）**

**（1.1）前 6 天（包括第 6 天）的统计单特征，作为第 7 天的特征；（10 维）**

（1.1.1）前 6 天（包括第 6 天）user 的总转化率；（平滑处理）（1 维）

（1.1.2）前 6 天（包括第 6 天）item / brand / shop / city / item_category_list / context_page_id / query / query1 / predict_category_property 的总转化率；（9 维）

**（1.2）前 6 天（包括第 6 天）的统计交叉特征，作为第 7 天的特征；（36 维）**

id 特征列表如下：

['item',
'brand',
'shop',
'item_category_list',
'city',
'predict_category_property',
'query1',
'query',
'context_page_id']

（1.2.1）前 6 天（包括第 6 天），item 与其之后 8 个 id 特征，交叉后的转化率；（8 维）

（1.2.2）前 6 天（包括第 6 天），brand 与其之后 7 个 id 特征，交叉后的转化率；（7 维）

（1.2.3）前 6 天（包括第 6 天），shop 与其之后 6 个 id 特征，交叉后的转化率；（6 维）

（1.2.4）前 6 天（包括第 6 天），item_category_list 与其之后 5 个 id 特征，交叉后的转化率；（5 维）

（1.2.5）前 6 天（包括第 6 天），city 与其之后 4 个 id 特征，交叉后的转化率；（4 维）

（1.2.6）前 6 天（包括第 6 天），predict_category_property 与其之后 3 个 id 特征，交叉后的转化率；（3 维）

（1.2.7）前 6 天（包括第 6 天），query1 与其之后 2 个 id 特征，交叉后的转化率；（2 维）

（1.2.8）前 6 天（包括第 6 天），query 与其之后 1 个 id 特征，交叉后的转化率；（1 维）

**（1.3）前 6 天和第 6 天的统计特征（用户行为编码），作为第 7 天的特征；（2 维）**

（1.3.1）前 6 天（包括第 6 天），user 的‘点击次数_购买次数’（id 类特征）；（1 维）

（1.3.2）第 6 天，user 的‘点击次数_购买次数’（id 类特征）；（1 维）

**（1.4）第 6 天的统计单特征，作为第 7 天的特征；（10 维）**

（1.4.1）第 6 天，user 的总转化率；（平滑处理）（1 维）

（1.4.2）第 6 天，item / brand / shop / city / item_category_list / context_page_id / query / query1 / predict_category_property 的总转化率；（9 维）

**（1.5）第 6 天的统计交叉特征，作为第 7 天的特征；（36 维）**

id 特征列表如下：

['item',
'brand',
'shop',
'item_category_list',

'city',
'predict_category_property',
'query1',
'query',
'context_page_id']

（1.5.1）第 6 天，item 与其之后 8 个 id 特征，交叉后的转化率；（8 维）
（1.5.2）第 6 天，brand 与其之后 7 个 id 特征，交叉后的转化率；（7 维）
（1.5.3）第 6 天，shop 与其之后 6 个 id 特征，交叉后的转化率；（6 维）
（1.5.4）第 6 天，item_category_list 与其之后 5 个 id 特征，交叉后的转化率；（5 维）
（1.5.5）第 6 天，city 与其之后 4 个 id 特征，交叉后的转化率；（4 维）
（1.5.6）第 6 天，predict_category_property 与其之后 3 个 id 特征，交叉后的转化率；（3 维）
（1.5.7）第 6 天，query1 与其之后 2 个 id 特征，交叉后的转化率；（2 维）
（1.5.8）第 6 天，query 与其之后 1 个 id 特征，交叉后的转化率；（1 维）

**（1.6）第 7 天上午的统计单特征，作为第 7 天下午的特征；（存在过拟合现象）（10 维）**

（1.6.1）第 7 天上午，user 的转化率；（1 维）
（1.6.2）第 7 天上午，item / brand / shop / city / item_category_list / context_page_id / query / query1 / predict_category_property 的交易率（mean）；（9 维）

**（1.7）第 7 天上午的统计交叉特征，作为第 7 天下午的特征；（存在过拟合现象）（45 维）**

id 特征列表如下：
['user',
'item',
'brand',
'shop',
'item_category_list',
'city',
'predict_category_property',
'query1',
'query',
'context_page_id']

