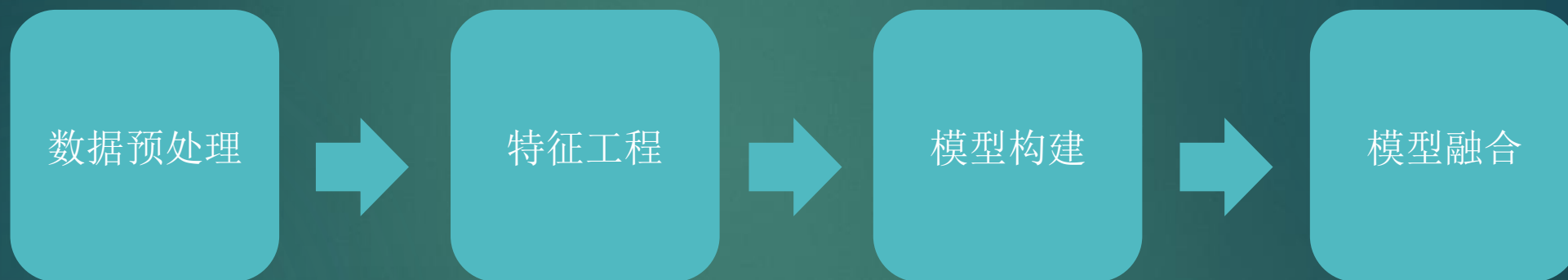


TNT_000

杨亚涛 黄春振 罗志鹏 赵伟

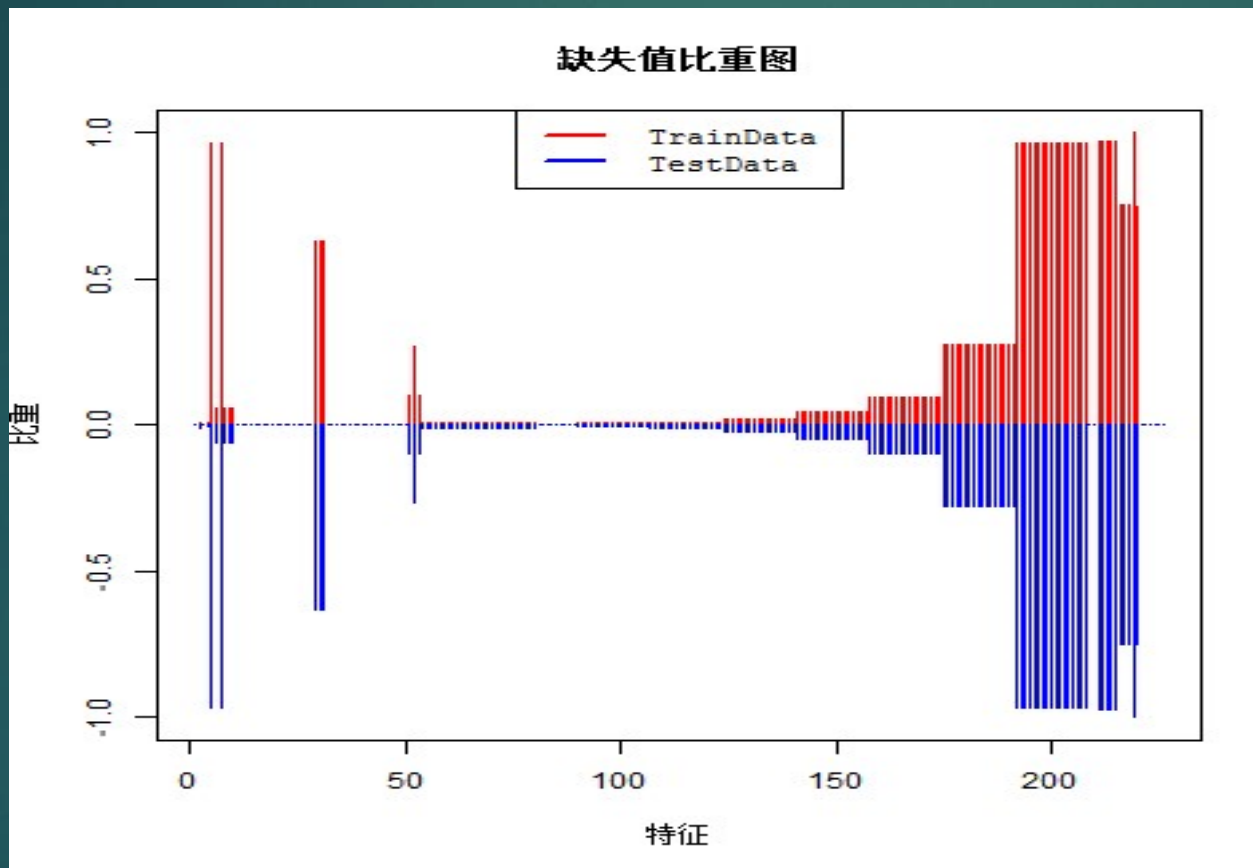
流程



数据预处理

3

● 缺失值处理



- 1) 缺失率低于0.2做缺失值填充, 数值型填充使用均值填充, 类别型使用众数填充。
- 2) 缺失率高于0.97做特征选择, 数值型缺失率很高的情况下, 更倾向于离散化, 在做特征选择的同时可以做下one-hot编码

