Python大作业报告

黄杨峻　17341059

陈飞佚　17341011

刘学海　17341111

选题及题目要求

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。

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。 关于AUC的含义与具体计算方法。

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: 用户线上点击/消费和优惠券领取行为

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

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

- 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
Coupon_id	优惠券ID
Date_received	领取优惠券日期
Probability	15天内用券概率，由参赛选手给出

解题思路和代码分析

解决思路

提供数据的区间是2016-01-01~2016-06-30，预测七月份用户领券使用情况，即用或者不用，我们将此问题转化为二分类问题，然后通过分类算法预测结果。首先就是特征工程，其中涉及对数据集合的划分，包括提取特征的区间和训练数据区间。接着就是从特征区间中提取特征，包括用户特征、商户特征、优惠券特征、用户商户组合特征、用户优惠券组合特征。最后，使用分类算法：GBDT , XGBOOST , RandomForest, LightBGM 对提取的特征进行训练。此外，我们研究了使用模型融合的不同方法：如加权融合、rank sort等方法对不同的分类算法进行模型融合，以观察是否能对模型的auc准确度造成提升。

数据集划分：

比赛主要提供了两份数据集：

Online数据集提供了用户相关的特征。

Offline数据集提供了用户相关的特征、商家相关的特征、优惠券相关的特征与用户——商家交互的特征。

我们采用了如下的划分方式：将数据集划分成多份训练数据集。

	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

特征训练集的划分可以增加训练样本，同时也方便了我们的调参实验。

特征提取

下面我们将展示本次比赛中我们使用到的特征。

- 用户线上相关的特征（取自online数据集）
  - 用户线上操作次数
  - 用户线上点击率
  - 用户线上购买率
  - 用户线上领取率
  - 用户线上不消费次数
  - 用户线上优惠券核销次数
  - 用户线上优惠券核销率
  - 用户线下不消费次数占线上线下总的不消费次数的比重
  - 用户线下的优惠券核销次数占线上线下总的优惠券核销次数的比重
  - 用户线下领取的记录数量占总的记录数量的比重

- 用户线下相关的特征（取自offline数据集）
  - 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0~50 / 50 ~ 200 / 200~500 减的优惠券核销率
  - 用户核销满0~50/50~200/200~500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（取自offline数据集）
  - 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-商家的交互特征：
  - 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

- 优惠券相关特征
  - 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

- 其他特征
  - 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
  - 商家被多少不同用户领取的数目
  - 商家发行的所有优惠券种类数目

- 用户特征的提取

下面我们举dataset3的用户特征提取为例，展示提取特征的过程。 使用官方提供的offline数据集提供的'user_id','merchant_id','coupon_id','discount_rate','distance','date_received','date' 的特征进行提取，得到的特征如下：

