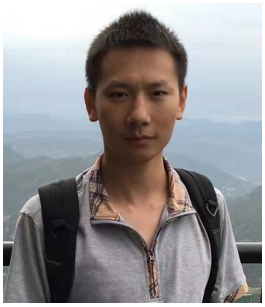
蓝鲸烧香隊-比赛攻略

一、 团队介绍

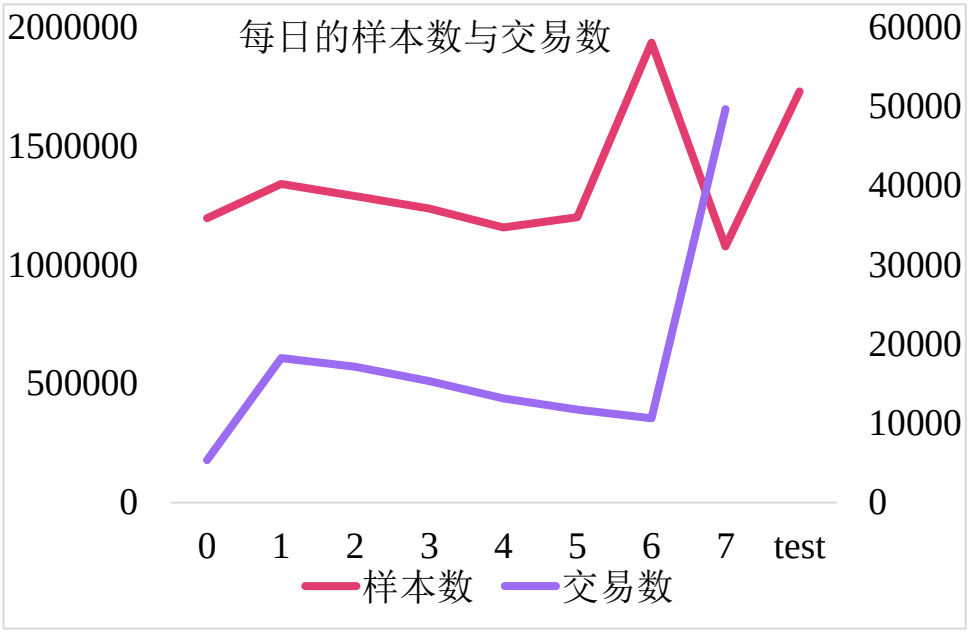
队长：BRYAN	队员：桑榆	队员：李困困
		
数据挖掘从业者，国内数据挖掘竞赛名将，天池数据科学家，IJCAI-17 冠军获得者。曾多次在国内外著名赛事中取得名次。	数据挖掘从业者，国内数据挖掘竞赛名将，天池数据大师，IJCAI-17 冠军获得者。曾多次在国内外著名赛事中取得名次。	数据挖掘从业者，国内数据挖掘竞赛名将。曾取得 CCF-蚂蚁金服-商场定位赛冠军等多项国内外著名赛事的名次。

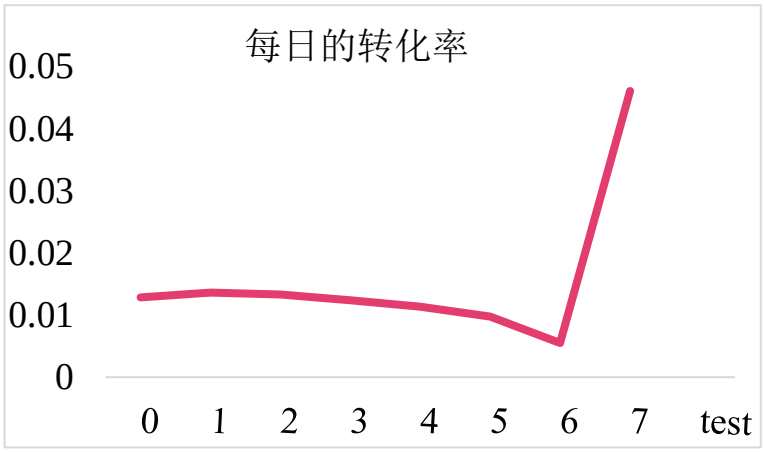
二、 赛题背景分析及理解

本赛题为搜索广告转化预估问题，一条样本包含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)。可以将问题抽象为二分类问题，重点对用户，商品，检索词，上下文，商店进行特征刻画，来训练模型。

