任务五：实战—阿里新手天池赛

目录

[赛题背景： 2](#_Toc520499072)

[赛题任务： 2](#_Toc520499073)

[字段表： 2](#_Toc520499074)

[数据探索： 3](#_Toc520499075)

[数据预处理： 4](#_Toc520499076)

[数据集划分：（见data\_split.py文件） 4](#_Toc520499077)

[特征工程：（见feature\_extract.py文件） 5](#_Toc520499078)

[模型设计：（见xgb.py文件） 7](#_Toc520499079)

[天池信息 8](#_Toc520499080)

**生活大实惠：O2O优惠券使用预测**

# **赛题背景：**

**随着移动设备的完善和普及，移动互联网+各行各业进入了高速发展阶段，这其中以O2O（Online to Offline）消费最为吸引眼球。据不完全统计，O2O行业估值上亿的创业公司至少有10家，也不乏百亿巨头的身影。O2O行业天然关联数亿消费者，各类APP每天记录了超过百亿条用户行为和位置记录，因而成为大数据科研和商业化运营的最佳结合点之一。 以优惠券盘活老用户或吸引新客户进店消费是O2O的一种重要营销方式。然而随机投放的优惠券对多数用户造成无意义的干扰。对商家而言，滥发的优惠券可能降低品牌声誉，同时难以估算营销成本。 个性化投放是提高优惠券核销率的重要技术，它可以让具有一定偏好的消费者得到真正的实惠，同时赋予商家更强的营销能力。本次大赛为参赛选手提供了O2O场景相关的丰富数据，希望参赛选手通过分析建模，精准预测用户是否会在规定时间内使用相应优惠券。**

**赛题任务：**

**通过分析线上和线下的数据分析，预测用户在2016年7月领取优惠券后15天以内的使用情况。**

**提供数据：ccf\_offline\_stage1\_test\_revised.csv，ccf\_offline\_stage1\_train.csv，ccf\_online\_stage1\_train.csv，sample\_submission.csv**

**字段表：**

Table 1: 用户线下消费和优惠券领取行为(ccf\_offline\_stage1\_train.csv)

|  |  |
| --- | --- |
| **Field** | **Description** |
| User\_id | 用户ID |
| Merchant\_id | 商户ID |
| Coupon\_id | 优惠券ID：null表示无优惠券消费，此时Discount\_rate和Date\_received字段无意义 |
| Discount\_rate | 优惠率：x \in [0,1]代表折扣率；x:y表示满x减y。单位是元 |
| Distance | user经常活动的地点离该merchant的最近门店距离是x\*500米（如果是连锁店，则取最近的一家门店），x\in[0,10]；null表示无此信息，0表示低于500米，10表示大于5公里； |
| Date\_received | 领取优惠券日期 |
| Date | 消费日期：如果Date=null & Coupon\_id != null，该记录表示领取优惠券但没有使用，即负样本；如果Date!=null & Coupon\_id = null，则表示普通消费日期；如果Date!=null & Coupon\_id != null，则表示用优惠券消费日期，即正样本； |

Table 2: 用户线上点击/消费和优惠券领取行为(cf\_online\_stage1\_train.csv)

|  |  |
| --- | --- |
| **Field** | **Description** |
| User\_id | 用户ID |
| Merchant\_id | 商户ID |
| Action | 0 点击， 1购买，2领取优惠券 |
| Coupon\_id | 优惠券ID：null表示无优惠券消费，此时Discount\_rate和Date\_received字段无意义。“fixed”表示该交易是限时低价活动。 |
| Discount\_rate | 优惠率：x \in [0,1]代表折扣率；x:y表示满x减y；“fixed”表示低价限时优惠； |
| Date\_received | 领取优惠券日期 |
| Date | 消费日期：如果Date=null & Coupon\_id != null，该记录表示领取优惠券但没有使用；如果Date!=null & Coupon\_id = null，则表示普通消费日期；如果Date!=null & Coupon\_id != null，则表示用优惠券消费日期； |

