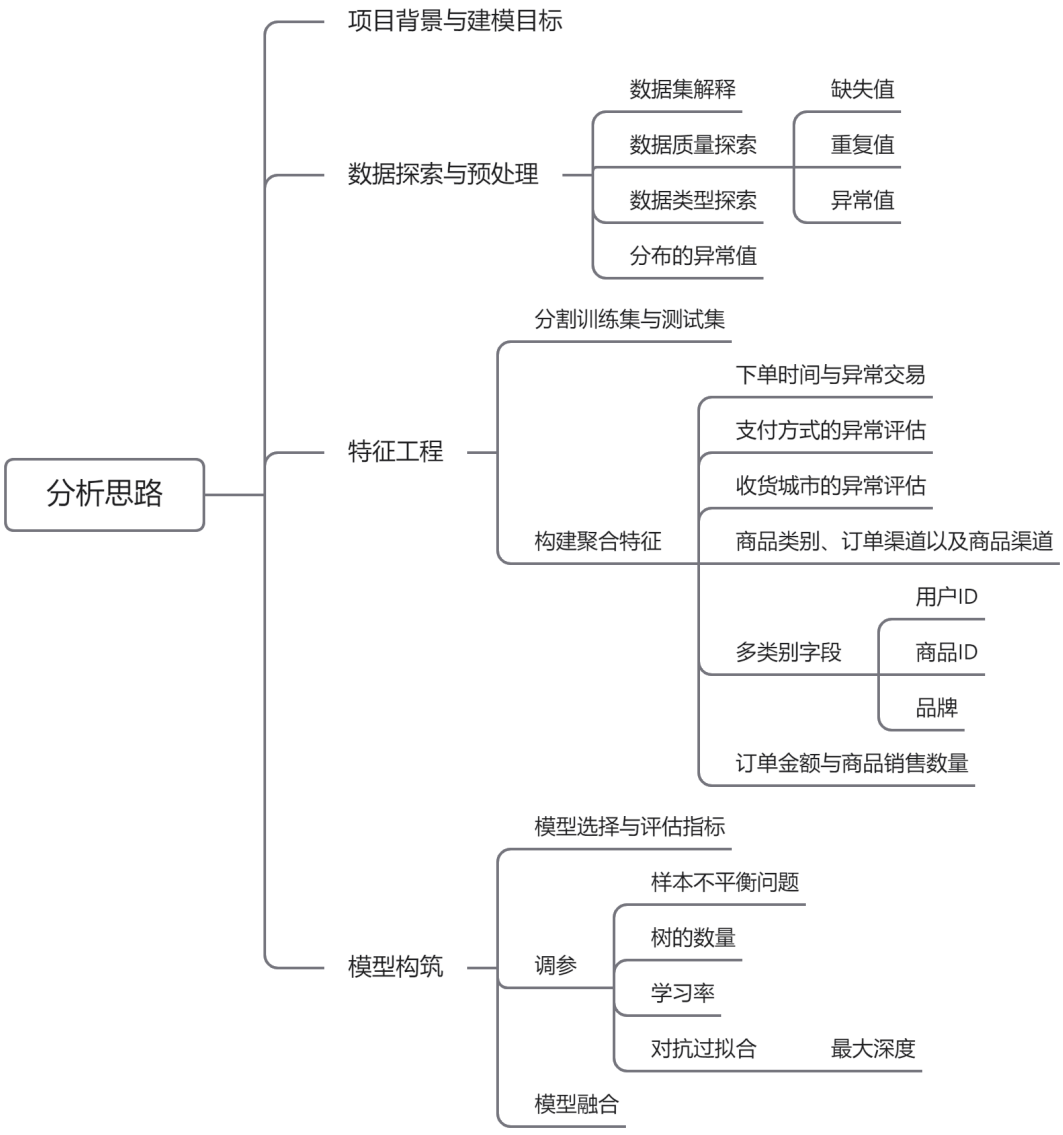
电商异常订单检测

一、项目背景与建模目标

本项目是针对电商平台交易数据的异常订单预测，数据来源于互联网。项目的分析目标是根据用户在平台的交易数据，预测订单是否异常。

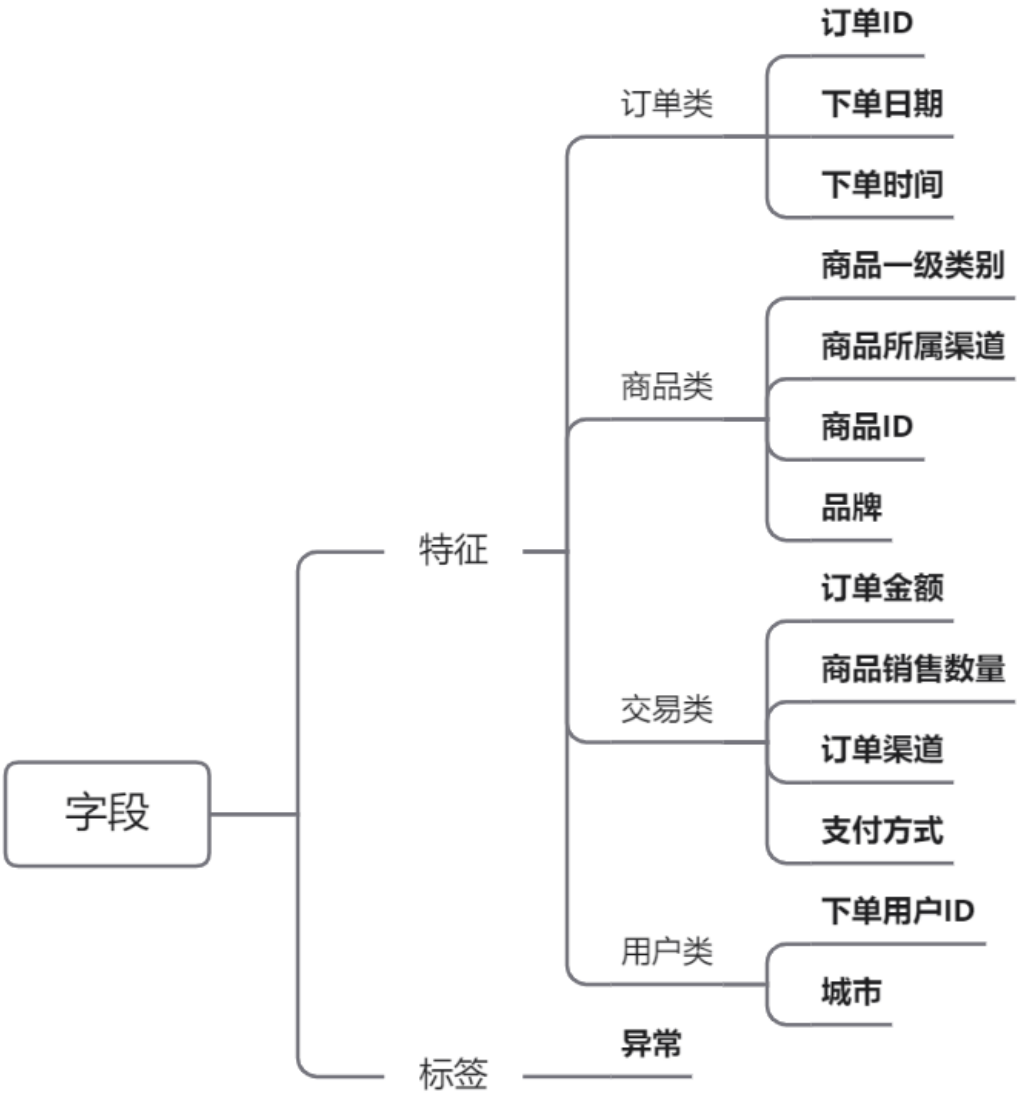
分析思路如下：



二、数据探索

1、数据集解释

数据规模为134190×14。数据集的字段如下：



2、数据质量探索

（1）、缺失值

数据集各列缺失比例如下：

```
订单ID          0.000000
下单日期          0.000000
下单时间          0.000000
商品一级类别        0.010358
商品所属渠道        0.000000
商品ID          0.000000
品牌            0.005753
订单金额          0.000007
商品销售数量        0.000000
订单渠道          0.000000
支付方式          0.000000
下单用户ID        0.000000
城市            0.000015
异常            0.000000
dtype: float64
```