user_id	count_merchant	user_min_distance	user_max_distance	user_mean_distance	user_median_distance	buy_use_coupon	buy_total	coupon_received	avg_user_date_datereceived_gap	min_user_date_datereceived_gap	max_user_date_datereceived_gap	buy_use_coupon_rate	user_co

```
#for dataset3
user3 = feature3[['user_id','merchant_id','coupon_id','discount_rate','distance','date_received','date']]

t = user3[['user_id']]
t.drop_duplicates(inplace=True)

t1 = user3[user3.date!='null'][['user_id','merchant_id']]
t1.drop_duplicates(inplace=True)
t1.merchant_id = 1
t1 = t1.groupby('user_id').agg('sum').reset_index()
t1.rename(columns={'merchant_id':'count_merchant'},inplace=True)

t2 = user3[(user3.date!='null')&(user3.coupon_id!='null')][['user_id','distance']]
t2.replace('null',-1,inplace=True)
t2.distance = t2.distance.astype('int')
t2.replace(-1,np.nan,inplace=True)
t3 = t2.groupby('user_id').agg('min').reset_index()
t3.rename(columns={'distance':'user_min_distance'},inplace=True)

t4 = t2.groupby('user_id').agg('max').reset_index()
t4.rename(columns={'distance':'user_max_distance'},inplace=True)

t5 = t2.groupby('user_id').agg('mean').reset_index()
t5.rename(columns={'distance':'user_mean_distance'},inplace=True)

t6 = t2.groupby('user_id').agg('median').reset_index()
t6.rename(columns={'distance':'user_median_distance'},inplace=True)

t7 = user3[(user3.date!='null')&(user3.coupon_id!='null')][['user_id']]
t7['buy_use_coupon'] = 1
t7 = t7.groupby('user_id').agg('sum').reset_index()
```

```
t8 = user3[user3.date!='null'][['user_id']]
t8['buy_total'] = 1
t8 = t8.groupby('user_id').agg('sum').reset_index()

t9 = user3[user3.coupon_id!='null'][['user_id']]
t9['coupon_received'] = 1
t9 = t9.groupby('user_id').agg('sum').reset_index()

t10 = user3[(user3.date_received!='null')&(user3.date!='null')][['user_id','date_received','date']]
t10['user_date_datereceived_gap'] = t10.date + ':' + t10.date_received
t10.user_date_datereceived_gap = t10.user_date_datereceived_gap.apply(get_user_date_datereceived_gap)
t10 = t10[['user_id','user_date_datereceived_gap']]

t11 = t10.groupby('user_id').agg('mean').reset_index()
t11.rename(columns={'user_date_datereceived_gap':'avg_user_date_datereceived_gap'},inplace=True)
t12 = t10.groupby('user_id').agg('min').reset_index()
t12.rename(columns={'user_date_datereceived_gap':'min_user_date_datereceived_gap'},inplace=True)
t13 = t10.groupby('user_id').agg('max').reset_index()
t13.rename(columns={'user_date_datereceived_gap':'max_user_date_datereceived_gap'},inplace=True)


user3_feature = pd.merge(t,t1,on='user_id',how='left')
user3_feature = pd.merge(user3_feature,t3,on='user_id',how='left')
user3_feature = pd.merge(user3_feature,t4,on='user_id',how='left')
user3_feature = pd.merge(user3_feature,t5,on='user_id',how='left')
user3_feature = pd.merge(user3_feature,t6,on='user_id',how='left')
user3_feature = pd.merge(user3_feature,t7,on='user_id',how='left')
user3_feature = pd.merge(user3_feature,t8,on='user_id',how='left')
user3_feature = pd.merge(user3_feature,t9,on='user_id',how='left')
user3_feature = pd.merge(user3_feature,t11,on='user_id',how='left')
user3_feature = pd.merge(user3_feature,t12,on='user_id',how='left')
user3_feature = pd.merge(user3_feature,t13,on='user_id',how='left')
user3_feature.count_merchant = user3_feature.count_merchant.replace(np.nan,0)
user3_feature.buy_use_coupon = user3_feature.buy_use_coupon.replace(np.nan,0)
user3_feature['buy_use_coupon_rate'] = user3_feature.buy_use_coupon.astype('float') /
user3_feature.buy_total.astype('float')
user3_feature['user_coupon_transfer_rate'] = user3_feature.buy_use_coupon.astype('float') /
user3_feature.coupon_received.astype('float')
user3_feature.buy_total = user3_feature.buy_total.replace(np.nan,0)
user3_feature.coupon_received = user3_feature.coupon_received.replace(np.nan,0)
user3_feature.to_csv('data/user3_feature.csv',index=None)
```

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	user_id	count_mer	user_min	user_max	user_mean	user_medi	buy_use_c	buy_total	coupon_re	avg_user	min_user	max_user	buy_use_c	user_coupon_transfer_rate		
2	1439408	1	0	0	0	0	1	2	4	28	28	28	0.5	0.25		
3	1832624	0					0	0	1					0		
4	2029232	1					0	2	2				0	0		
5	2747744	0					0	0	1					0		
6	196342	1					0	1	1				0	0		
7	163606	0					0	0	1					0		
8	94107	0					0	0	1					0		
9	253750	1					0	1	2				0	0		
10	343660	0					0	0	1					0		
11	4191584	1					0	2	0				0			
12	1113008	1	0	0	0	0	1	2	4	6	6	6	0.5	0.25		
13	2881376	1	0	0	0	0	1	2	1	8	8	8	0.5	1		
14	4061024	1					0	1	2				0	0		
15	7073472	1					0	1	3				0	0		