ThirdParty_Info_Period7_10, [-1,19309], [0,480], [1,107], [2,48], [3,26], [4,15], [5,6], [6,4], [7,2], [8,2]↓

数据预处理

- 异常值处理 – 形式异常数据处理

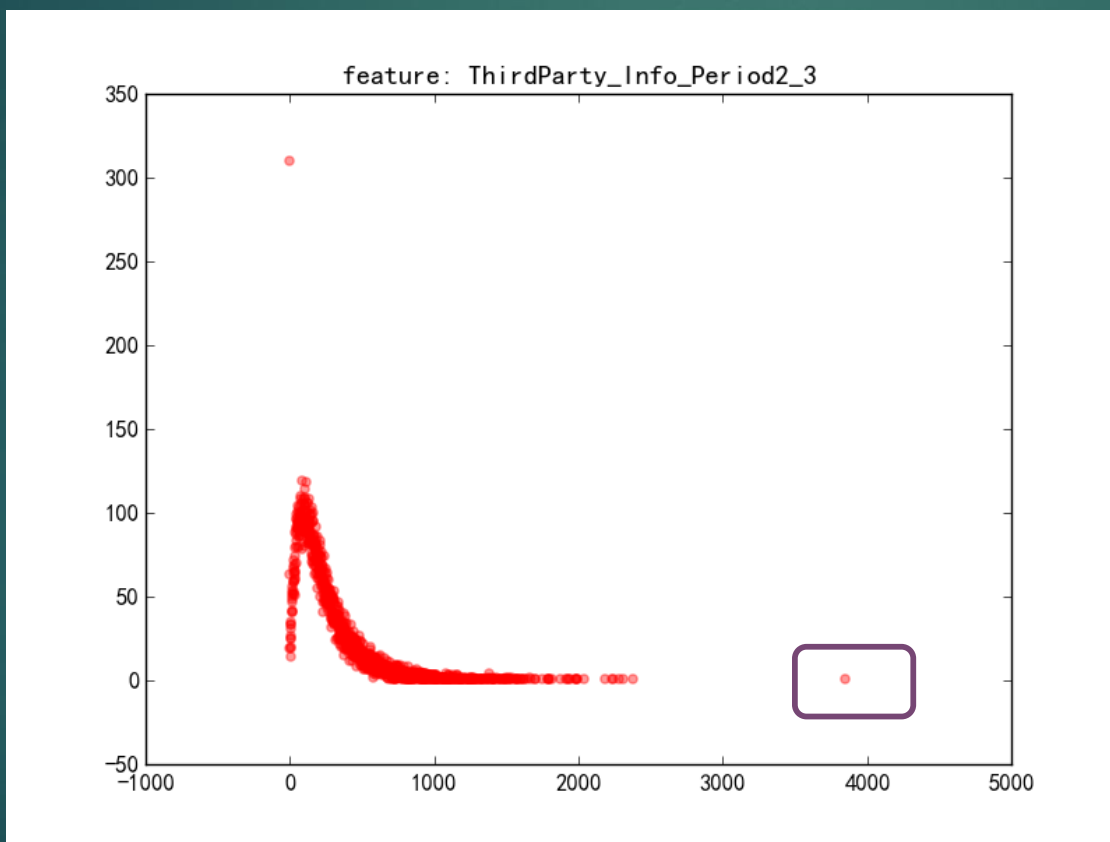
UserInfo_9, [不详,2792], [中国电信,823], [中国电信,1347], [中国移动,1151], [中国移动,10374], [中国联通,572], [中国联通,2940]