Table 3：用户O2O线下优惠券使用预测样本(ccf\_offline\_stage1\_test\_revised.csv)

|  |  |
| --- | --- |
| **Field** | **Description** |
| User\_id | 用户ID |
| Merchant\_id | 商户ID |
| Coupon\_id | 优惠券ID：null表示无优惠券消费，此时Discount\_rate和Date\_received字段无意义 |
| Discount\_rate | 优惠率：x \in [0,1]代表折扣率；x:y表示满x减y。单位是元 |
| Distance | user经常活动的地点离该merchant的最近门店距离是x\*500米（如果是连锁店，则取最近的一家门店），x\in[0,10]；null表示无此信息，0表示低于500米，10表示大于5公里； |
| Date\_received | 领取优惠券日期 |

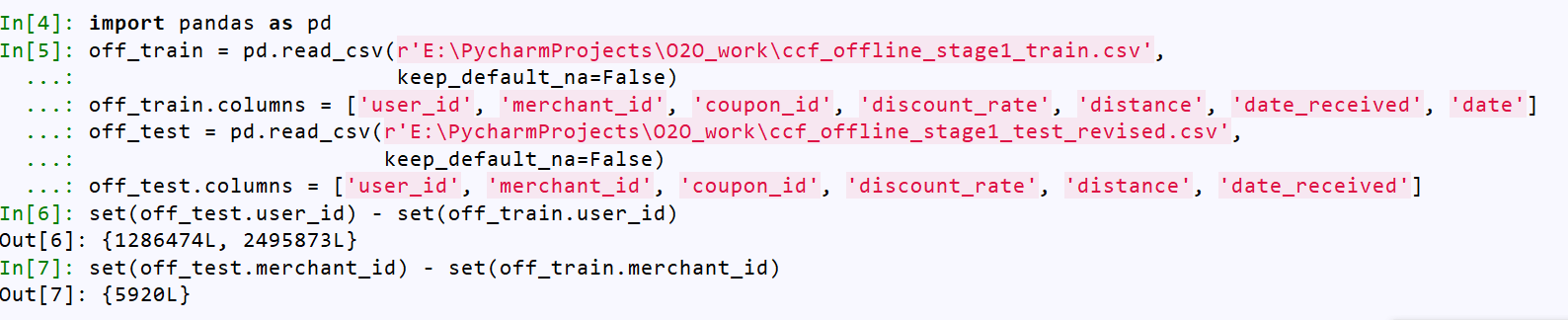
Table 4：选手提交文件字段，其中user\_id,coupon\_id和date\_received均来自Table 3,而Probability为预测值

|  |  |
| --- | --- |
| **Field** | **Description** |
| User\_id | 用户ID |
| Merchant\_id | 商户ID |
| Coupon\_id | 优惠券ID：null表示无优惠券消费，此时Discount\_rate和Date\_received字段无意义 |
| Date\_received | 领取优惠券日期 |
| Probability | 15天内用券概率，由参赛选手给出 |

**数据探索：**

**首先对所给的数据进行预处理分析，对给予的几个基本数据集进行分析。**

**赛题主要给了offtrain，ontrain和offtest三个数据集。通过前两个数据集分析，获得训练模型，然后对offtest数据集中的记录进行预测。然而通过实际分析，发现offtest中的关键属性user\_id和merchant\_id几乎都在offtrain中出现，另一方面ontrain数据集中特有的action属性并存在于offtest数据集中，并且缺乏了distance这个关键属性。理论上也确实最好使用线下的训练集来训练线下消费的模型，对线下消费情况进行分析，在此题中我并没有利用到线上数据集，猜测该数据集应该是在那个比赛的第二赛季起作用。**

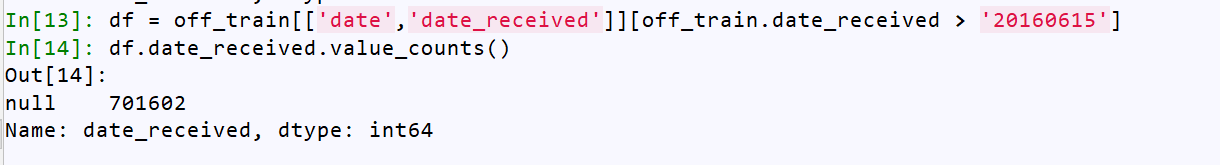


**可见offtest中绝大部分用户和商户都出现在offtrain中，只有一个新商家和两个新用户，故只用offtrain做数据集。**

# **数据预处理：**

**对于offtrain和offtest中都存在不少缺失值，且都以‘null’字符串存在。在read\_csv读取时数据集通过设置keep\_default\_na=False使得缺失以原有的‘null’字符串保留，而不是设置为np.nan。并没有使用其他方式在一开始进行缺失值填充也是后面使用xgboost模型时能自动处理缺失值。这样做导致了一个不好的后果就是在特征处理是先是使用groupby在通过merge左连接不可避免的产生了写nan类型空值，与原有的‘null’冲突，虽然可以通过df.replace()来替换，但是效率还是比较底下，这个问题希望在接下来的工作中解决统一。**