三、 核心思路

(1) 数据分析





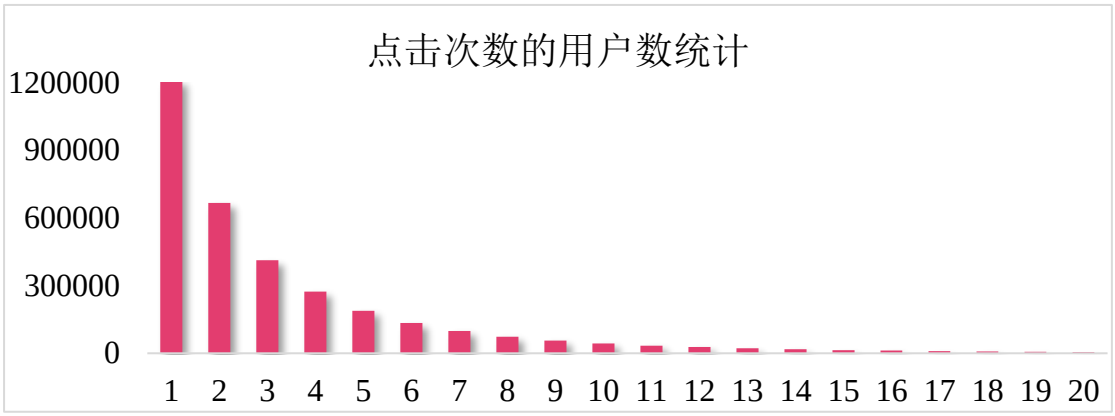
通过数据分析我们发现，训练数据的前 7 天转化率维持在 1%左右，但是在 6 号转化率偏低，在预测当天 7 号的上午转化率超过 4%，所以这是一个对特定促销日进行预测的问题。重点需要刻画用户，商品，店铺，检索词等关键信息在预测日前面 7 天的行为，预测日前一天的行为，预测日当天的行为。

另外 7 号的样本量远远超过前面每天的样本量均值，是我们重点需要关注的时间区间。主模型是基于 7 号上午的样本进行训练，前面

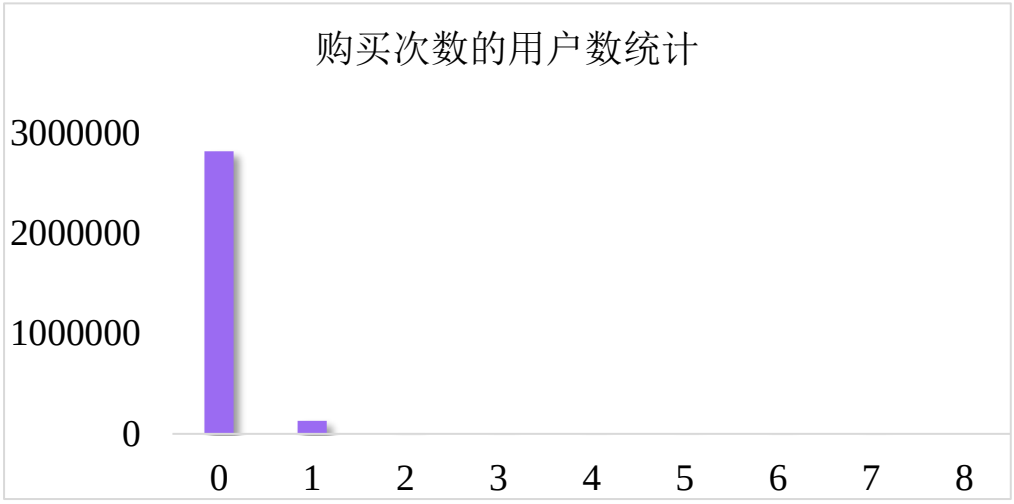
7 天的数据辅助训练。由于预估时间为 7 号下午，时间相关的特征没法在上午训练，为了弥补 7 号上午训练带来是数据和信息损失，我们队伍采用了两种方式:，一种采用前面 7 天训练模型预估 7 号得到概率作为新的特征，一种是训练 7 号之前以及 7 号上午训练全量模型进行加权。