UserInfo_8, [七台河,5], [七台河市,3], [万州,20], [三明,82], [三明市,6], [三门峡,43], [三门峡市,3], [上海市,24], [上饶,42], [上饶市,9], [不详,2792],

数据预处理

5

- 异常值处理 – 数值异常数据处理



判断异常值标准:

$$x_{outlier} = \{x | x \geq \text{mean}(x) + 4 * \text{sd}(x)\}$$

异常值情况

注: X: 特征值
Y: 该特征值样本个数

特征工程

6

Master特征工程

LogInfo特征工程

Userupdate特征工程

特征工程-Master特征

7

1.Master_Numeric_Feature

2.Master_OneHot_Feature

3.Master_Location_Feature

4.Master_Weight_Feature

5.Master_TimeInfo_Feature


6.Master_Miss_feature

7. Master_ThirdParty_Feature

1.Master_Numeric_Feature

数值类型特征包含连续型的数值类型特征和类别型的数值类型特征

WeblogInfo_13,	Numerical↓
WeblogInfo_14,	Numerical↓
WeblogInfo_15,	Numerical↓
WeblogInfo_16,	Numerical↓
WeblogInfo_17,	Numerical↓
WeblogInfo_18,	Numerical↓
UserInfo_5,	Categorical↓
UserInfo_6,	Categorical↓
UserInfo_7,	Categorical↓
UserInfo_8,	Categorical↓
UserInfo_9,	Categorical↓
UserInfo_10,	Numerical↓
UserInfo_11,	Categorical↓
UserInfo_12,	Categorical↓
UserInfo_13,	Categorical↓
UserInfo_14,	Categorical↓
UserInfo_15,	Categorical↓
UserInfo_16,	Categorical↓
UserInfo_17,	Categorical↓
UserInfo_18,	Numerical↓



```
UserInfo_11, [0.0,10091], [1.0,1000], [nan,18909]
UserInfo_12, [0.0,7187], [1.0,3904], [nan,18909]
UserInfo_13, [0.0,3953], [1.0,7138], [nan,18909]
UserInfo_14, [0,246], [1,275], [2,5756], [3,11423], [4,6628], [5,4083], [6,1589]
UserInfo_15, [0,246], [1,266], [2,5698], [3,11473], [4,6615], [5,4130], [6,1572]
UserInfo_16, [0,2], [1,18026], [2,9412], [3,797], [4,735], [6,1028]
UserInfo_17, [1,26120], [2,3880]
```