- 商户特征的提取：

```
merchant1 = feature1[['merchant_id','coupon_id','distance','date_received','date']]

t = merchant1[['merchant_id']]
t.drop_duplicates(inplace=True)

t1 = merchant1[merchant1.date!='null'][['merchant_id']]
t1['total_sales'] = 1
t1 = t1.groupby('merchant_id').agg('sum').reset_index()

t2 = merchant1[(merchant1.date!='null')&(merchant1.coupon_id!='null')][['merchant_id']]
t2['sales_use_coupon'] = 1
t2 = t2.groupby('merchant_id').agg('sum').reset_index()

t3 = merchant1[merchant1.coupon_id!='null'][['merchant_id']]
t3['total_coupon'] = 1
t3 = t3.groupby('merchant_id').agg('sum').reset_index()

t4 = merchant1[(merchant1.date!='null')&(merchant1.coupon_id!='null')][['merchant_id','distance']]
t4.replace('null',-1,inplace=True)
t4.distance = t4.distance.astype('int')
t4.replace(-1,np.nan,inplace=True)
```

```
t5 = t4.groupby('merchant_id').agg('min').reset_index()
t5.rename(columns={'distance':'merchant_min_distance'},inplace=True)

t6 = t4.groupby('merchant_id').agg('max').reset_index()
t6.rename(columns={'distance':'merchant_max_distance'},inplace=True)

t7 = t4.groupby('merchant_id').agg('mean').reset_index()
t7.rename(columns={'distance':'merchant_mean_distance'},inplace=True)

t8 = t4.groupby('merchant_id').agg('median').reset_index()
t8.rename(columns={'distance':'merchant_median_distance'},inplace=True)


merchant1_feature = pd.merge(t,t1,on='merchant_id',how='left')
merchant1_feature = pd.merge(merchant1_feature,t2,on='merchant_id',how='left')
merchant1_feature = pd.merge(merchant1_feature,t3,on='merchant_id',how='left')
merchant1_feature = pd.merge(merchant1_feature,t5,on='merchant_id',how='left')
merchant1_feature = pd.merge(merchant1_feature,t6,on='merchant_id',how='left')
merchant1_feature = pd.merge(merchant1_feature,t7,on='merchant_id',how='left')
merchant1_feature = pd.merge(merchant1_feature,t8,on='merchant_id',how='left')
merchant1_feature.sales_use_coupon = merchant1_feature.sales_use_coupon.replace(np.nan,0) #fillna with 0
merchant1_feature['merchant_coupon_transfer_rate'] = merchant1_feature.sales_use_coupon.astype('float') /
merchant1_feature.total_coupon
merchant1_feature['coupon_rate'] = merchant1_feature.sales_use_coupon.astype('float') /
merchant1_feature.total_sales
merchant1_feature.total_coupon = merchant1_feature.total_coupon.replace(np.nan,0) #fillna with 0
merchant1_feature.to_csv('data/merchant1_feature.csv',index=None)
```

- 优惠券特征的提取

  声明函数：

```
def calc_discount_rate(s):
    s =str(s)
    s = s.split(':')
    if len(s)==1:
        return float(s[0])
    else:
        return 1.0-float(s[1])/float(s[0])
def get_discount_man(s):
    s =str(s)
    s = s.split(':')
    if len(s)==1:
        return 'null'
    else:
        return int(s[0])
def get_discount_jian(s):
    s =str(s)
    s = s.split(':')
    if len(s)==1:
        return 'null'
    else:
        return int(s[1])
 def is_man_jian(s):
    s =str(s)
    s = s.split(':')
    if len(s)==1:
        return 0
    else:
        return 1
```