# ****数据集划分：（见data\_split.py文件）****



**观察到offtrain数据集在6.15之后并没有记录任何领券行为。因此在offtrain的特征提取区间取6。15以前。最后决定使用划窗法采用两个训练集dataset1和dataset2，测试集dataset3。如下：**

|  |  |  |
| --- | --- | --- |
|  | 预测区间（提取Label） | 特征区间（提取Feature） |
| 训练集1 | 20160414~20160514 | 20160101~20160413 |
| 训练集2 | 20160515~20160615 | 20160201~20160514 |
| 测试集 | 20160701~20160731 | 20160315~20160630 |

# ****特征工程：（见feature\_extract.py文件）****

**数据集拥有：**'user\_id', 'merchant\_id', 'coupon\_id', 'discount\_rate', 'distance', 'date\_received', 'date'

**七大基本特征。通过这七个基本特征进行交叉组合又可得到新的特征群。如：user\_merchant组合特征可以表示用户对于商家的偏好，进一步细分可得用户在该商家消费次数，使用优惠券等消费次数等等特征。最终决定采用五个特征来划分特征。分别为用户特征，优惠券特征，商家特征，用户—商家特征，其他特征。实际上基本特征之间组合远不止五种，但由于个别特征群特征数量太少，便划分到其他特征上，比如我就把用户—距离特征归属到用户特征群上。总的特征列表如下：**

**用户特征**

|  |  |
| --- | --- |
| **用户特征** | **feature about user** |
| **用户去过的商家数** | **user\_count\_merchants** |
| **用户使用优惠券的最小距离**  **用户使用优惠券的最大距离** | **user\_min\_distance**  **user\_max\_distance** |
| **用户使用优惠券最多的距离** | **user\_most\_distance** |
| **用户使用优惠券的平均距离** | **user\_avg\_distance** |
| **用户总的消费次数** | **buy\_total** |
| **用户使用优惠券的次数** | **buy\_use\_coupon** |
| **用户领取到的优惠券的数量** | **user\_coupon\_received** |
| **用户使用优惠券的时间间隔** | **user\_time\_gap** |
| **用户使用优惠券的平均时间间隔** | **user\_avg\_time\_gap** |
| **用户使用优惠券的最长时间间隔** | **user\_max\_time\_gap** |
| **用户使用优惠券的最短时间间隔** | **user\_min\_time\_gap** |
| **用户使用优惠券的最多的时间间隔** | **user\_most\_time\_gap** |
| **用户核销优惠券的最高折扣率** | **user\_max\_discount\_rate** |
| **用户核销优惠券的最低折扣率** | **user\_min\_discount\_rate** |
| **用户核销优惠券的平均折扣率** | **user\_avg\_discount\_rate** |
| **用户使用优惠券购买的比例** | **user\_buy\_use\_coupon\_rate** |
| **用户领取到的优惠券核销率** | **user\_coupon\_trans\_rate** |
| **用户平均在每个商家的核销优惠券次数** | **user\_avg\_cons\_of\_merchant** |
| **用户平均在每个商家的核销率** | **user\_avg\_coupon\_of\_merchant** |