2.Master_OneHot_Feature

9

待One-Hot编码操作的类型包含类别型（除去地址）特征和数值型特征集中值的类别不超过12个的特征

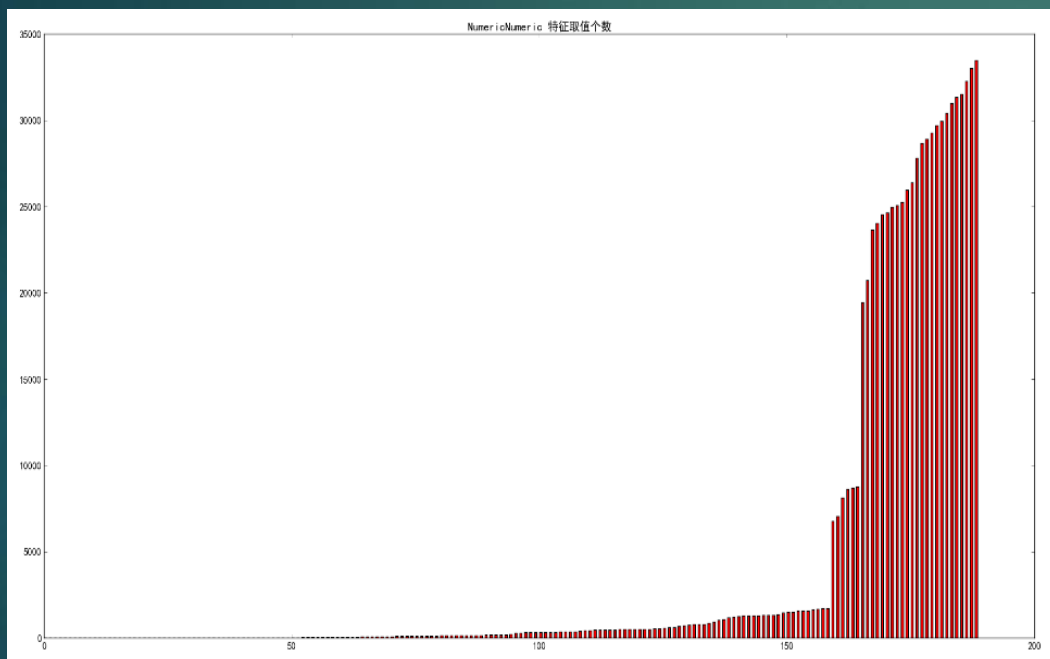