（1.7.1）第 7 天上午，user 与其之后 9 个 id 特征，交叉后的交易率（mean）；（9 维）
（1.7.2）第 7 天上午，item 与其之后 8 个 id 特征，交叉后的交易率（mean）；（8 维）
（1.7.3）第 7 天上午，brand 与其之后 7 个 id 特征，交叉后的交易率（mean）；（7 维）
（1.7.4）第 7 天上午，shop 与其之后 6 个 id 特征，交叉后的交易率（mean）；（6 维）
（1.7.5）第 7 天上午，item_category_list 与其之后 5 个 id 特征，交叉后的交易率（mean）；（5 维）
（1.7.6）第 7 天上午，city 与其之后 4 个 id 特征，交叉后的交易率（mean）；（4 维）
（1.7.7）第 7 天上午，predict_category_property 与其之后 3 个 id 特征，交叉后的交易率（mean）；（3 维）
（1.7.8）第 7 天上午，query1 与其之后 2 个 id 特征，交叉后的转化率；（2 维）
（1.7.9）第 7 天上午，query 与其之后 1 个 id 特征，交叉后的转化率；（1 维）

**（2） 排名特征；（12+18+15+12+6=63 维）**

（2.1）前 6 天 / 第 6 天 / 第 7 天上午，user 转化率在 brand / shop / item_category_list / city 下面的

排名；（3*4=12 维）

（2.2）前 6 天 / 第 6 天 / 第 7 天上午，item 转化率在 brand / shop / item_category_list / city / query1 / query 下面的排名；（3*6=18 维）

（2.3）前 6 天 / 第 6 天 / 第 7 天上午，shop 转化率在 brand / item_category_list / city / query1 / query 下面的排名；（3*5=15 维）

（2.4）前 6 天 / 第 6 天 / 第 7 天上午，brand 转化率在 shop / city / query1 / query 下面的排名；（3*4=12 维）

（2.5）前 6 天 / 第 6 天 / 第 7 天上午，item_category_list 转化率在 query1 / query 下面的排名；（3*2=6 维）

## 9.2.4 第四部分：统计点击特征，点击数量占比特征（36+198+48=282 维）

**（1）前 6 天 / 第 6 天 / all 天，统计单 id 类点击数特征；（36 维）**

（1.1）前 6 天 / 第 6 天 / all days，user / item / brand / shop / item_category_list / city / cate/ top10_property / predict_category_property / context_page_id / query / query1 点击数；（3*12=36 维）

**（2）前 6 天 / 第 6 天 / all 天，统计 id 特征交叉特征的点击数量（3* 11*12/2 =198 维）**

id 列表如下：

['user',
'item',
'brand',
'shop',
'item_category_list',
'city',
'cate',
'top10_property',
'predict_category_property',
'context_page_id',
'query1',
'query']

（2.1）前 6 天 / 第 6 天 / all days，user 与其之后 11 个 id 特征，交叉后的点击数量；（11 维）

（2.2）前 6 天 / 第 6 天 / all days，item 与其之后 10 个 id 特征，交叉后的点击数量；（10 维）

（2.3）前 6 天 / 第 6 天 / all days，brand 与其之后 9 个 id 特征，交叉后的点击数量；（9 维）

（2.4）前 6 天 / 第 6 天 / all days，shop 与其之后 8 个 id 特征，交叉后的点击数量；（8 维）

（2.5）前 6 天 / 第 6 天 / all days，item_category_list 与其之后 7 个 id 特征，交叉后的点击数量；（7 维）

（2.6）前 6 天 / 第 6 天 / all days，city 与其之后 6 个 id 特征，交叉后的点击数量；（6 维）

（2.7）前 6 天 / 第 6 天 / all days，cate 与其之后 5 个 id 特征，交叉后的点击数量；（5 维）

（2.8）前 6 天 / 第 6 天 / all days，predict_category_property 与其之后 4 个 id 特征，交叉后的点击数量；（4 维）