其中'商品一级类别'缺失了1.03%，‘品牌’缺失了0.57%。‘订单金额’与‘城市’缺失比例过低，直接删除。考虑'商品一级类别'与‘品牌’缺失或许与‘交易异常’有联系，计算得出

- 缺少(商品一级类别)的样本中异常样本占比为11.727%
- 缺少(品牌)的样本中异常样本占比为20.984%

该数值说明，商品并不会因为缺少品牌信息或者缺少商品一级类别信息就被判断为交易异常，两者联系并不大。

- 全部交易异常样本中，缺少(商品一级类别)的异常样本占比为0.573%
- 全部交易异常样本中，缺少(品牌)的异常样本占比为0.569%

并且，全部交易异常的样本中，缺少品牌信息或者缺少商品一级类别信息的占比非常小，可以直接删除。

（2）、异常值

没有小于0等业务逻辑上不该出现的异常值。同一'订单ID'下总是出现不同'下单日期'同一'下单时间'的情况，说明许多样本的'下单日期'特征有误。删除'下单日期'这个特征。

（3）、重复值

整个数据集有8个重复值，“订单ID”列数据中存在18554个重复值，因此每个样本大概率并不是订单，可能是订单中的一个商品/一笔交易。删除数据集的8个重复值。验证得知，一个订单ID下的样本标签几乎是一致的，订单ID对标签的预测有影响，保留这个特征代入数据进行建模。

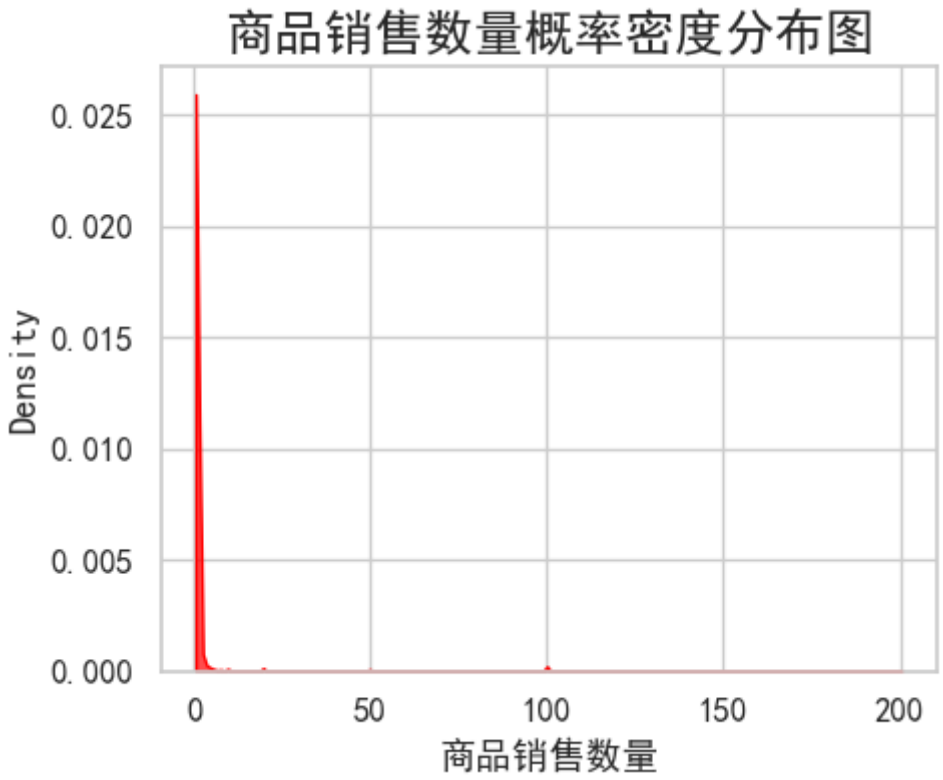
3、数据类型探索

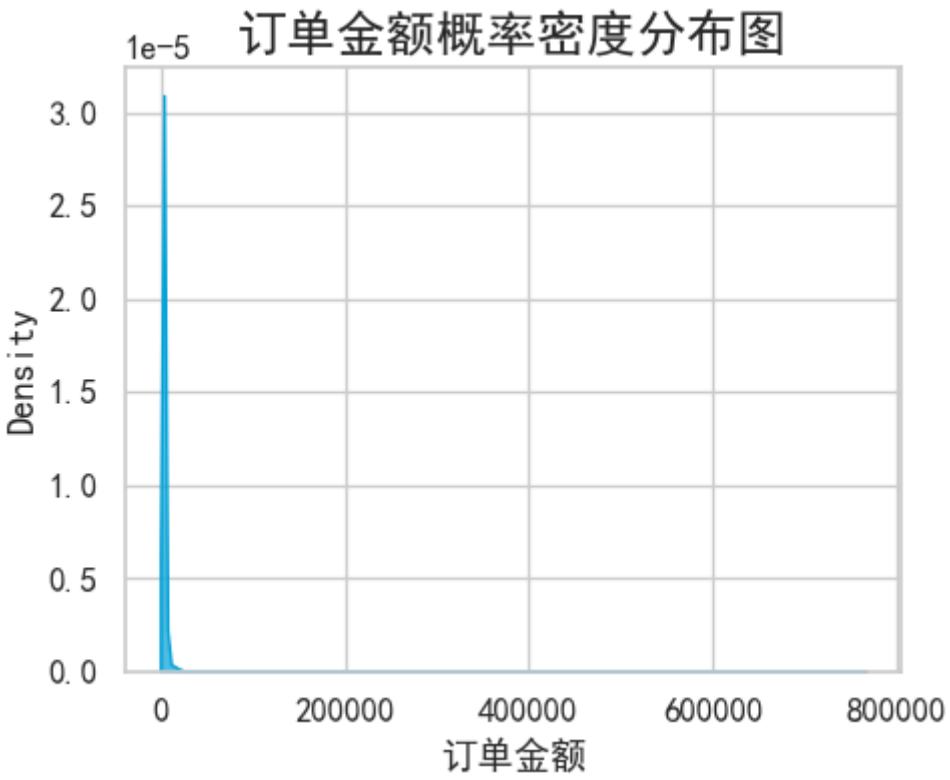
```
0   订单ID     134190 non-null  int64
1   下单日期     134190 non-null  object
2   下单时间     134190 non-null  object
3   商品一级类别  132800 non-null  object
4   商品所属渠道  134190 non-null  object
5   商品ID     134190 non-null  int64
6   品牌       133418 non-null  object
7   订单金额     134189 non-null  float64
8   商品销售数量  134190 non-null  int64
9   订单渠道     134190 non-null  object
10  支付方式     134190 non-null  object
11  下单用户ID  134190 non-null  object
12  城市       134188 non-null  object
13  异常       134190 non-null  int64
```