(2)　用户分析

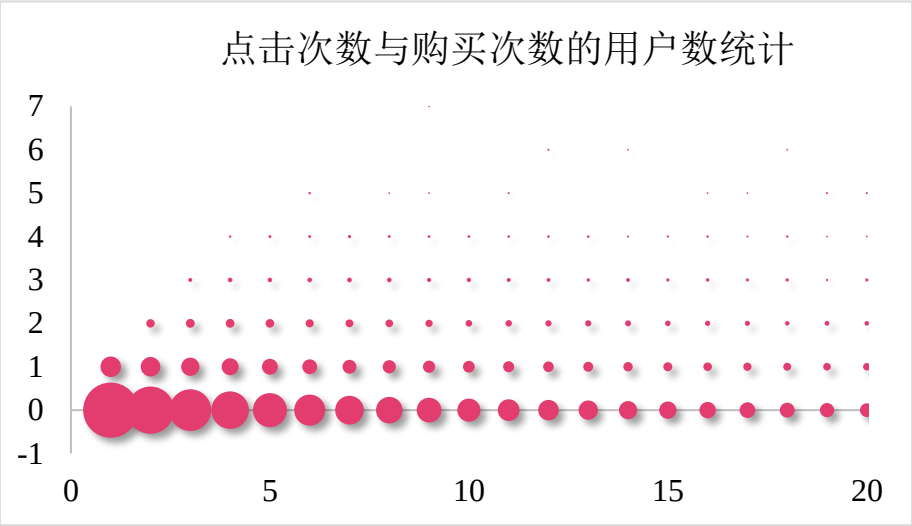
从下图可以看到大部分用户的点击次数集中在 5 次以下，8 天的时间内点击 5 次，说明这是一个低频诉求的场景。



在下图中可以看到，大部分用户都没有购买行为，少量用户购买了一次，本次竞赛的目标预测用户是否购买，少量的购买行为构成了数据的长尾分布形势。



下图是用户点击次数和购买次数的关系，横轴点击数，纵轴购买数。可以看到数据是呈左下角分布的趋势，也就是说购买行为发生在少量点击次数的情况下，说明这是一个即时兴趣，目标明确的场景。我们需要重点刻画用户当前状态。



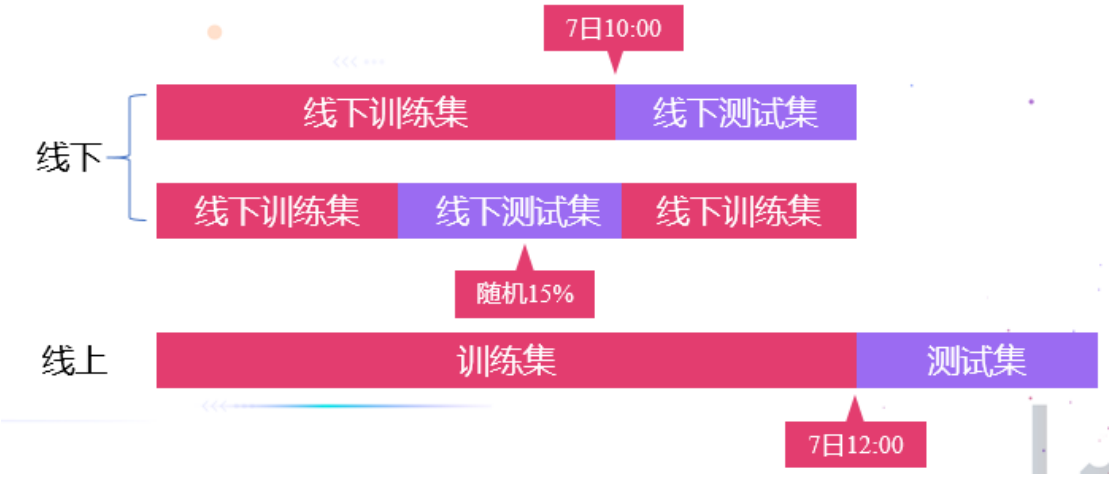
从用户分析中，我们发现，点击一次的用户占据较大的比例，这部分无法通过历史行为的特征刻画表征，因此提供的 query 信息是表征这部分用户的关键；同时，绝大部分用户没有发生购买行为，因此，负样本中包含了大量的信息，另外评估指标是 logloss，需要精确预测购买概率，所以并未对负样本进行采样，避免破坏正负样本分布。