  优惠券特征的提取部分：

```
dataset1['day_of_week'] = dataset1.date_received.astype('str').apply(lambda
x:date(int(x[0:4]),int(x[4:6]),int(x[6:8])).weekday()+1)
dataset1['day_of_month'] = dataset1.date_received.astype('str').apply(lambda x:int(x[6:8]))
dataset1['days_distance'] = dataset1.date_received.astype('str').apply(lambda x:
(date(int(x[0:4]),int(x[4:6]),int(x[6:8]))-date(2016,4,13)).days)
dataset1['discount_man'] = dataset1.discount_rate.apply(get_discount_man)
dataset1['discount_jian'] = dataset1.discount_rate.apply(get_discount_jian)
dataset1['is_man_jian'] = dataset1.discount_rate.apply(is_man_jian)
dataset1['discount_rate'] = dataset1.discount_rate.apply(calc_discount_rate)
d = dataset1[['coupon_id']]
d['coupon_count'] = 1
d = d.groupby('coupon_id').agg('sum').reset_index()
dataset1 = pd.merge(dataset1,d,on='coupon_id',how='left')
dataset1.to_csv('data/coupon1_feature.csv',index=None)
```

- 用户-商户交互特征的提取

```
all_user_merchant = feature3[['user_id','merchant_id']]all_user_merchant.drop_duplicates(inplace=True)t =
feature3[['user_id','merchant_id','date']]t = t[t.date!='null']
[['user_id','merchant_id']]t['user_merchant_buy_total'] = 1t =
t.groupby(['user_id','merchant_id']).agg('sum').reset_index()t.drop_duplicates(inplace=True)t1 =
feature3[['user_id','merchant_id','coupon_id']]t1 = t1[t1.coupon_id!='null']
[['user_id','merchant_id']]t1['user_merchant_received'] = 1t1 =
t1.groupby(['user_id','merchant_id']).agg('sum').reset_index()t1.drop_duplicates(inplace=True)t2 =
feature3[['user_id','merchant_id','date','date_received']]t2 = t2[(t2.date!='null')&
(t2.date_received!='null')][['user_id','merchant_id']]t2['user_merchant_buy_use_coupon'] = 1t2 =
t2.groupby(['user_id','merchant_id']).agg('sum').reset_index()t2.drop_duplicates(inplace=True)t3 =
feature3[['user_id','merchant_id']]t3['user_merchant_any'] = 1t3 =
t3.groupby(['user_id','merchant_id']).agg('sum').reset_index()t3.drop_duplicates(inplace=True)t4 =
feature3[['user_id','merchant_id','date','coupon_id']]t4 = t4[(t4.date!='null')&(t4.coupon_id=='null')]
[['user_id','merchant_id']]t4['user_merchant_buy_common'] = 1t4 =
t4.groupby(['user_id','merchant_id']).agg('sum').reset_index()t4.drop_duplicates(inplace=True)user_mercha
nt3 = pd.merge(all_user_merchant,t,on=['user_id','merchant_id'],how='left')user_merchant3 =
pd.merge(user_merchant3,t1,on=['user_id','merchant_id'],how='left')user_merchant3 =
pd.merge(user_merchant3,t2,on=['user_id','merchant_id'],how='left')user_merchant3 =
pd.merge(user_merchant3,t3,on=['user_id','merchant_id'],how='left')user_merchant3 =
pd.merge(user_merchant3,t4,on=
['user_id','merchant_id'],how='left')user_merchant3.user_merchant_buy_use_coupon =
user_merchant3.user_merchant_buy_use_coupon.replace(np.nan,0)user_merchant3.user_merchant_buy_common =
user_merchant3.user_merchant_buy_common.replace(np.nan,0)user_merchant3['user_merchant_coupon_transfer_ra
te'] = user_merchant3.user_merchant_buy_use_coupon.astype('float') /
user_merchant3.user_merchant_received.astype('float')user_merchant3['user_merchant_coupon_buy_rate'] =
user_merchant3.user_merchant_buy_use_coupon.astype('float') /
user_merchant3.user_merchant_buy_total.astype('float')user_merchant3['user_merchant_rate'] =
user_merchant3.user_merchant_buy_total.astype('float') /
user_merchant3.user_merchant_any.astype('float')user_merchant3['user_merchant_common_buy_rate'] =
user_merchant3.user_merchant_buy_common.astype('float') /
user_merchant3.user_merchant_buy_total.astype('float')user_merchant3.to_csv('data/user_merchant3.csv',ind
ex=None)
```