图1 数值型（Numeric）特征每一维取值的个数

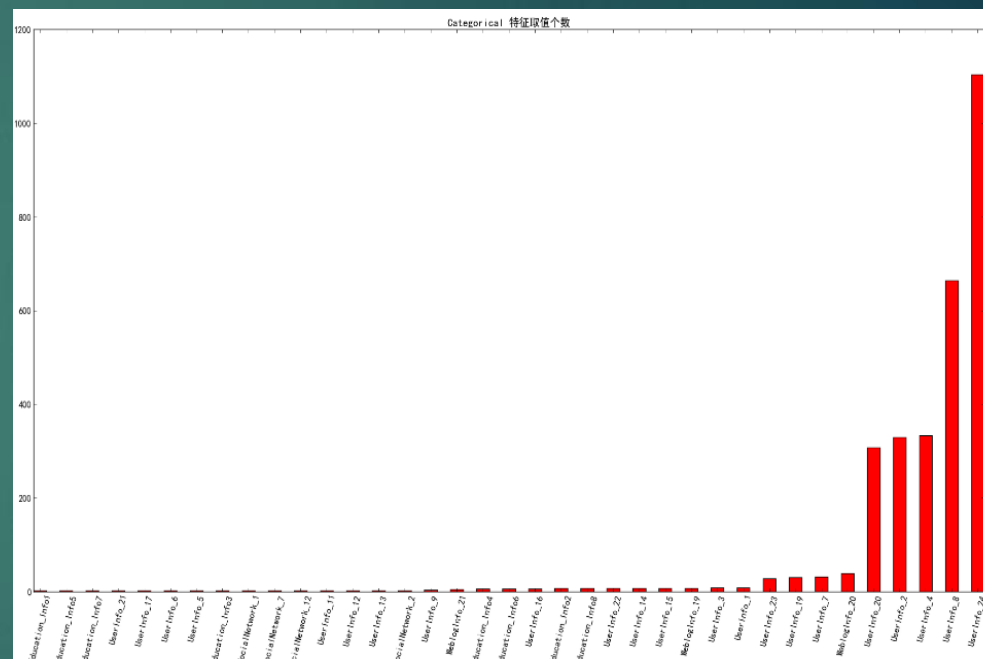
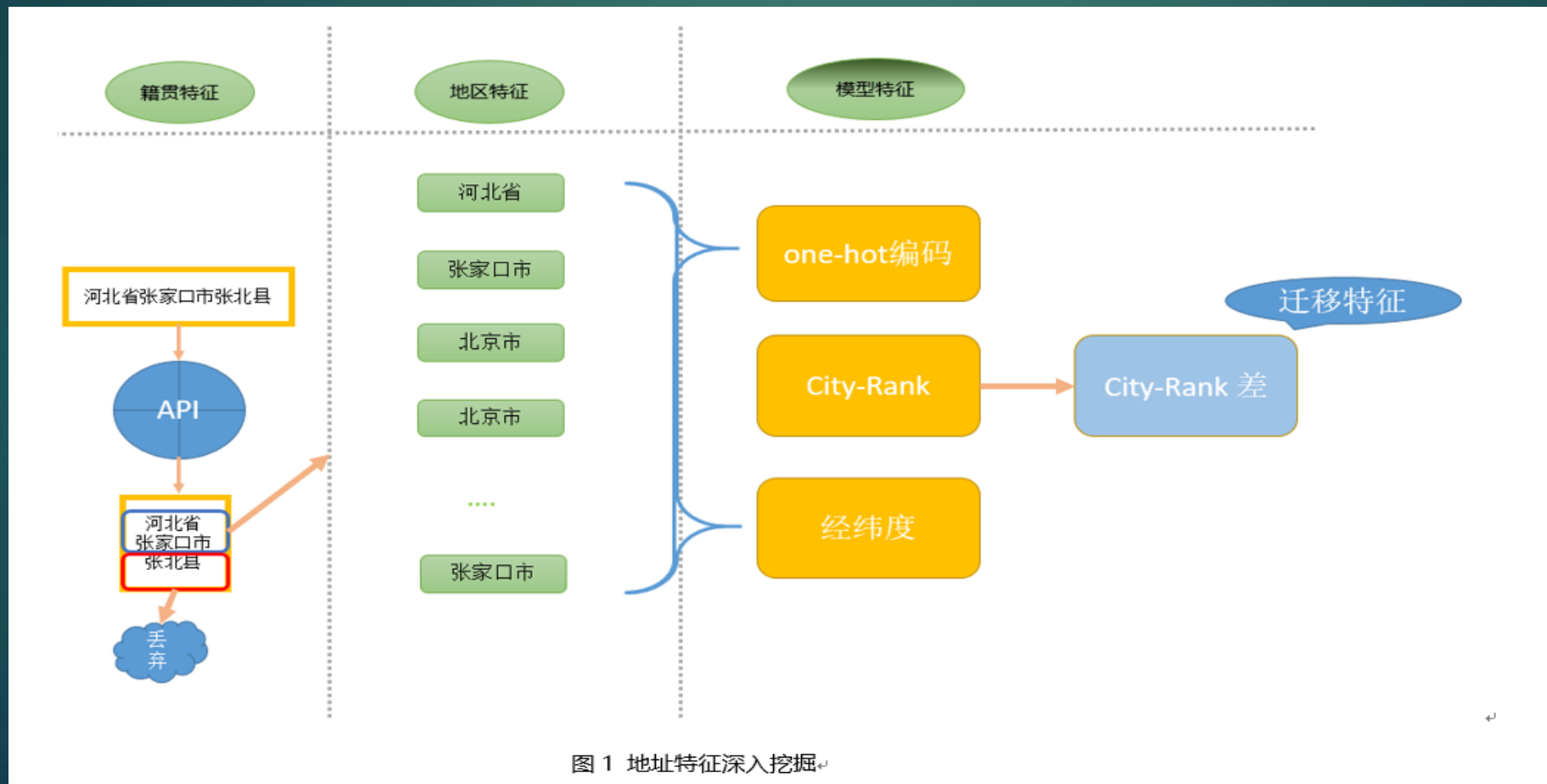