(3) 预处理

缺失值填充：id 类特征使用众值填充，数值特征均值填充

挖掘隐藏信息：针对 item_property_list 列，统计 property 出现次数，保留出现次数的 top1~top10 作为新的 id 特征；针对 predict_category_property 列，直接按顺序保留 top1~top10 的类别作为新的 id 特征。针对 item_category_list 列，因为第一个大类都相同，取第二个类别作为新的 id 类特征。

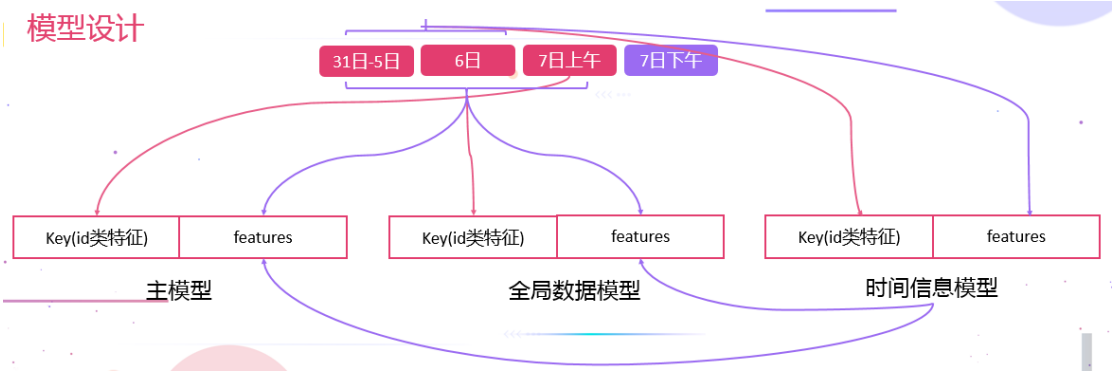
(4) 线下划分



由于线上提交的次数有限，因此，建立稳定的线下是取胜的关键。为提升我们优化算法的效率，减少线上成绩的运气性成分，同时避免我们的算法过度依赖于线上数据集，我们认真地进行了线下测试，分别采取 7 日上午最后两个小时，7 日上午随机 15%的数据进行验证，只有两者在线下均有提升，我们才进行线上提交。因此，我们始终确保我们在线上验证的优化在线下均有显著的提升。

(5) 模型设计

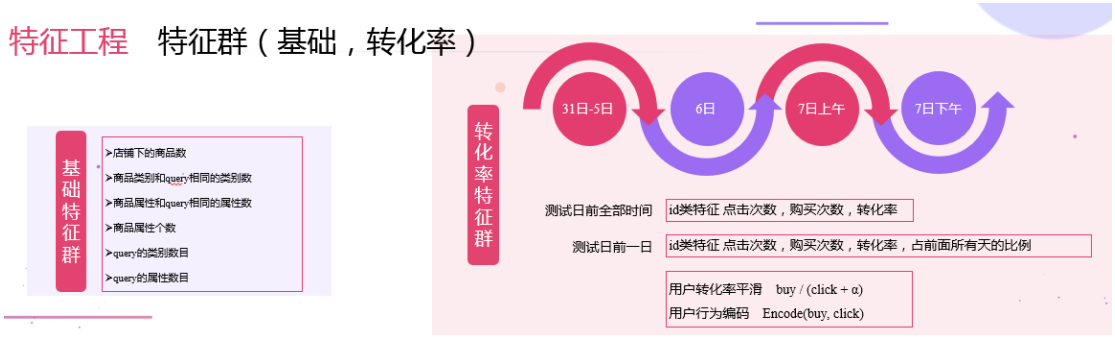
我们采用了 3 种数据划分方式训练模型，主模型使用 7 号上午的数据作为训练样本，对 31-5 号，6 号，7 号的数据提取特征。全局数据模型使用全部带标签的样本作为训练样本，使用全部数据提取特征。时间信息模型使用 31-6 号的数据作为训练样本，对 31-6 数据提取特征。训练的时间信息模型对 7 号全天的样本进行预测，将预测结果(携带了时间信息)作为新的特征添加到前面的模型中，来弥补前面模型对时间刻画的缺失。