**商家特征**

|  |  |
| --- | --- |
| **商家特征** | **feature about merchant** |
| **商家总的交易次数** | **total\_sales** |
| **商家的优惠券发行数量** | **total\_coupon** |
| **商家发行的优惠券种类数** | **total\_coupon\_types** |
| **商家优惠券交易的数量** | **coupon\_sales** |
| **商家优惠券交易占比率** | **merchant\_coupon\_sales\_rate** |
| **商家优惠券被使用率** | **merchant\_coupon\_used\_rate** |
| **商家优惠券交易的平均距离** | **merchant\_avg\_distance** |
| **商家优惠券交易的最大距离** | **merchant\_max\_distance** |
| **商家优惠券交易的最小距离** | **merchant\_min\_distance** |
| **商家优惠券交易的众数距离** | **merchant\_most\_distance** |
| **商家发行优惠券的最高折扣率** | **merchant\_max\_discount\_rate** |
| **商家发行优惠券的最低折扣率** | **merchant\_min\_discount\_rate** |
| **商家发行优惠券的平均折扣率** | **merchant\_avg\_discount\_rate** |
| **商家发行优惠券的众数折扣率** | **merchant\_most\_discount\_rate** |

**优惠券特征**

|  |  |
| --- | --- |
| **优惠券特征** | **feature about coupon** |
| **优惠券类型（是否为满减类)** | **is\_man\_jian** |
| **相同折扣率的优惠券发行数量** | **discount\_rate\_total\_nums** |
| **计算相同折扣率的优惠券核销数量** | **discount\_rate\_used\_nums** |
| **满减的满额** | **discount\_man** |
| **满减的减额** | **discount\_jian** |
| **真实折扣率** | **discount\_rate** |
| **优惠券领取时的工作日** | **day\_of\_week** |
| **该优惠券领取时的月份** | **day\_of\_month** |
| **相同id的优惠券的数量** | **coupon\_count** |

**用户-商家特征**

|  |  |
| --- | --- |
| **用户-商家特征** | **feature between user and merchant** |
| **用户在该商家的总消费次数** | **user\_merchant\_buy\_total** |
| **用户在该商家收到的优惠券总数** | **user\_merchant\_received** |
| **用户在该商家购物使用的优惠券数量** | **user\_merchant\_buy\_use\_coupon** |
| **用户在该商家购物不使用的优惠券数量** | **user\_merchant\_buy\_no\_coupon** |
| **用户商家总记录条数** | **user\_merchant\_records** |
| **商家发行优惠券核销率** | **user\_merchant\_coupon\_trans\_rate** |
| **用户在该商家使用优惠券消费比例** | **user\_merchant\_use\_coupon\_rate** |
| **用户在该商家不使用优惠券消费比例** | **user\_merchant\_no\_coupon\_rate** |

**其他特征**

|  |  |
| --- | --- |
| **其他特征** | **other feature** |
| **当天收到的优惠券数量** | **nowday\_receive\_coupon** |
| **收到第一张同样优惠券的时间** | **user\_first\_same\_coupon\_date** |
| **收到最后一张同样优惠券的时间** | **user\_last\_same\_coupon\_date** |
| **优惠券(coupon\_id)是第一张优惠券** | **is\_first** |
| **优惠券(coupon\_id)是最后张优惠券** | **is\_last** |

**对于优惠券特征day\_of\_week使用了get\_dummies()方法进行独热编码（One-Hot encoding）。分解为周一至周日七个属性。特征之间的距离计算变得更加合理。并在此基础上添加了‘is\_weekend’特征。**

**通过对数据集的特征提取，合并**

# **模型设计：**（见xgb.py文件）****

**网上观看论坛以及和同学讨论，得知在单模型下xgboost的效果最好，于是偷懒只使用了xgboost来融合而模型。事实上xgboost对本赛题真的非常契合，效果拔群。其优势有一下几点：xgboost内置处理缺失值规则，解决了本赛题最终数据集中大量缺失值问题。xgboost会一直分裂到指定的最大深度(max\_depth)，然后回过头来剪枝，使得xgboost相对于其他采用贪心算法剪枝的模型效果要好很多。**

**Xgboost参数设置，第一，采用gbtree的booster迭代模式。Pairwise中中Rank排序算法。根据赛题要求对于有效数据的度量方法（eval\_metric）采用auc，曲线下面积来度量。至于其他参数则在推荐范围区间内进行手动微调。参数params最终如下：**