算法

RandomForest

算法思路

每次随机选取部分样本和部分特征维度进行节点分裂，通过生成多棵树来拟合样本。

模型特点

与别的算法相比，随机森林最大的特点是它每次分裂节点都对特征进行采样，选取子集来进行拟合。适用于当多种特征相关性较大的情况。

Python实现

```
dataset12 = pd.concat([dataset1,dataset2],axis=0)
scalar=preprocessing.StandardScaler()
features=scalar.fit_transform(dataset12_x)
clf=RandomForestClassifier(n_estimators=100,oob_score=True)
features=numpy.nan_to_num(features)
clf.fit(features,dataset12_y)
```

randomforestclassifier 的fit参数：

max_features:每次分裂采样时选取的最大特征维数

n_estimators:弱学习器最大迭代次数

max_depth:最大决策树深度

参数

使用默认参数

GBDT

算法思路

产生特征组合后，通过逻辑回归生成弱分类器Cart_tree，最后通过多个弱分类器进行残差拟合生成强分类器。

模型特点

GBDT主要通过降低偏差来实现拟合，并且各个树之间是串行生成的，必须用后一棵树去拟合前一棵树。因此速度较慢。另一方面与随机森林相比，GBDT对异常值更加敏感。

Python实现

```
scalar = preprocessing.StandardScaler()
features = scalar.fit_transform(dataset12_x)
features = numpy.nan_to_num(features)


clf = ensemble.GradientBoostingClassifier(max_depth =6, learning_rate =0.05,n_estimators=200 )
clf.fit(features, dataset12_y)  # Training model
dataset3_x = scalar.fit_transform(dataset3_x)
dataset3_x = numpy.nan_to_num(dataset3_x)
temp = clf.predict_proba(dataset3_x)[:,1]
```

GradientBoostingClassifier：

n_estimator:弱学习器最大迭代次数

learning_rate:每个弱学习器权重缩减系数

调参结果

max_depth =6, learning_rate =0.05,n_estimators=200

Lightgbm 优化

优化目的

lightgbm对内存和速度进行了优化，但这同样会带来特征离散化后找到的分割点不是最优分割点的问题。

Python实现

lgb.train()参数：

boostint_type:训练方式

objective:目标

learning_rate:学习率

调参结果

```
params = {
    'task': 'train',
    'boosting_type': 'gbdt',
    'objective': 'regression',
    'metric': {'l2', 'auc'},
    'num_leaves': 31,
    'learning_rate': 0.05,
    'feature_fraction': 0.9,
    'bagging_fraction': 0.8,
    'bagging_freq': 5,
    'verbose': 0
}
```

XGBoost

算法思路

通过节点分裂来生成一棵树，再不断添加树来拟合上一次拟合的残差。

求叶子节点分数目标函数

$$Obj = \sum_{i=1}^{n} l(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k)$$

Training loss　　Complexity of the Trees

其中第一部分为估测值和目标值的差距，第二部分为正则化项，防止过度拟合。

分裂节点算法

$$\min_{j,s}\left[\min_{c_1}\sum_{x_i \in R_1(j,s)}(y_i - c_1)^2 + \min_{c_2}\sum_{x_i \in R_2(j,s)}(y_i - c_2)^2\right]$$

其中R1是分到左子树的特征值范围，R2是分到右子数的特征值范围。c1和c2分别是左子树和右子树代表的分数值。

拟合残差

把实际分数值减掉上一次决策树的分数值，生成新的决策树来拟合上一次的残差。这样做可以避免一棵树过拟合的问题。

python实现

```
dataset1 = xgb.DMatrix(dataset1_x,label=dataset1_y)
dataset2 = xgb.DMatrix(dataset2_x,label=dataset2_y)
dataset12 = xgb.DMatrix(dataset12_x,label=dataset12_y)
dataset3 = xgb.DMatrix(dataset3_x)

watchlist = [(dataset12,'train')]
model = xgb.train(params,dataset12,num_boost_round=3500,evals=watchlist)
```

xgb.train()参数:

booster:可选择gbtree和gblinear两种，分别对应于基于树的结构进行提升运算，和线性的提升运算。

num_features:在过程中用到的特征维数。

调参结果

params = {'booster': 'gbtree',
'objective': 'rank:pairwise',
'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,
'nthread': 12
}

