MY RESUME

个人项目介绍

求职意向|算法实习生|机器学习/数据挖掘方向

竞聘人：夏杨

目录

CONTENTS

01 天池大赛：O2O优惠券使用预测

01 天池大赛：O2O优惠券使用预测

（排名：前1%，AUC：0.7948）（Top1：0.8116）

问题描述

数据

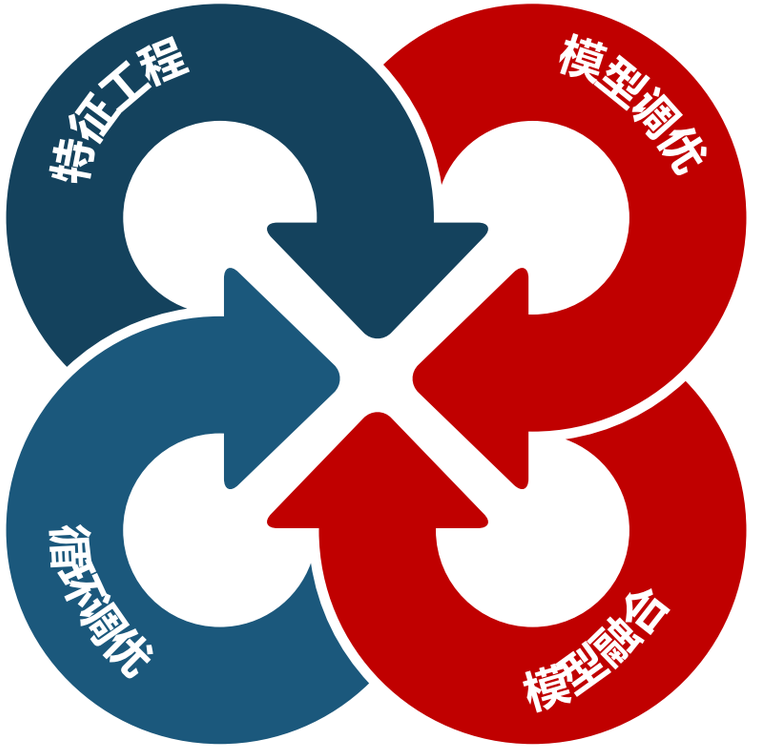
本赛题提供用户在2016年1月1日至2016年6月30日之间真实线上线下消费行为，预测用户在2016年7月领取优惠券后15天以内的使用情况。

注意： 为了保护用户和商家的隐私，所有数据均作匿名处理，同时采用了有偏采样和必要过滤。

评价方式

本赛题目标是预测投放的优惠券是否核销。针对此任务及一些相关背景知识，使用优惠券核销预测的平均AUC（ROC曲线下面积）作为评价标准。 即对每个优惠券coupon_id单独计算核销预测的AUC值，再对所有优惠券的AUC值求平均作为最终的评价标准。

主要工作



特征工程

字段表

Table 1: 用户线下消费和优惠券领取行为

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O2O线下优惠券使用预测样本

Field	Description
User_id	用户ID
Merchant_id	商户ID
Coupon_id	优惠券ID
Discount_rate	优惠率：x \in [0,1]代表折扣率；x:y表示满x减y.
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