该数据集的有效特征很多都是object类型，有很大的数据预处理空间，唯一的浮点数类型特征是订单金额。

4、分布的异常值

数据中有且只有交易金额、交易数量这两个连续型特征，且许多离散型特征的"分布"对当下的预测并无太大意义(如品牌、商品一级类别、城市)，因此只对两个连续型特征做统计。





两个分布右偏严重，存在大量长尾数据，使用箱线图法则进行异常检测，检测出29949个样本被标记为特征分布异常值，其中被标记为交易异常的有7368个样本，超过整体异常值的1/4。因此异常值不能直接删除。

按'是否有特征异常'构成新的特征，计算该特征与标签的相关性。

	特征异常	异常
特征异常	1.000000	0.040428
异常	0.040428	1.000000

特征异常与实际交易异常之间相关性非常微弱，可以不使用这个特征。从电商业务角度来看，异常基本出现在用户支付大量金额，或者一次性购入大量物品时，在交易中比较普遍，因此不处理异常值。

三、特征工程

1、分割训练集与测试集

在该数据集中，每个样本是针对单一商品的一笔交易，多笔交易构成了一个订单，且在一个订单下所有样本的标签是一致的。一笔交易异常，整个订单都异常。此时，用一个订单中的部分交易去预测相同订单中的其他交易是否异常没有意义。

因此在分割训练集和测试集时，需要**保证同一订单下的样本不会被同时分到训练集与测试集中**，测试集的订单ID一定是训练集中没有出现过的订单ID。需要对订单ID，而非整个样本进行分割，然后再根据分割出的训练集ID和测试集ID构建训练集和测试集。

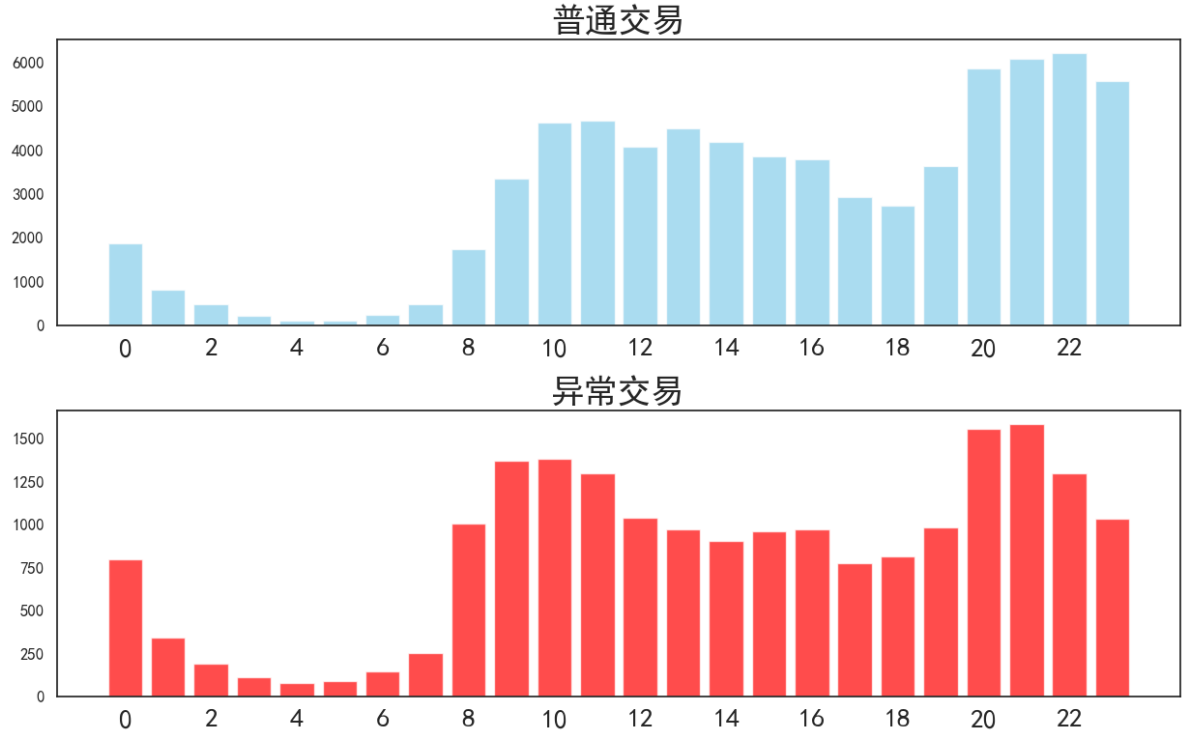
（1）、从全部的订单ID中，按照30%的比例，抽样出测试用的ID

（2）、根据分割出的订单ID，将数据中的样本标记出来

实际样本比例为0.3002239。

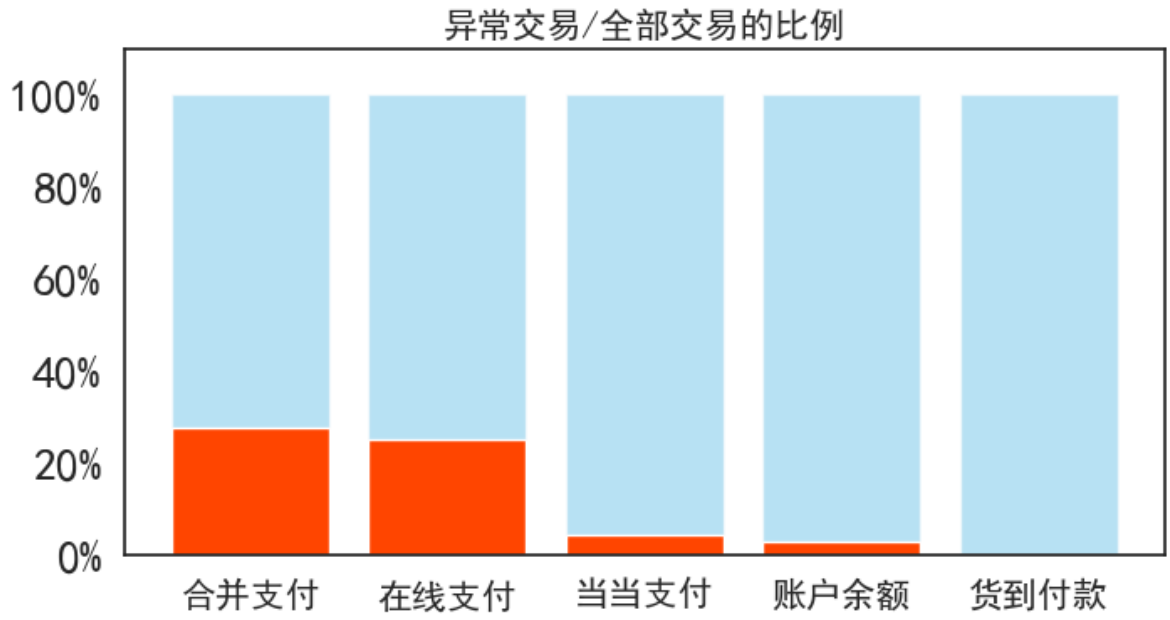
2、下单时间与异常交易

提取下单小时，异常订单的下单小时与普通订单下单小时每个区间的交易数量对比分析如下：



异常订单与正普通订单的下单时间没有明显区别，都集中在8-24点，且趋势相似，发生在深夜的异常订单很少，因此下单小时可能与交易是否异常无关。保留这个特征，转化为小时和分钟。新建'下单小时'，下单分钟（>30分钟为1，≤30分钟为0）两列，删除下单时间列，并进行特征编码。