（2.9）前 6 天 / 第 6 天 / all days，context_page_id 与其之后 3 个 id 特征，交叉后的点击数量；（3 维）

（2.10）前 6 天 / 第 6 天 / all days，query1 与其之后 2 个 id 特征，交叉后的点击数量；（2 维）

（2.11）前 6 天 / 第 6 天 / all days，query 与其之后 1 个 id 特征，交叉后的点击数量；（1 维）

**（3）前 6 天 / 第 6 天 / all 天，分别统计指定的 16 个交叉点击数量，计算占比特征；（3*16=48 维）**
**（说明：自己指定，未全部列举）**

[count(['user','query']) / count(query),
count(['user','query1']) / count(query1),
count(['user','shop']) / count(shop),
count(['user','item']) / count(item),
count(['item','shop']) / count(shop),
count(['item', 'brand']) / count(brand),
count(['item','city']) / count(city),

```
count(['item','cate']) / count(cate),
count(['item','top10_property']) / count(top10_property),
count(['item','context_page_id']) / count(context_page_id),
count(['item','query1']) / count(query1),
count(['item','item_category_list']) / count(item_category_list),
count(['item','query']) / count(query),
count(['brand', 'shop']) / count(shop),
count(['shop','city']) / count(city),
count([ 'shop','context_page_id']) / count(context_page_id)]
```

### 9.2.5 第五部分：不购买_购买特征，趋势特征，一次性购买特征，出现_购买特征，item_shop 属性变化特征（7+225+12+3+40=287 维）

**（1）不购买_购买特征；（7 维）**

（1.1）统计 user 前 6 天连续点击行为（未购买）最大次数；
（1.2）统计 user 第 6 天连续点击行为（未购买）最大次数；
（1.3）统计 user 第 6 天购买的 unique(item)个数；
（1.4）统计 user 第 6 天点击未购买 unique(item)个数；
（1.5）统计 user 第 6 天购买 item 所属的 unique(shop)个数；
（1.6）统计 user 第 6 天点击未购买 item 所属的 unique(shop)个数；
（1.7）计算 user 第 6 天 (1.4) / (1.6)；

**（2）趋势特征；（54+63+108=225 维）**

（2.1）统计过去第 1,2,3,4,5,7 天（共 6 天，第 6 天的点击数据已经被统计）item / brand / shop / item_category_list / city / predict_category_property / context_page_id / query1 / query 点击数量；（6*9=54 维）

（2.2）统计过去每天（共 7 天）item / brand / shop / item_category_list / city / predict_category_property / context_page_id / query1 / query 购买数量；（7*9=63 维）

（2.3）统计过去 6 天的 item / brand / shop / item_category_list / city / predict_category_property / context_page_id / query1 / query，点击 / 购买趋势（后一天的数据除以当前天的数据），比上一天高为 1，否则为 0；（6 *9*2 维=108 维）；

**（3）一次性购买特征；（12 维）**

解释：一次性购买，即转化率=1。

（3.1）前 6 天 / 第 6 天 / 第 7 天上午，[user, item]一次性购买的次数/item 购买次数（3*1 维）；
（3.2）前 6 天 / 第 6 天 / 第 7 天上午，[user, shop]一次性购买的次数/shop 购买次数（3*1 维）；
（3.3）前 6 天 / 第 6 天 / 第 7 天上午，[user, query]一次性购买的次数/query 购买次数（3*1 维）；
（3.4）前 6 天 / 第 6 天 / 第 7 天上午，[user, query1]一次性购买的次数/ query1 购买次数（3*1 维）；

**（4）出现_购买特征；（3 维）**

（4.1）第 6 天，item_id / query / query1 第一次出现到购买的时间差（时间单位：秒）；（3 维）

**（5）item_shop 属性变化特征（40 维）**

（5.1）item_price_level / item_sales_level / item_collected_level / item_pv_level

计算 4 个属性：前 6 天的方差、均值；第 7 天均值与前 6 天均值的差值；第 7 天均值与第 6 天均值的差值；（4*2+4+4=16 维）

（5.2）shop_review_num_level / shop_review_positive_rate / shop_star_level / shop_score_service / shop_score_delivery / shop_score_description