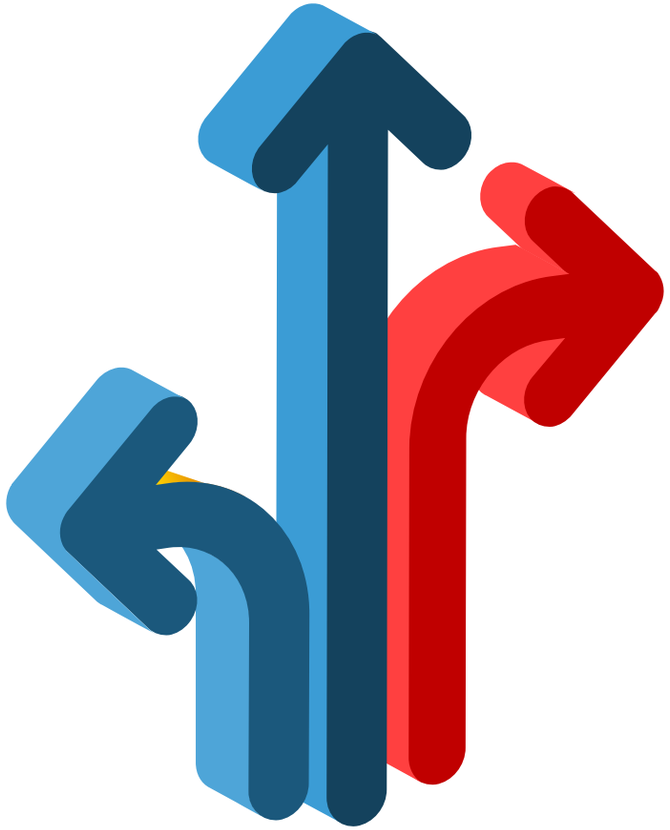
原始数据特点：

样本数约为175万，原始数据特征很少

特征工程

思路二

使用深度神经网络提取特征



思路三

人工特征提取

思路一

直接使用原始特征进行建模

特征工程

1. 用户相关的特征
2. 商家相关的特征
3. 优惠券相关的特征
4. 用户-商家交互特征
5. 其它特征

特征工程

用户相关的特征

用户领取优惠券次数

用户获得优惠券但没有消费的次数

用户获得优惠券并核销次数

用户领取优惠券后进行核销率

用户满050/50200/200~500 减的优惠券核销率

用户核销满050/50200/200~500减的优惠券占所有核销优惠券的比重

用户核销优惠券的平均/最低/最高消费折率

用户核销过优惠券的不同商家数量，及其占所有不同商家的比重

用户核销过的不同优惠券数量，及其占所有不同优惠券的比重

用户平均核销每个商家多少张优惠券

用户核销优惠券中的平均/最大/最小用户-商家距离

商家相关的特征

商家优惠券被领取次数

商家优惠券被领取后不核销次数

商家优惠券被领取后核销次数

商家优惠券被领取后核销率

商家优惠券核销的平均/最小/最大消费折率

核销商家优惠券的不同用户数量，及其占领取不同的用户比重

商家优惠券平均每个用户核销多少张

商家被核销过的不同优惠券数量

商家被核销过的不同优惠券数量占所有领取过的不同优惠券数量的比重

商家平均每种优惠券核销多少张

商家被核销优惠券的平均时间率

商家被核销优惠券中的平均/最小/最大用户-商家距离

用户-商家交互特征

用户领取商家的优惠券次数

用户领取商家的优惠券后不核销次数

用户领取商家的优惠券后核销次数

用户领取商家的优惠券后核销率

用户对每个商家的不核销次数占用户总的不核销次数的比重

用户对每个商家的优惠券核销次数占用户总的核销次数的比重

用户对每个商家的不核销次数占商家总的不核销次数的比重

用户对每个商家的优惠券核销次数占商家总的核销次数的比重

特征工程

优惠券相关的特征

优惠券类型(直接优惠为0, 满减为1)