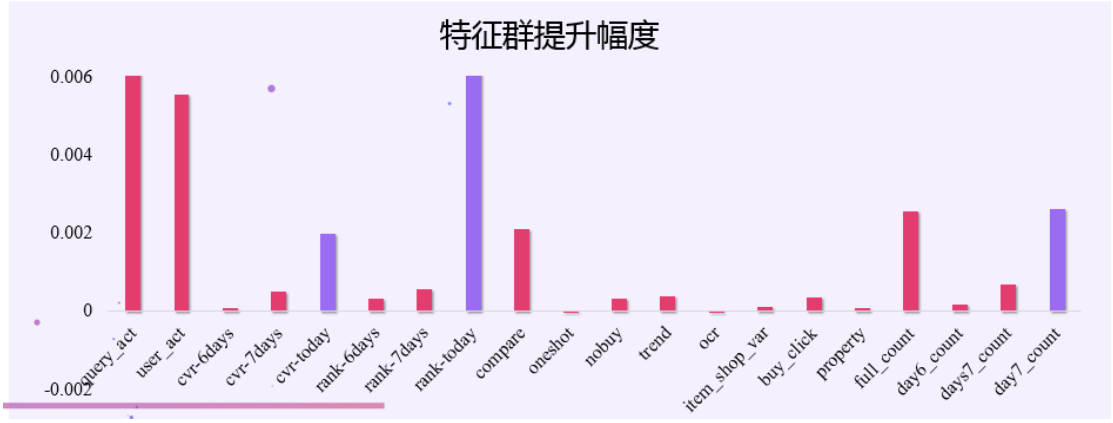
(6) 特征工程

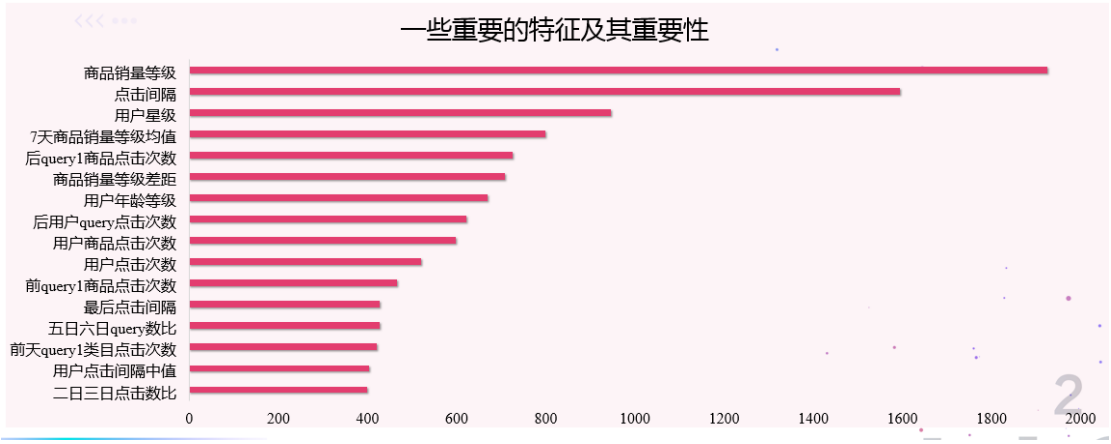
特征工程是模型提分的关键，我们从简单到复杂建立了基础特征群，转化率特征群，排名特征群，比例特征群，类特征群，竞争特征群，业务特征群等多种特征群，对用户及行为进行了细致的刻画。

在原始特征的基础上的一些简单扩充与统计，由于用户的行为过于稀疏，提取用户转化率的时候做了平滑，另外对用户购买点击行为做编码。在 query 交互，用户交互，竞争特征群中，计算量较大，采用并行的方式提取，提升效率。