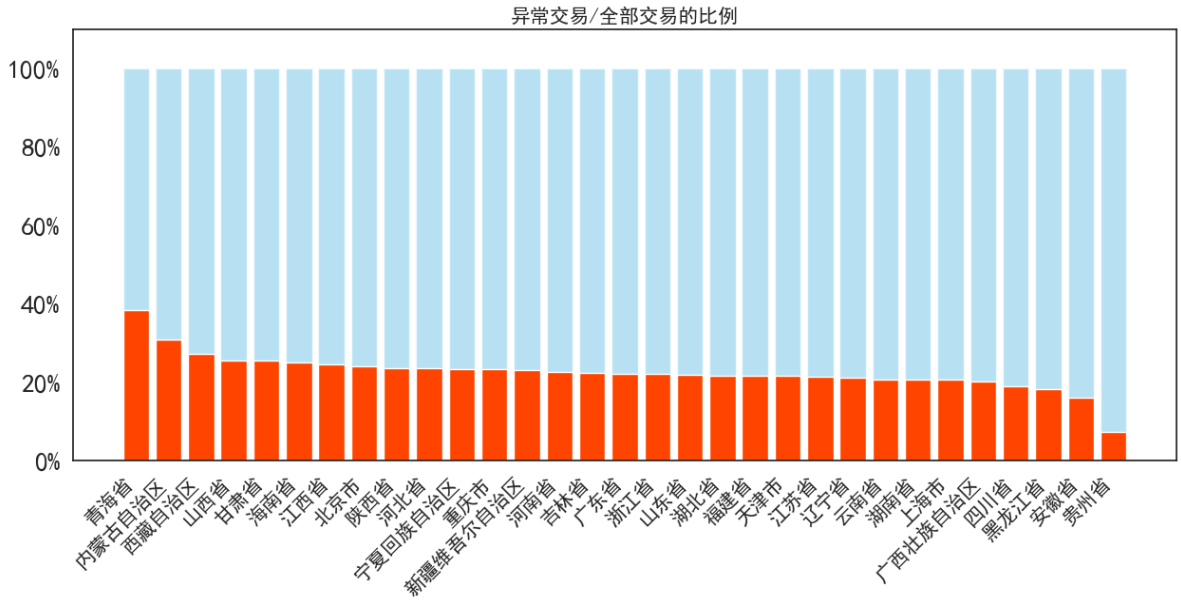
3、支付方式的异常风险评估



合并支付和在线支付两种方式的异常率很高，占了该支付方式下全部样本的25%以上，剩下的支付方式风险很小。将各支付方式的异常率作为聚合特征放入特征矩阵中，**建立新特征'支付方式异常率'。完成'支付方式'列的特征编码。**

4、收货地点的风险评估

从理论上来说，收货地点与交易异常应该不存在明显的联系，使用异常率来进行相关的评估。城市种类远远超出一般的离散型特征分类个数，需要降低城市类别数量再进行风险评估。按照省进行分类，再按照省份聚合。



异常率在省份上表现出较大的差异，其中青海、内蒙古自治区的异常率远高于其他城市，可能是由于物流发展程度较低导致。其他大部分城市的异常率都在20~25%之间。

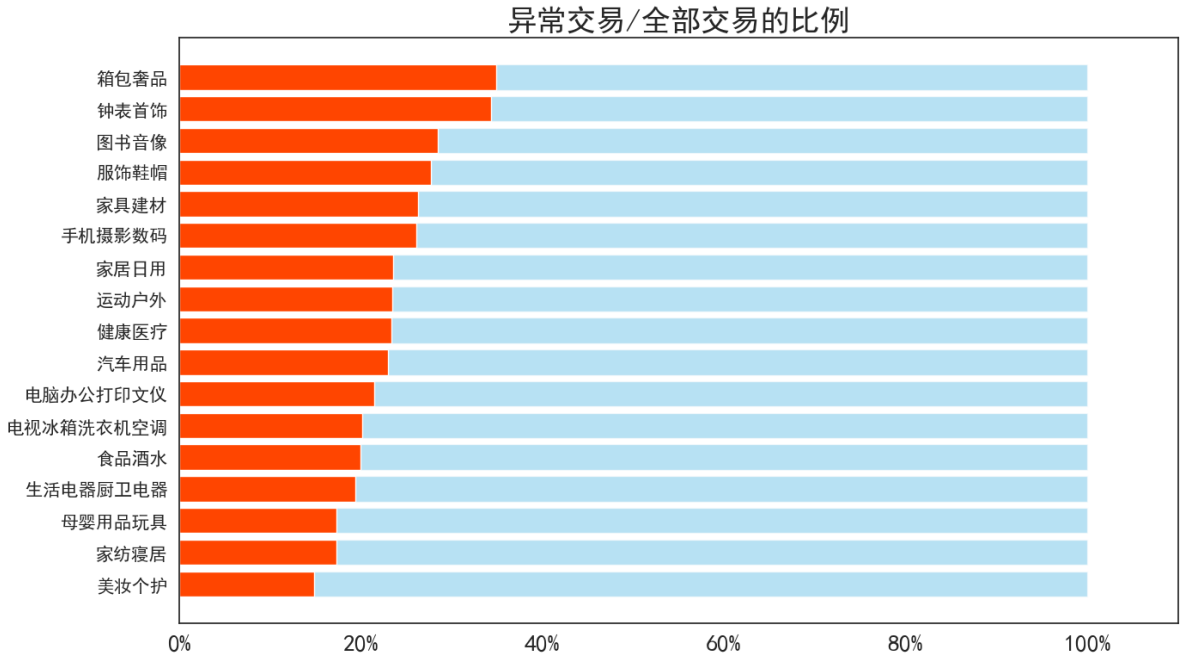
依据省份进行聚合的异常率作为新的特征放入特征矩阵。**创建新特征'省份'，'省份异常率'，对'城市'和'省份'进行特征编码**，经检验，测试集中城市有三个缺失值，将这三个缺失值编码为362（训练集中有362种城市）。

5、商品类别、订单渠道以及商品渠道

如果某一类/某一件商品具有较高的退货、敲诈、纠纷风险，那这一类/一件商品的订单中、异常订单的比例应该很高。若存在这种商品，则可以在特征工程中单独标注高风险商品。同理，对于订单渠道和商品渠道，也可以按异常率增加特征。

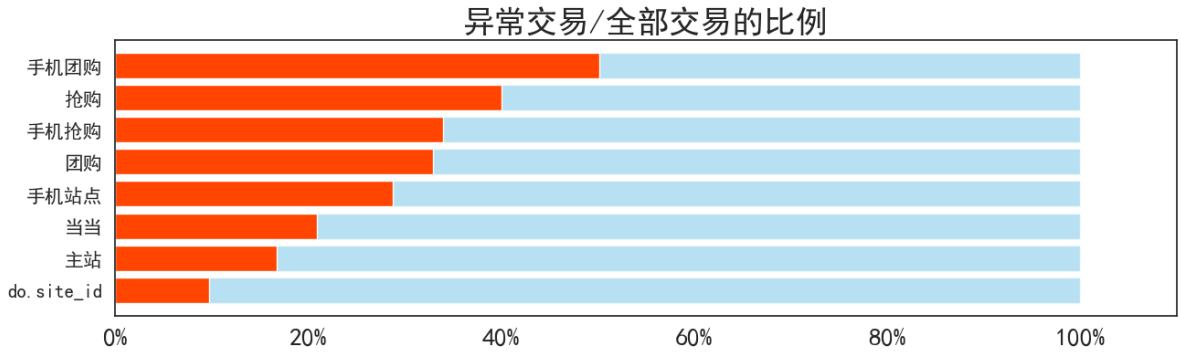
- **商品一级类别**

对比每种商品类别下的异常订单率：



从结果来看，异常交易比例最高的是钟表首饰、箱包奢品、图书音像，最低的是美妆个护、家纺、母婴用品。

- **订单渠道**



手机团购、抢购模式出现异常的可能性更高，移动端比PC段出现异常的可能性更高。

- **商品所属渠道**

```
商品所属渠道
POP     0.294727
GO      0.178732
Name: 异常, dtype: float64
```