优惠券折率

满减优惠券的最低消费

历史出现次数

历史核销次数

历史核销率

历史核销时间率

领取优惠券是一周的第几天

领取优惠券是一月的第几天

历史上用户领取该优惠券次数

历史上用户消费该优惠券次数

历史上用户对该优惠券的核销率

其它特征

用户领取的所有优惠券数目

用户领取的特定优惠券数目

用户此次之后/前领取的所有优惠券数目

用户此次之后/前领取的特定优惠券数目

用户上/下一次领取的时间间隔

用户领取特定商家的优惠券数目

用户领取的不同商家数目

用户当天领取的优惠券数目

用户当天领取的特定优惠券数目

用户领取的所有优惠券种类数目

商家被领取的优惠券数目

商家被领取的特定优惠券数目

特征工程

这里会出现一个问题：

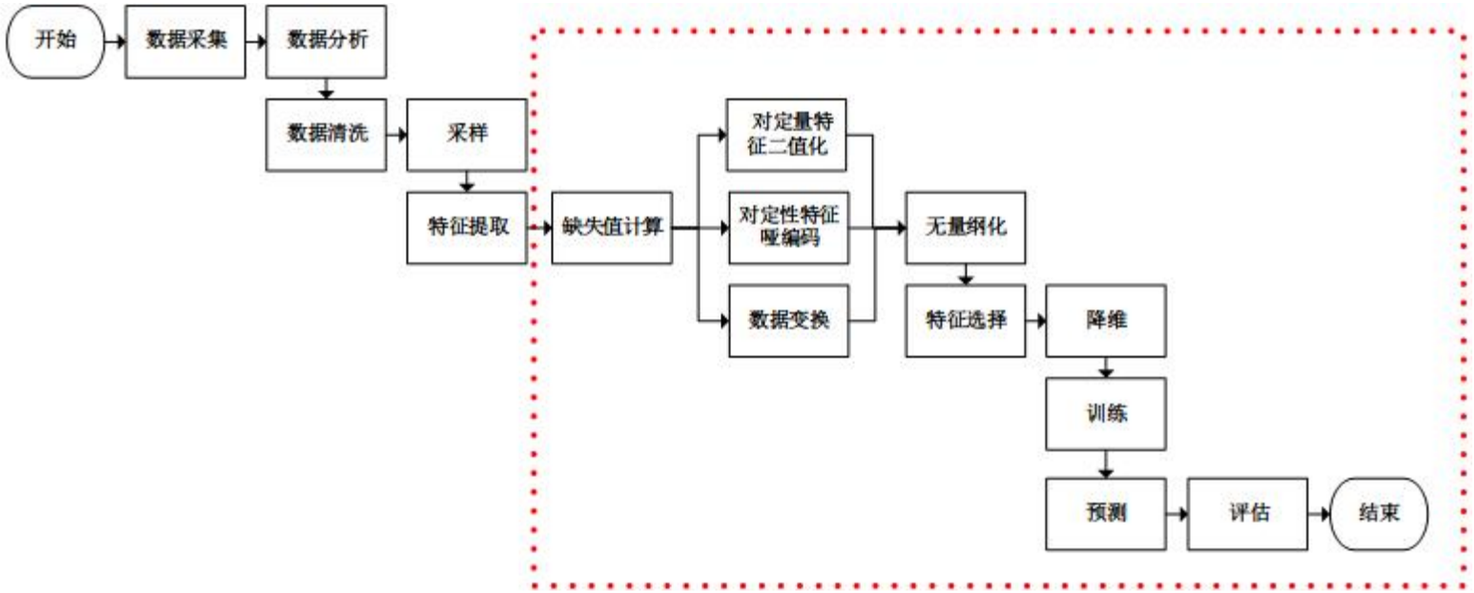
特征提取中很多特征需要消费日期这一个字段，对于训练集好说，但是对于预测集就是让你预测消费日期，所以不可能有这个字段，所以对于测试集很多特征提取不出来。