图2 类别型（Categorical）特征每一维取值的个数

3.Master_Location_Feature

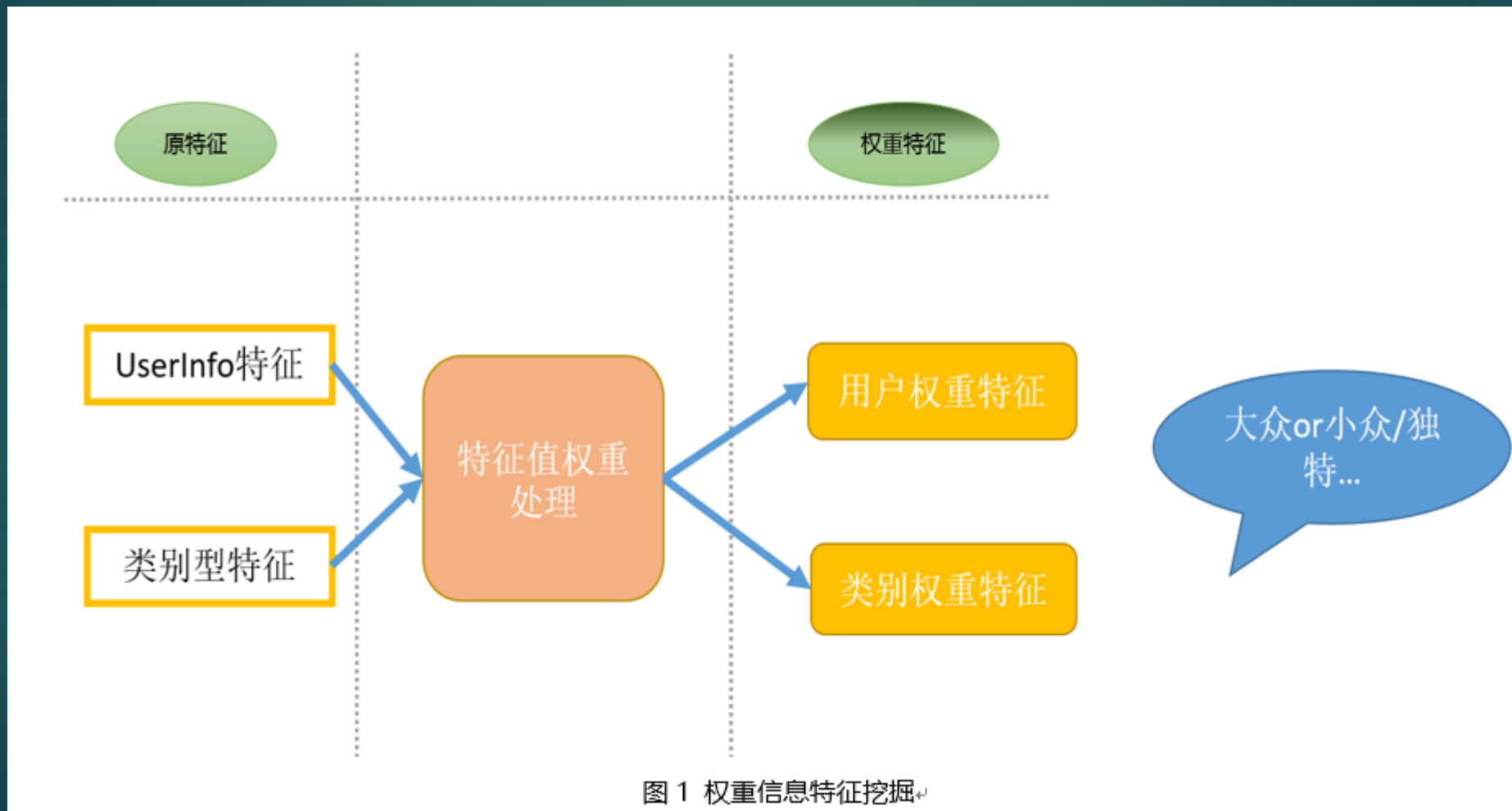
Master 中含有location信息的特征，涉及到地址的特征维度有：UserInfo_2, UserInfo_7, UserInfo_4, UserInfo_8, UserInfo_9, UserInfo_24, UserInfo_20, UserInfo_22, UserInfo_19。