POP店铺异常率较高，GO(官方自营店铺)的异常率较低。

分别加入新特征'商品一级类别异常率','商品所属渠道异常率','订单渠道异常率'，对'商品一级类别','商品所属渠道','订单渠道'进行特征编码。

6、多类别字段

用户ID、商品ID、商品品牌是三个特殊的离散型特征，在一般的数据中，ID这种带有一定唯一性的特征，可以采取直接删除或无视的处理方法，但在特殊的预测场景下，订单可能因为涉及到异常用户、异常商品甚至异常品牌因此被判做"异常"，所以需要保留用户ID、商品ID、商品品牌这三个特征，还需要对这三个特征进行探索和处理。

这三个特征有以下两个共同点：

（1）一个ID/品牌只对应一个样本的情况

在一个商品类别/一个收货城市下，最少也有几十上百次交易，因此交易中的异常率可以被认为是大量实验后的可信结果。但一个商品或一个品牌可能在全年之中只有寥寥几笔交易（甚至一笔），对于用户来说，全年可能只进行一次交易。这种情况下无论交易是异常还是正常，都无法被用来评估该商品/用户的风险。因此需要将样本分为两类:

(1)交易过少无法判断的类型

(2)交易量足够、可以衡量风险的类型

对交易过少的类型进行特殊标记，对交易足够的类型使用异常率。

（2）对比其他离散型特征如商品以及类别、城市等，这三个特征中的类别数量尤其地多

在分割训练集与测试集时，一般默认离散字段中的所有类别都会同时出现在训练集和测试集中。例如城市、商品一级类别，train与test中应该都包含了所有的类别，这样才能使用相同的字典去对字段进行编码。但当一个离散字段中的类别数量非常多时，就无法保证所有的类别会同时出现在训练集和测试集中了。如此带来的问题是，测试集中出现了训练集中从未见过的分类或数据，不能再使用同样的字典进行编码，否则会留下空值。

先来分析异常率

- **用户ID**

大部分人都只下了一单，这些只下了一单的人无法分析异常率。对这些人给与-1的用户异常率，向算法表明这类人与其他类型不同，提取下单次数超过1次的用户，计算异常率。

- **商品ID**

人为规定，商品销售数量小于10个的商品，是交易过少，无法判断其异常风险的商品。计算商品销售数量≥10的商品异常率

- **品牌**

与商品ID一样，使用商品销售数量来衡量品牌交易量。均值为43，可以规定销售数量小于10个的品牌为交易过少无法判断其异常风险的商品对销售数量≥10的品牌，计算异常率。

将用户异常率、商品异常率、品牌异常率作为聚合特征，放入特征矩阵。

对训练集来说，交易量不足的部分标注为-1，交易量足够的部分标注为异常率。

对测试集来说，需要检查该测试样本的ID是否出现在训练集中，如果出现，则使用训练集中的标注，如果不出现，则标记为-1。

7、订单金额与销售数量

在电商环境中，大部分交易的对象都是个人用户，因此如果订单的金额过于巨大、或一次性售出的数量过于巨大，那订单存在异常的可能性就很高。

对订单金额与销售数量进行分箱，并将每个箱内异常交易的比例进行对比。各箱之间差异较大，根据分箱后的结果创造新特征。

将**分箱结果**放置到训练集和测试集中

将分箱聚合后的**平均金额/平均销量**放到训练集和测试集中

将分箱聚合后**异常率**放到训练集和测试集中

四、模型构筑：使用模型融合进行预测

1、模型选择与评估指标

本项目选择随机森林、GBDT和XGBoost，使用roc_auc作为评估指标。

算法	初始训练
随机森林	0.92711
GBDC	0.93506
XGBoost	0.93312

2、参数调优

集成模型的默认参数都是基于经验精心设计的，使用默认参数时，模型就已经达到了某种上限，因此调参在集成模型上基本是对结果进行"微调"，集成模型调参之后的表现不会与默认参数的表现差距太大。因此，为了大幅度提升模型的效果，需要先考虑一些激进的方案。

（1）样本不平衡

对于一个不均衡的样本来说，处理样本不均衡问题往往能够大幅度提升模型auc和准确率。XGBoost的默认参数中默认会将正负样本比例修正为1:1，而随机森林和GBDT则不会做出这样的修正，在这种情况下，三个模型在AUC上的表现看起来高度相似，就说明样本不均衡对于现在的数据集来说可能不是问题。

无论如何，需要去做出尝试。GBDT没有修复样本不均衡的参数，因此GBDT无法在这个点上进行调整，先来看看xgboost和随机森林的效果。

训练集的异常样本占比为0.216778。

- **随机森林**

参数class_weight分别输入{0:0.5,1:0.5},{0:0.5,1:1},'balanced',{0:0.5,1:2},{0:0.5,1:2.5},{0:0.5,1:3}，计算

```
{0: 0.5, 1: 0.5}
        rf_mean:0.92711
        rf_var:1.6078274716054552e-06
{0: 0.5, 1: 1}
        rf_mean:0.92781
        rf_var:1.8249276246385685e-06
balanced
        rf_mean:0.92802
        rf_var:1.5927988731141121e-06
{0: 0.5, 1: 2}
        rf_mean:0.92817
        rf_var:2.4615356108817417e-06
{0: 0.5, 1: 2.5}
        rf_mean:0.92822
        rf_var:1.995297062898542e-06
{0: 0.5, 1: 3}
        rf_mean:0.92819
        rf_var:7.815637153552289e-07
```

可以看到，class_weight的调整只是对随机森林的结果造成了微弱的影响。随着少数类比例的增加，AUC分数先有一个上升，后有所下降，auc最高的0.5:2.5就是最好的比例。暂定对**随机森林的class_weight使用0.5:2.5。**

- **xgboost**

使用参数scale_pos_weight，在这个参数中需要输入多数类比少数类的比例(sum(negative instances) / sum(positive instances)。默认的参数值是1,即XGBoost默认就对不均衡样本做出了处理。先尝试0.5,0.3,0.1三种选项。

```
0.5
        xgb_mean:0.93378
        xgb_std:7.957847222655731e-07
0.3
        xgb_mean:0.93378
        xgb_std:6.515941772162028e-07
0.1
        xgb_mean:0.93483
        xgb_std:1.1014901297317455e-06
```