采用SW的方式对数据集进行划分：

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

特征工程

接下来对提取出来的特征进一步处理：



本项目采用树模型进行特征选择：

xgboost：get_fscore (以weight为评价方式)
weight - 该特征在所有树中被用作分割样本的特征的次数。

特征工程

最后得到一个52维特征：

[(0, 'discount_rate'),
(1, 'distance'),
(2, 'day_of_month'),
(3, 'days_distance'),
(4, 'discount_man'),
(5, 'discount_jian'),
(6, 'is_man_jian'),
(7, 'total_sales'),
(8, 'sales_use_coupon'),
(9, 'total_coupon'),
(10, 'merchant_min_distance'),
(11, 'merchant_max_distance'),
(12, 'merchant_mean_distance'),
(13, 'merchant_median_distance'),
(14, 'merchant_coupon_transfer_rate'),
(15, 'coupon_rate'),
(16, 'count_merchant'),
(17, ‘user_min_distance’),
(18, 'user_max_distance'),
(19, 'user_mean_distance'),
(20, 'user_median_distance'),
(21, 'buy_use_coupon'),
(22, 'buy_total'),
(23, 'coupon_received'),
(24, 'avg_user_date_datereceived_gap'),
(25, 'min_user_date_datereceived_gap'),
(26, 'max_user_date_datereceived_gap'),
(27, 'buy_use_coupon_rate'),
(28, 'user_coupon_transfer_rate'),
(29, 'user_merchant_buy_total'),
(30, 'user_merchant_received'),
(31, 'user_merchant_buy_use_coupon'),
(32, 'user_merchant_any'),
(33, 'user_merchant_buy_common'),
(34, 'user_merchant_coupon_transfer_rate'),
(35, 'user_merchant_coupon_buy_rate'),
(36, 'user_merchant_rate'),
(37, 'user_merchant_common_buy_rate'),
(38,'this_month_user_receive_same_coupn_count'),
(39,'this_month_user_receive_all_coupon_count'),
(40,'this_month_user_receive_same_coupon_lastone'),
(41,'this_month_user_receive_same_coupon_firstone'),
(42, 'this_day_receive_all_coupon_count'),
(43,'this_day_user_receive_same_coupon_count'),
(44, 'is_weekend'),
(45, 'weekday1'),
(46, 'weekday2'),
(47, 'weekday3'),
(48, 'weekday4'),
(49, 'weekday5'),
(50, 'weekday6'),
(51, 'weekday7')]