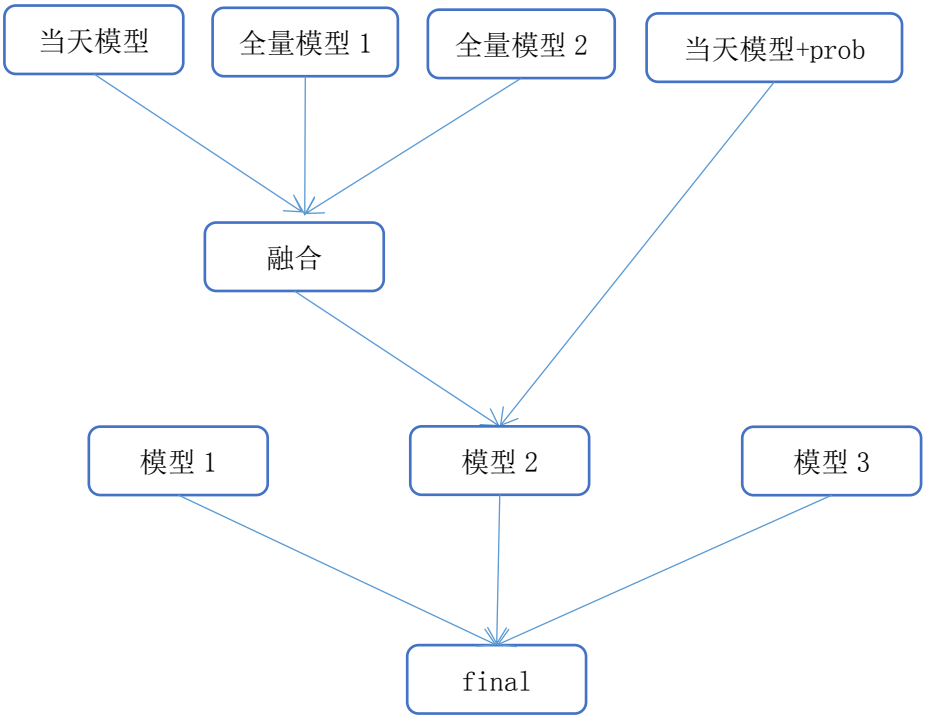


为了提高效率，我们采用分批测试特征群的方式进行线下验证。其中紫色特征群是过拟合的特征群，线下表现突出，在线上表现平平。究其原因是因为，这些特征都是对当天上午的数据进行统计，即使我们使用交叉的方式提取，尽量避免数据穿越，由于上午下午数据分布有差异，所以依然没能很好的克服过拟合。





(7) 模型融合



不同模型好而不同是融合提升的关键，我们队伍队员分别做了三个高分模型，每个模型在样本与特征上均有差异，因此，通过融合进一步提升了成绩。最好的单模型依然能保持 top2 的成绩。

四、 工程优化

为了使模型更具业务实用性，我们对代码进行了优化，主要包括下面四个方面：

1）id 类特征重编码，直接当作特征，树模型深度设置-1，避免了 onehot 大量占用内存空间，实际效果和 onehot 相当。

2）数据并行，主模型 2 小时提取完全部特征。由于 python 的多进程中，子进程会拷贝父进程状态，如果直接把数据分块然后使用多进程会导致内存暴涨，所以我们的解决方案是先将数据分块存为磁盘文件，然后在多进程任务中分别读取各自数据提取特征，最后合并特征，有效的减少了内存占用。

3）数据合并，训练预测数据一起提取特征。直接使用 day,hour 等字段在提取特征完毕之后划分训练，验证，测试集。提高了特征提取，线下测试，线上预测的流程效率。

4)特征分批测试，提高效率。由于复赛数据量比较大，如果使用 warper 类的特征选择方法会浪费大量时间，所以我们直接按特征群分批测试，使用原始特征+测试特征群的方式进行线下验证，少量的特征得以快速迭代验证。

5）并行特征提取的关键代码

数据分块存储

```
def query_data_prepare():
    data=pd.read_csv('../data/origion_concat.csv')
    data=data[data.day>=6]
    data = data.sort_values(by=['user_id', 'context_timestamp']).reset_index(drop=True)
    users = pd.DataFrame(list(set(data['user_id'].values)), columns=['user_id'])
    l_data = len(users)
    size = math.ceil(l_data / processor)
    for i in range(processor):
        start = size * i
        end = (i + 1) * size if (i + 1) * size < l_data else l_data
        user = users[start:end]
        t_data = pd.merge(data, user, on='user_id').reset_index(drop=True)
        t_data.to_csv('../data/user_data/query_'+str(i)+'.csv',index=False)
        print(len(t_data))
```