从结果来调整参数scale_pos_weight是有效的，随着少数类比例越来越大，模型的auc是上升的，尝试更激进的方案。

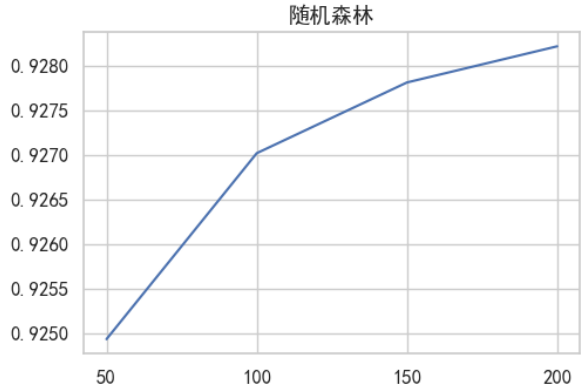
```
0.05
        xgb_mean:0.93499
        xgb_std:1.418242009439362e-07
0.03
        xgb_mean:0.93523
        xgb_std:8.191463106040282e-07
0.01
        xgb_mean:0.93515
        xgb_std:1.2536964882767838e-06
```

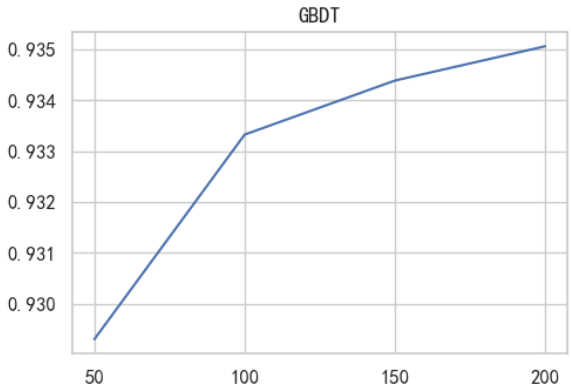
均值最高的大约是负样本：正样本比例为3：100时，xgboost的auc是最高的。暂定**scale_pos_weight的比例为0.03。**

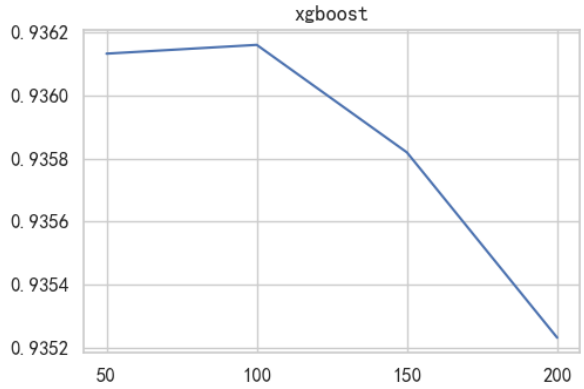
算法	初始训练	样本均衡
随机森林	0.92711	0.92822 **(+)**
GBDC	0.93506	—
XGBoost	0.93312	0.93523 **(+)**

（2）树的数量

减少树的数量



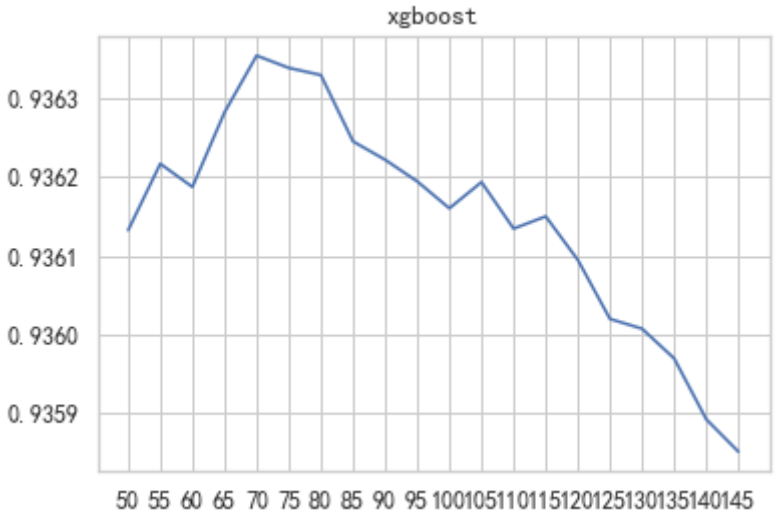




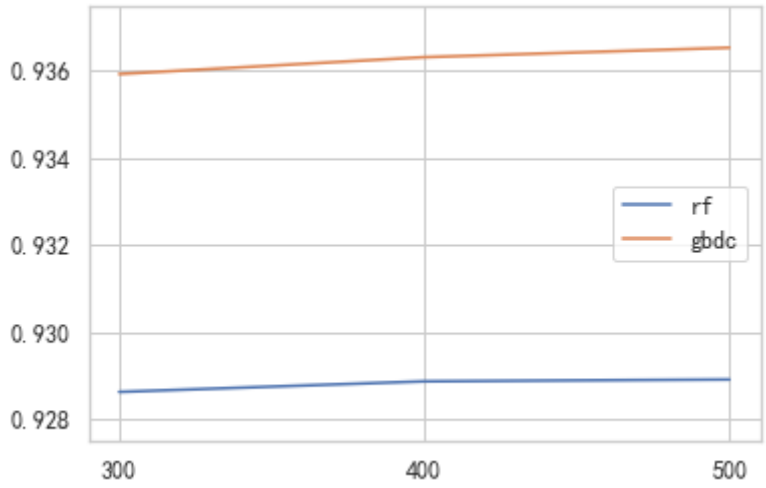
对于xgboost来说，100棵树左右auc分数是理想的

对于随机森林和GBDT来说，随着树的持续增长，交叉验证中的auc分数还在增长

对xgboost进行50-150之间的精密调参，对森林与gbdt继续增加树的数量。



可以发现最高auc出现第70棵树时，auc达到了0.93636，对于**xgboost确定参数num_boost_round=70**



随机森林与GBDT的结果随着树的数量上升、持续增长中，说明模型还有潜力，但在PC端运行时，再增加树的数量会增加太多运算时间和运算成本，因此需要考虑换一种思路来让模型更快地提升分数。

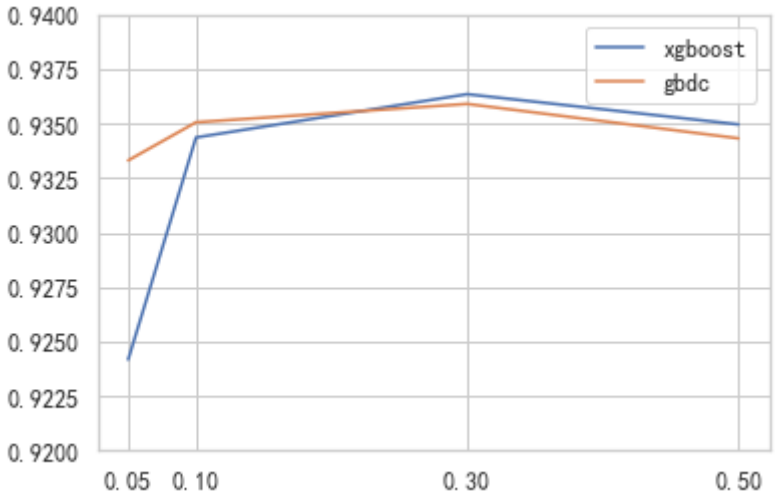
对于GBDT来说，可以调整学习率，让树迭代得更快，以此来尝试减少树的数量。对于随机森林来说，可以从对抗过拟合的角度来考虑，降低每棵树的复杂度，虽然树的数量很多，但是可以尽量加快建树的速度基于这样的结论，GBDT的树数量暂定200，森林的树数量暂定为500。

算法	初始训练	样本均衡	树的数量
随机森林	0.92711	0.92822 **(+)**	0.92892 **(+)**
GBDC	0.93506	—	—
XGBoost	0.93312	0.93523 **(+)**	0.93636 **(+)**