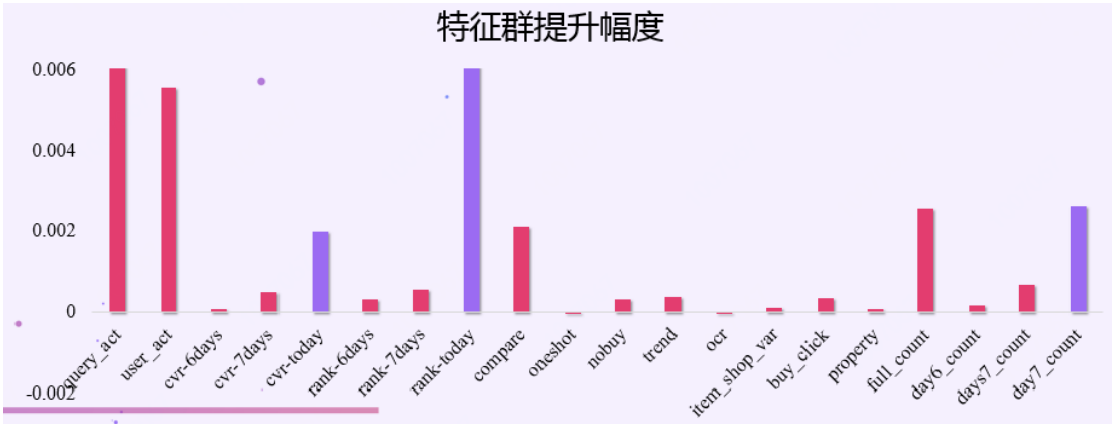
计算 6 个属性：前 6 天的方差 / 均值；第 7 天均值与前 6 天均值的差值；第 7 天均值与第 6 天均值的差值；（6*2+6+6=24 维）

### 9.2.6 第六部分：强制 cross 特征（100*99/2=4950 维）

**（1）使用上面特征训练模型，选择 top100 的特征强制相除，组合成新的特征；**

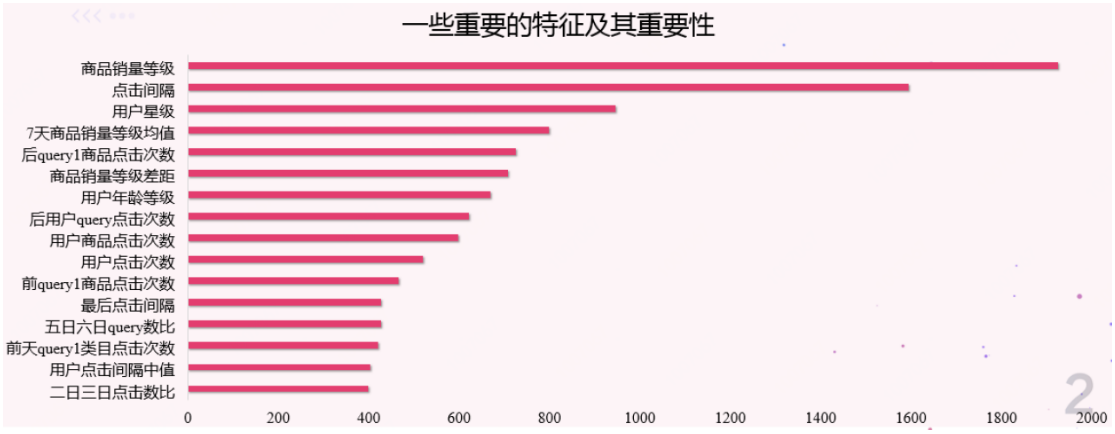
## 9.3 Top2 特征重要度

### 9.3.1 特征群重要度结果展示




query（查询）交互特征；
user 交互特征；
当天转化率；
转化率排名特征；
竞争特征；
统计点击特征&占比特征；
第 7 天点击统计特征；

### 9.3.2 特征重要度结果展示



# 参考链接

（1）官方网址：
https://tianchi.aliyun.com/competition/information.htm?spm=5176.100067.5678.2.5b0e2163b2Tv1G&raceId=231647
（2）第一名 github 地址：https://github.com/plantsgo/ijcai-2018/blob/master/eda_solve.ipynb
（3）第二名 github 地址：https://github.com/YouChouNoBB/ijcai-18-top2-single-mole-solution
（4）第二名思路讲解：https://tianchi.aliyun.com/forum/videoStream.html#postsId=5531
（5）第三名 github 地址：https://github.com/luoda888/2018-IJCAI-top3