特征提取

```
def run_query_feature(i):
    data=pd.read_csv('../data/user_data/query_'+str(i)+'.csv')
    features=[]
    for index,row in data.iterrows():
        feature={}
        feature['instance_id']=row['instance_id']
        if index%100==0:
            print(index)
        col=['user_id','predict_category_property','context_timestamp','day','query1','query','item_id'
        tmp=data[data['user_id']==row['user_id']][['instance_id']+col]
        before_query_cnt=len(tmp[(tmp['predict_category_property']==row['predict_category_property'])&
        before_query_1_cnt = len(tmp[(tmp['query1'] == row['query1']) & (tmp['context_timestamp'] < row
```

特征合并

```
def query_feature():
    res = []
    p = Pool(processor)
    for i in range(processor):
        res.append(p.apply_async(run_query_feature, args=( i,)))
        print(str(i) + ' processor started !')
    p.close()
    p.join()
    data=pd.concat([i.get() for i in res])
    data.to_csv('../data/query_all.csv',index=False)
```

五、 比赛经验总结

（1）深刻的赛题理解

对赛题进行认真而理性的分析和全面而深入的思考，对不了解之处做相应的调研。

（2）细致的数据分析

从各个维度对数据进行细致的观察和分析，从中挖掘出重要的规律。

（3）海量的特征

多角度地提取有效特征，构造广阔而高质量的特征海洋，确保没有遗漏有用的信息。

（4）强力的模型

训练多组不同采样方式、不同特征的强力模型，并将它们融合成威

力巨大的终极模型。

（5）未完成的思考

赛题背景是搜索转化预估，可以直接使用的数据是用户已经点击过的数据，实际上我们还可以拿到用户看到过，但是没点击的数据来辅助训练。一个 query 会出多个商品，用户可能只点击了其中一个，如何获取用户看到的其他商品呢？关键还是在 query 上，如果有其他用户用同样的 query 进行了搜索并且点击了不同的商品，那么这个商品可能就是被其他没有点击的用户看到过的。

(6)比赛与实际业务的差距

在本次比赛中，我们使用了大量的特征以及模型融合，其中存在两个需要讨论的问题。首先是特征，我们使用了部分用户当前状态的特征，比如距离上一次点击时间间隔，距离下一次点击时间间隔。第一个特征在实际业务中，需要实时提取，如何设计实时特征的计算框架，性能能否跟上都是需要考虑的问题。第二个特征距离下一次点击时间间隔，这个特征甚至在实际业务中根本提取不到，属于未来的信息，但是在比赛中却可以利用到。如果把预测时间段的用户数据调整为一个用户只出现一次，那么这个问题就可以得到很好的解决。随之而来的另一个问题是，用户只出现一次，就无法统计到用户当前状态的其他未利用到未来信息的特征，像商品店铺的统计信息也不完全，会引起一个信息缺失的问题。所以如何在比赛与实际业务中平衡数据的利用程度是一个需要考虑到的问题。另外一个是模型设计的问题，实际业务中几乎不太可能会用到 stack 之类的模型融合方案，模型复杂度

带来的计算代价和线上预估时间的代价可能会超过模型融合性能提升带来的收益，实际业务简单加权融合可能会成为多数时候的选择。本次比赛我们选择了 LightGBM 模型，因为数据量少，训练快，可以在线下快速迭代。在实际业务中，使用的更多的模型可能是 LR，FFM，DNN 之类的模型，实际业务的数据是海量的，这些模型更能学习到稳定鲁棒的参数，并且预估速度更快。由于正负样本比例悬殊，如果考虑训练效率的话，其实也可以对负样本进行采样后训练，比如 LR 模型训练之后通过对截距项的修正，依然可以保持预估的数据符合实际分布。

六、 团队亮点

（1）强大的阵容

聚集三位优秀的数据挖掘竞赛选手。

（2）一致的目标

队员们对本次比赛的目标一致而明确（虽然最终并未达成）。

（3）良好的沟通

队员们频繁地对赛题进行讨论，及时地同步各自的进展。

（4）完美的配合

队员们分别训练不同的模型，彼此的模型差异极大，特别适合进行融合。