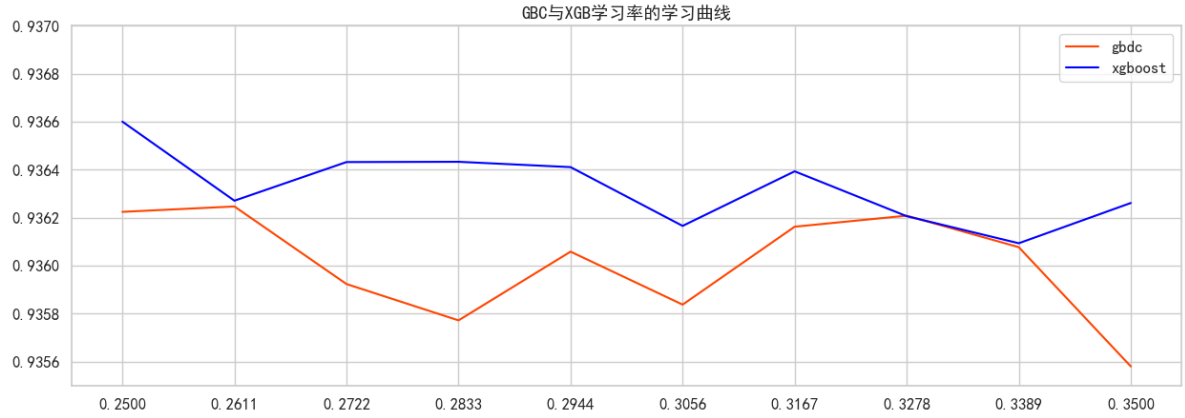
(3) 学习率

随机森林没有学习率调整的参数，但XGBoost与GBDT的学习率都是可以调整的，其中GBDT的参数learning_rate(默认0.1)，XGBoost的参数是eta(默认0.3)

对这两个算法，可以尝试将学习率的取值向两边拓展，尝试分别取值0.05,0.1,0.3,0.5。

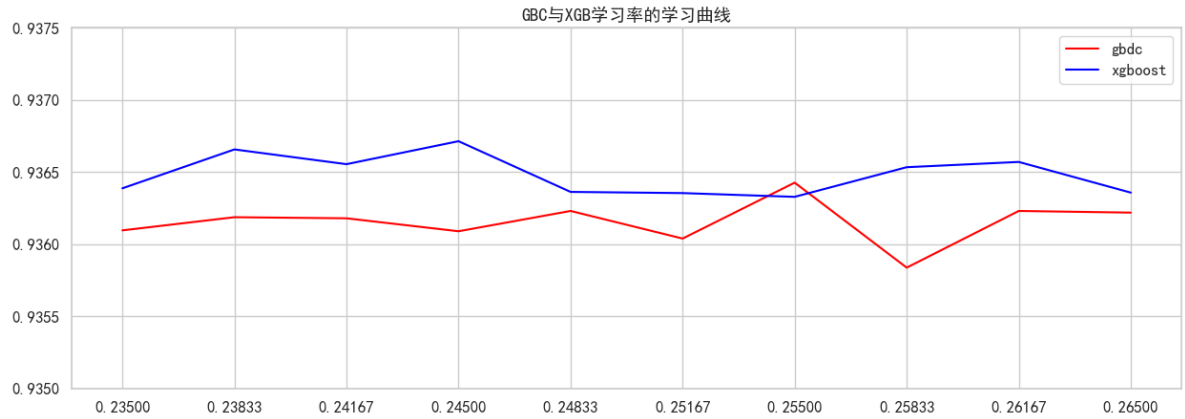


对GBDT来说，将学习率从0.1增加到0.3，能够一定程度上提升模型的效果。对XGBoost来说，默认值0.3是更适合的数字。因此可以尝试在0.3周围小范围进行调整:



GBDC最高分是0.93625,最高分对应的学习率是0.2611111111111111
XGBoost最高分是0.93660,最高分对应的学习率是0.25

继续缩小范围:

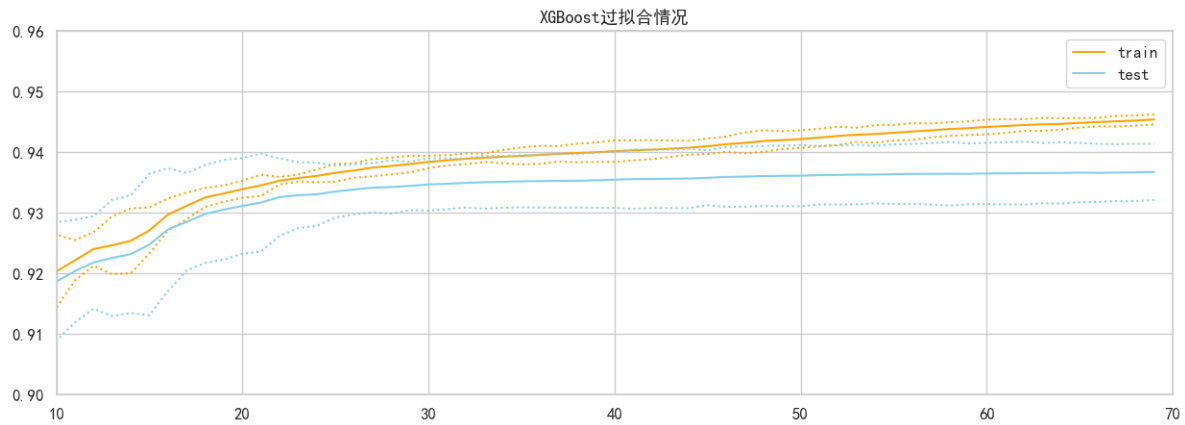


GBDC最高分是0.93643,最高分**对应的学习率是0.255**
XGB最高分是0.93671,最高分**对应的学习率是0.245**

算法	初始训练	样本均衡	树的数量	学习率
随机森林	0.92711	0.92822 **(+)**	0.92892 **(+)**	—
GBDC	0.93506	—	—	0.93643 **(+)**
XGBoost	0.93312	0.93523 **(+)**	0.93636 **(+)**	0.93671 **(+)**

（4）对抗过拟合

XGBoost过拟合情况:



从结果来看，xgboost上的过拟合不是特别严重，这与xgboost的树数量调整为70有很大的关系。同时xgboost的默认参数也是偏向于天生控制过拟合（例max_depth=6），因此xgboost在抑制过拟合后得到好结果的可能性就非常小了。

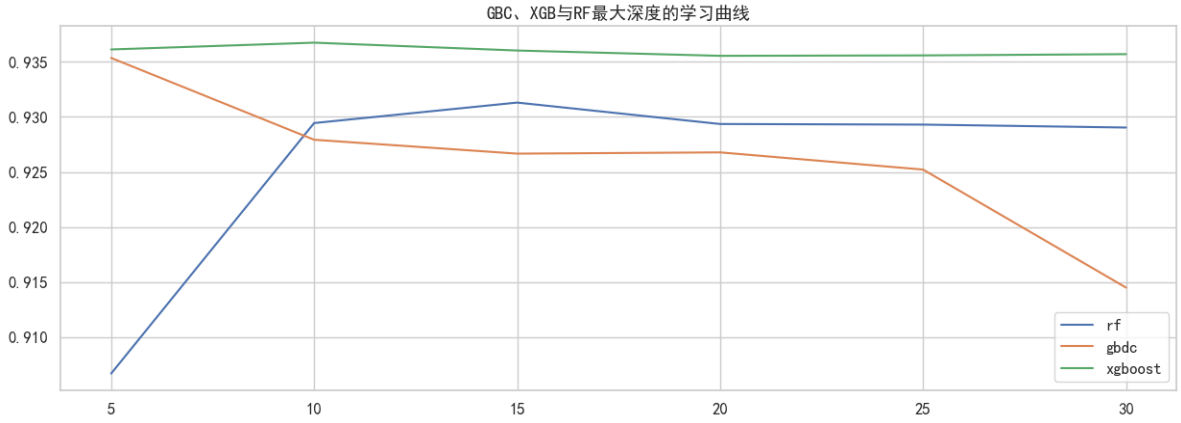
sklearn中的交叉验证并不能调出训练集的结果，无法判断随机森林与GBDT是否处于过拟合的状态，但从树的数量来看，应该是比xgboost有更大的空间的。