特征工程

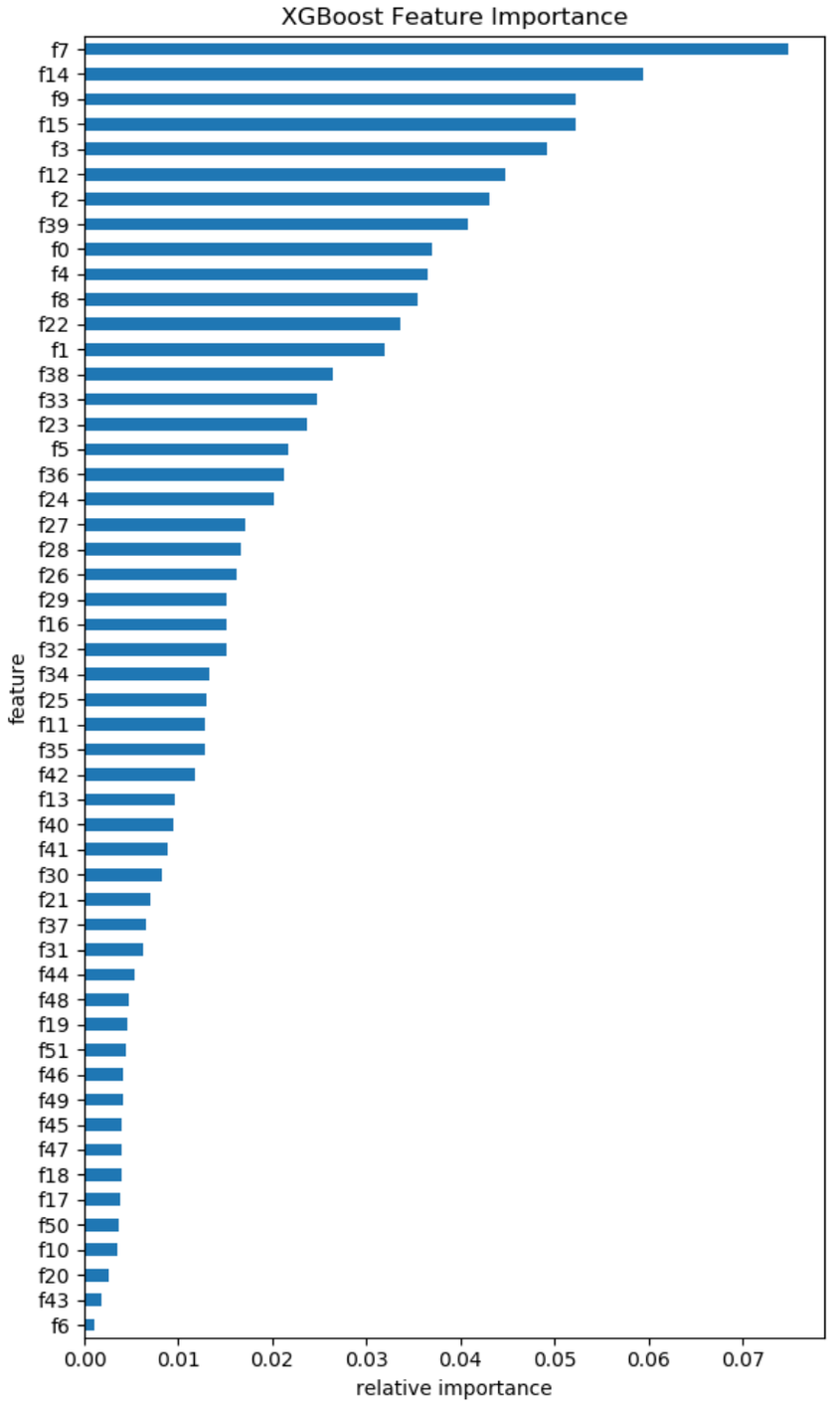
最后得到一个52维特征：

重要特征：

(7, 'total_sales'),
(9, 'total_coupon'),
(14, 'merchant_coupon_transfer_rate'),
(15, 'coupon_rate')

不重要特征：

(6, 'is_man_jian'),
(17, 'user_min_distance'), （可能是因为有较多缺失值）
(18, 'user_max_distance'),
(43, 'this_day_user_receive_same_coupon_count'),
(50, 'weekday6'),



模型调优

最终处理后的训练集为39万个样本，52维特征：

备选的模型有XGBoost、GBDT、SVM、LR、RF

首先SVM和LR淘汰:

SVM解决小样本情况下的机器学习问题上有很大的优势，而这是一个大样本问题;
LR处理非线性特征效果不好

模型调优

对比GBDT和XGBoost:

与GBDT相比，xgBoosting有以下进步：

我自己理解大概三个方面：

第一、代价函数方面
第二、防止过拟合方面
第三、独有的trick（近似算法；近似直方图算法；支持并行；缺失值处理）

模型调优

XGBoost模型调参：

我理解这个参数调优的顺序就是从底层到上层

第一步：确定学习速率和基学习器数目（使用XGB.CV）

学习速率的值为0.1；其他参数按默认值初始化

第二步：max_depth 和 min_child_weight 参数调优（以下都是使用GridSearchCV）

第三步：gamma参数调优

Gamma指定了节点分裂所需的最小损失函数下降值。 这个参数的值越大，算法越保守。

第四步：调整subsample 和 colsample_bytree 参数

第五步：正则化参数调优

第六步：降低学习速率

最后，我们使用较低的学习速率，以及使用更多的决策树。我们可以用XGBoost中的CV函数来进行这

```
params = {'booster': 'gbtree',
          'objective': 'binary:logistic',
          'eval_metric': 'auc',
          'gamma': 0.1,
          'min_child_weight': 1.1,
          'max_depth': 5,
          'lambda': 10,
          'subsample': 0.7,
          'colsample_bytree': 0.7,
          'colsample_bylevel': 0.7,
          'eta': 0.01,
          'tree_method': 'exact',
          'seed':0
          }
```