数据封装方式：用xgb.Dmatrix(data,label)

模型融合

线性加权

这里我们用的是投票法和自定义权值法

- 投票法

  根据RF,GBDT,XGB三个模型的AUC面积排名, 分别占1/6,2/6,3/6的权值,即

  Label = 1/6*RF(Test_data)+2/6*GBDT(Test_data)+3/6*XGBOOST(Test_data)

- 自定义权值法

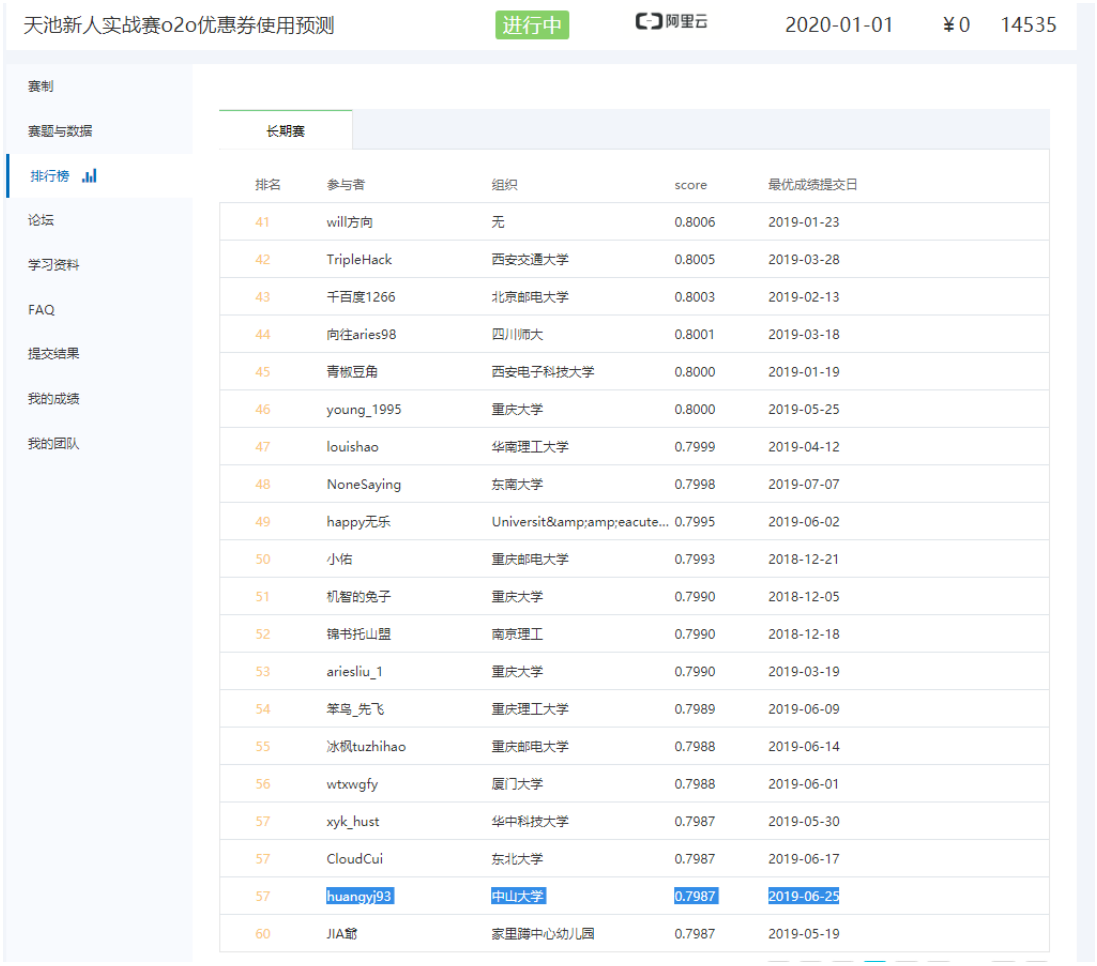
  实验过程中发现, 随机森林模型的预测结果不尽人意, 所以我把它从训练模型中剔除, 把GBDT权值改为0.3, 把XGB权值改为0.7, 即