尝试控制过拟合来提升模型的泛化能力。

在树模型的参数中，大量的参数都是用来对抗过拟合的，这与树模型天生学习能力较强、很容易的过拟合的性质有关。在对抗过拟合的参数中，最有效的是max_depth，如果max_depth展现出一定的效果，可以继续调节其他控制过拟合的参数。如果max_depth完全无效、甚至让模型效果变弱，那说明现在调整过拟合可能不是一个很好的选择。

在GBDT的默认参数中，max_depth=3，因此GBDT本身应该就带有一些抗过拟合的性质。随机森林的max_depth默认值为None，因此理论上来说森林的树应该比GBDT的树要深很多。

可以尝试调小max_depth，来观察模型的效果：

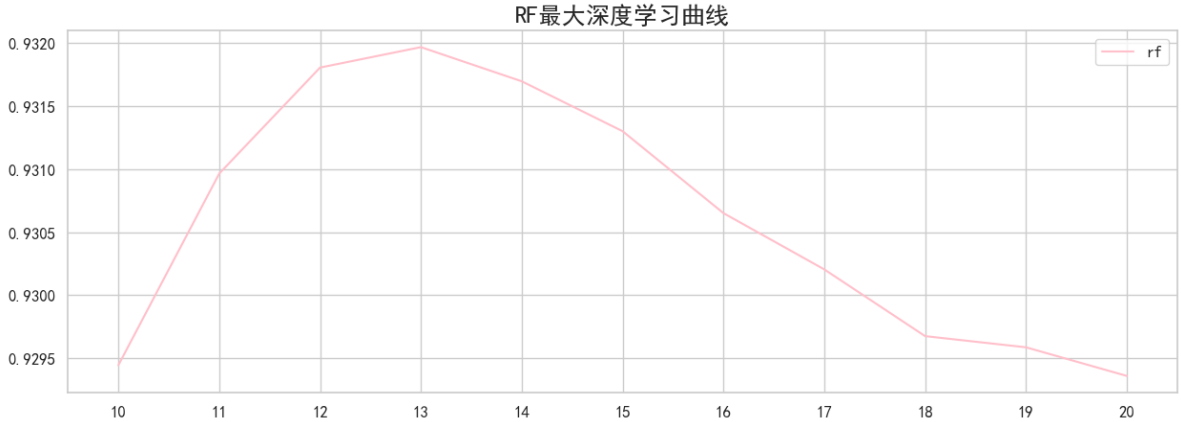


GBDC最高分是0.93535,最高分对应的最大深度是5
XGB最高分是0.93674,最高分对应的最大深度是10
rf最高分是0.93130,最高分对应的最大深度是15

不难发现，在max_depth从5到10，森林的AUC提升较大，这说明随机森林的确存在过拟合的情况，并且使用max_depth可以很好地控制过拟合、提升模型效果。

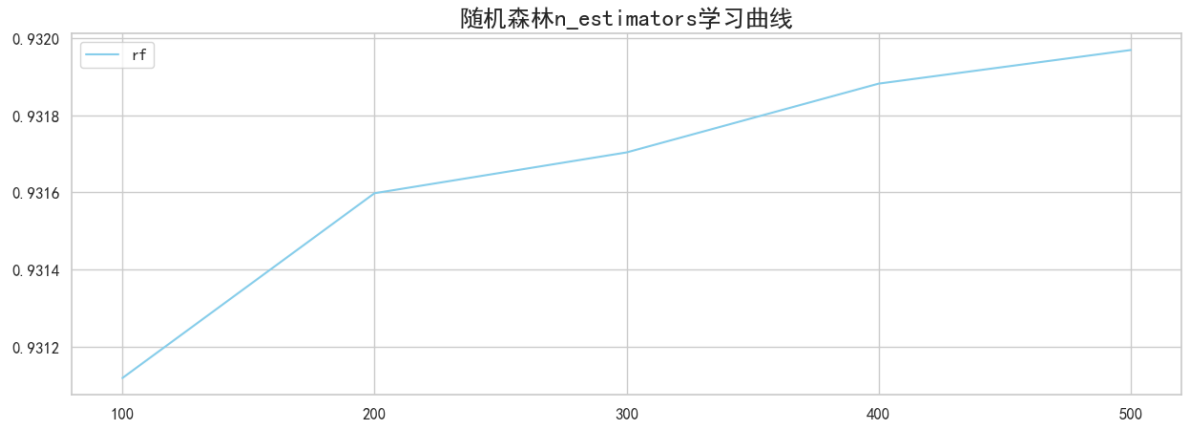
相对的，**GBDT与XGBoost在改变max_depth之后都没有取得比默认值更高的结果。**在这两个算法的效果都高于森林的情况下，可以以提升随机森林的效果为主，这样之后在进行模型融合时才能够提升融合模型的综合水平。

现在对随机森林的max_depth进行更加情细的调整。



可见，**最大深度为13**时模型的效果很好，达到了0.93197

现在反过来调整随机森林树的数量



虽然限制了过拟合，但并没有改变随机森林的结果随着树的数量增多持续上升的情况，**从对模型效果的追求来看，最终选择500棵树。**

算法	初始训练	样本均衡	树的数量	学习率	最大深度
随机森林	0.92711	0.92822 **(+)**	0.92892 **(+)**	—	0.93197 **(+)**
GBDC	0.93506	—	—	0.93643 **(+)**	—
XGBoost	0.93312	0.93523 **(+)**	0.93636 **(+)**	0.93671 **(+)**	—

（5）模型融合

现在对于三个模型，使用软投票规则。首先在每个模型上对训练集进行训练，并且让模型输出训练集的预测概率。根据三个模型的ROC分数，规定随机森林、GBDC、xgboost权重分别为4、0.8、0.2。将所有的样本为1、0的概率分别按权重加和，将为1的概率压缩到[0,1]之间，当做最终的概率来输入auc计算函数。得整体的auc为0.960415。对测试集进行相同操作，得到的整体auc为0.819920。

- **总结**

从结果来看，模型在测试集上的表现还算不错，虽然远低于训练集，整个模型还是处于过拟合状态，但是这个分数在业务上可以起到"预警"作用了。在训练集的交叉验证中，模型已经展现出了很好的泛化能力，但鉴于数据本身的属性，测试集与训练集的数据属性/分布可能差异太大，(例如新订单包含了太多过去的订单没有的信息)，导致测试集上的结果不是那么尽如人意。对训练集相似的数据，模型可以给出很好的泛化结果。

- **改进空间**

1、使用业务中获得的标签相关的规则，例如，相同订单号的交易必然有相同的标签，来修正最终的预测结果

2、继续进行更精细的调参，提升模型的泛化能力

3、利用特征合成方法，形成更多聚合特征，来提升数据的质量

4、建立更多的集成模型，增加融合模型中的模型数量