模型融合

Stacking

训练了XGBoost，GBDT，RF三种单模型，其中XGBoost表现最好，GBDT次之，RF相比之下最差；

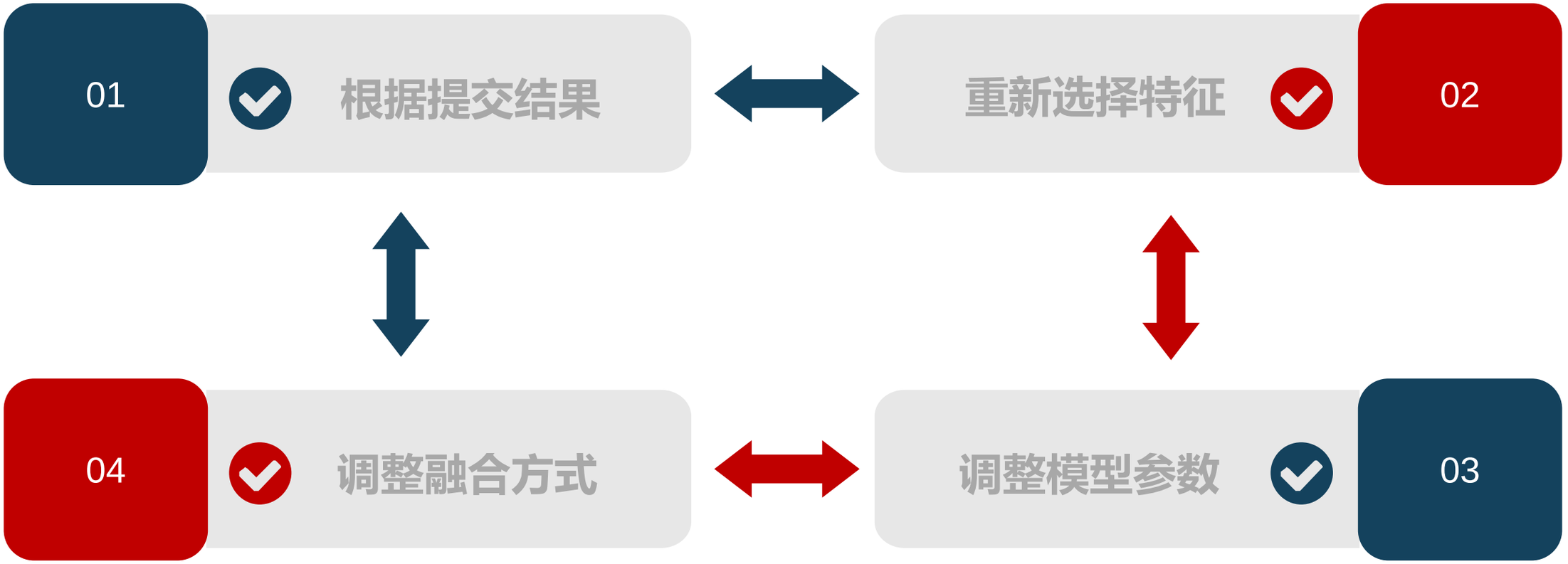
尝试两层stacking模型，level1 在D1上训练了2个XGBoost，2个GBDT，将这些模型的预测结果作为level2的feature，在D2上训练第二层模型。stacking模型的结果相比单模型有细微的提升，但这点提升相对于模型复杂度带来的计算代价显得微不足道。

Average

2个XGBoost（1个过拟合、1个欠拟合）、2个GBDT（1个过拟合、1个欠拟合）加权average；

提升0.02左右

循环调优



总结

比赛总结：

特征工程决定上限，模型只是接近这个上限；

实际工程中，大部分的时间花在了去发现自己有多傻；

熟悉数据挖掘各个环节的流程；

特征处理还不够完善，尝试优化特征的提取和选择；

模型优化还有有待完善，由于计算机的性能限制，很多参数搜索跑的很粗糙，模型融合上很多更复杂的融合也没有跑完

与第一名差距

比赛总结：

特征工程方面：

特征工程决定上限，模型只是接近这个上限；特征中可能会有一些magic特征（尤其权重比较大的特征）我没有提取出来，毕竟大家都是采用人工方式提取特征，这个很取决于工程经验；

模型融合方面：

我模型融合的过程比较简单，只使用了一个加权average和简单的两层stacking；实际上应该尝试更加复杂融合方式，但是融合方式稍加复杂对于计算量就是指数倍的增加。例如选择一个4模型10-fold stacking，那么第一层我就需要训练40个模型，我当时的电脑性能不够。

MY RESUME

THANK YOU

个人简历 | 竞聘求职 | 应聘简历

竞聘人：小派派