**params={'booster':'gbtree',**

**'objective': 'rank:pairwise',**

**'eval\_metric':'auc',**

**'gamma':0.1,**

**'min\_child\_weight':1,**

**'max\_depth':5,**

**'lambda':10,**

**'subsample':0.7,**

**'colsample\_bytree':0.8,**

**'colsample\_bylevel':0.8,**

**'eta': 0.01,**

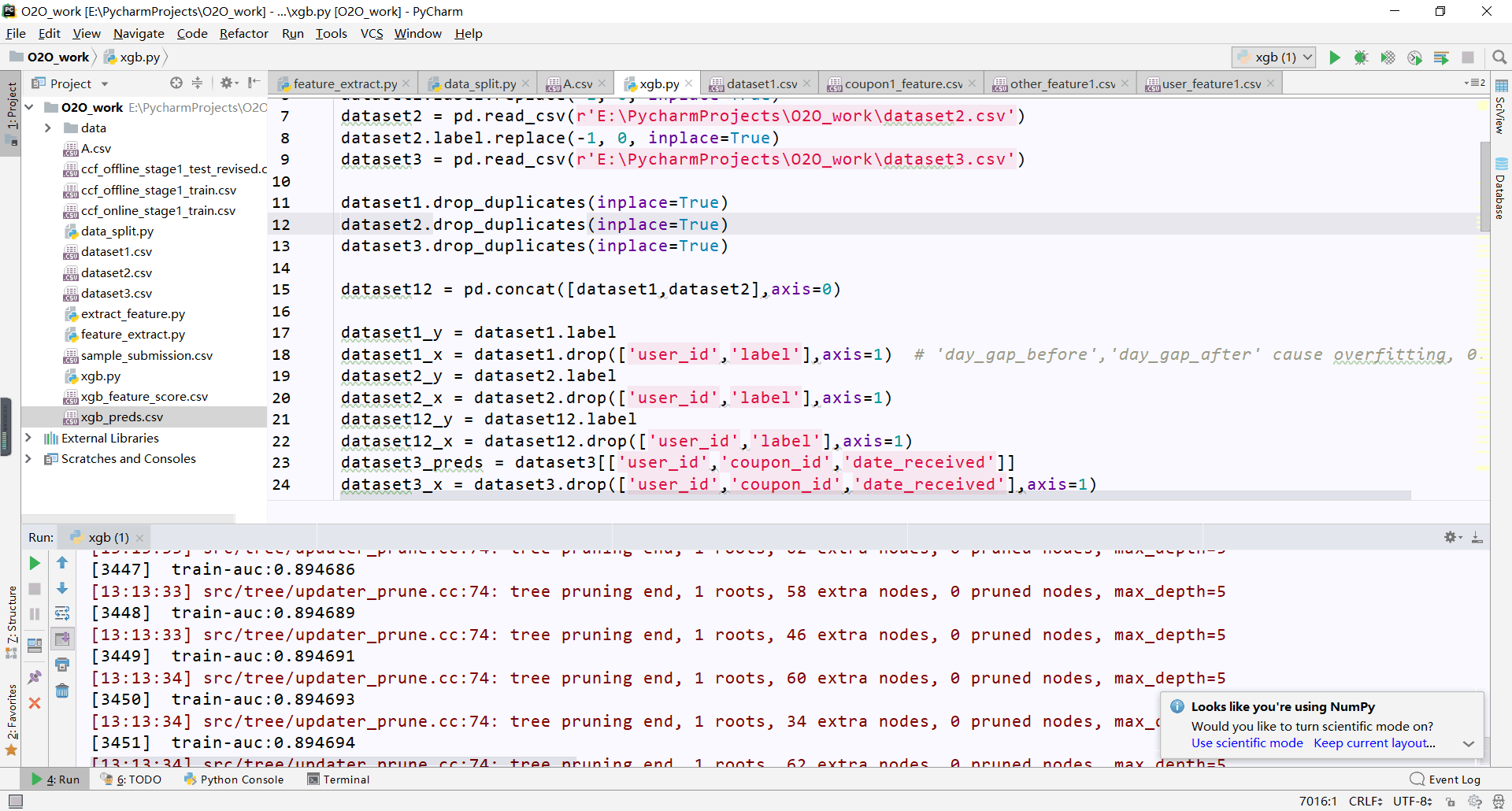
**'tree\_method':'exact',**

**'seed':0,**

**'nthread':12**

**}**

**设置迭代的次数为3500，最终模拟模型auc为0.895左右，线上成绩为0.76564078。**



# 天池信息

最终成绩如下：**天池ID：cauchyguo**



历史提交记录：