  Label = 0.3*GBDT(Test_data)+0.7*XGBOOST(Test_data)

实验结果

模型	成绩
Randomforest	0.7731
GBDT	0.7973
GBDT+lightgbm	0.7753
XGBOOST	0.7987
Voting Ensemble	0.7856
Customized Weight Ensemble	0.7875

最终成绩： 57/14535


天池新人实战赛o2o优惠券使用预测
进行中
阿里云
2020-01-01
¥0
14535
赛制
赛题与数据
排行榜
论坛
学习资料
FAQ
提交结果
我的成绩
我的团队
长期赛
排名 参与者 组织 score 最优成绩提交日
41 will方向 无 0.8006 2019-01-23
42 TripleHack 西安交通大学 0.8005 2019-03-28
43 千百度1266 北京邮电大学 0.8003 2019-02-13
44 向往aries98 四川师大 0.8001 2019-03-18
45 青椒豆角 西安电子科技大学 0.8000 2019-01-19
46 young_1995 重庆大学 0.8000 2019-05-25
47 louishao 华南理工大学 0.7999 2019-04-12
48 NoneSaying 东南大学 0.7998 2019-07-07
49 happy无乐 Universit&eacute... 0.7995 2019-06-02
50 小佑 重庆邮电大学 0.7993 2018-12-21
51 机智的兔子 重庆大学 0.7990 2018-12-05
52 锦书托山盟 南京理工 0.7990 2018-12-18
53 ariesliu_1 重庆大学 0.7990 2019-03-19
54 笨鸟_先飞 重庆理工大学 0.7989 2019-06-09
55 冰枫tuzhihao 重庆邮电大学 0.7988 2019-06-14
56 wtxwgfy 厦门大学 0.7988 2019-06-01
57 xyk_hust 华中科技大学 0.7987 2019-05-30
57 CloudCui 东北大学 0.7987 2019-06-17
57 huangyj93 中山大学 0.7987 2019-06-25
60 JIA爺 家里蹲中心幼儿园 0.7987 2019-05-19

提交记录

- 黄杨峻

日期: 2019-06-26 19:34:00 排名: 3
score: 0.7856

score

日期: 2019-06-25 23:14:00 排名: 1
score: 0.7987

日期: 2019-06-24 23:07:00 排名: 3
score: 0.7973

日期: 2019-06-24 16:30:00 排名: 11
score: 0.4879

日期: 2019-06-23 22:11:00 排名: 4
score: 0.7731

日期: 2019-06-22 16:58:00 排名: 6
score: 0.4661

- 陈飞佚

日期: 2019-07-10 21:51:00 排名: 2
score: 0.7753

日期: 2019-06-30 18:48:00 排名: 3
score: 0.5000

日期: 2019-06-23 22:10:00 排名: 4
score: 0.7731

暂无更多数据

- 刘学海

长期赛

日期: 2019-06-26 19:53:00 **排名:** 2

score: 0.7875

暂无更多数据

实验总结

- 做数据分析，确定分析目的与方法论很重要。本题中的数据分析的目的是：是预测用户领取优惠券后15天以内的使用情况，而使用情况的只有两种：即使用与未使用。是二分类问题，考虑到是求解使用概率，使用xgboost算法较为合适。
- 模型的准确性，特征的提取很关键。
- 对使用python进行数据分析、特征提取、模型训练的理解更进一步。
- xgboost在该问题中表现出很高的准确率，收敛速度也相对较快。gbdt的准确率稍微比xgboost差，但也表现不俗。rf的准确率差强人意。
- 模型的准确性，特征的提取很关键。
- 对使用python进行数据分析、特征提取、模型训练的理解更进一步。
- 在该实验中，模型融合并不能给准确率带来提升，因为使用到的算法分类方法近似。