4.Master_Weight_Feature

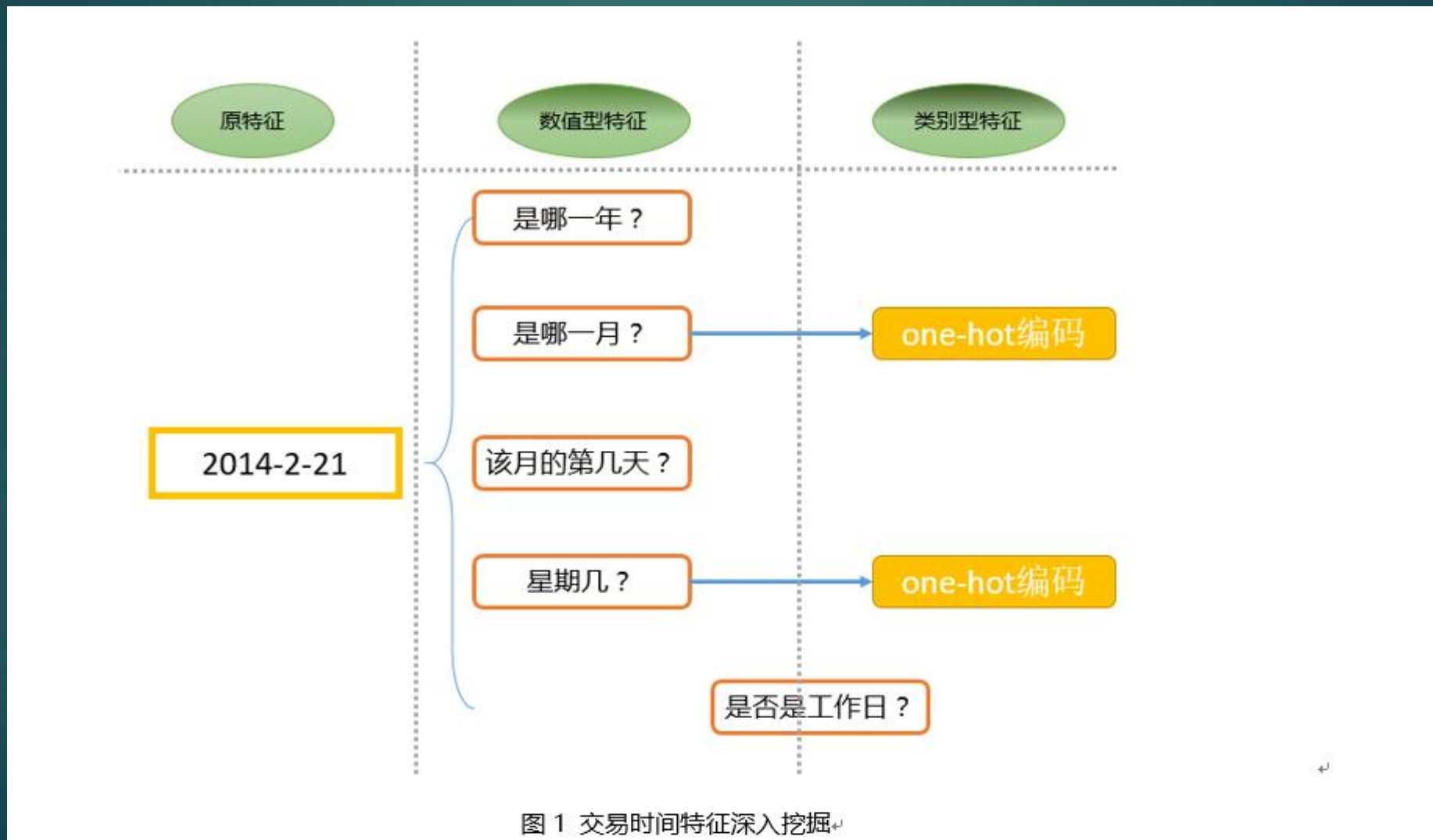
11

Master 中我们对类别型和部分UserInfo的数值型特征进行了Weight类型特征提取，以一个特征为单位进行特征处理，计算出该列特征的每个取值的个数在此列特征中的比例，然后用比例信息代替特征原来的取值。



5.Master_TimeInfo_Feature

Master 中含有交易时间信息的特征，我们对交易时间做了丰富的特征映射。这样增强了模型对时间的识别能力。



6.Master_Miss_feature

从数据可知：“空白”、“-1”、“NULL”、“不详”都表示缺失值。

我对每一维特征取值为“空白”、“-1”、“NULL”和“不详”的值置为1，其他的取值置为0，获得用户的缺失值信息。

-1	“北京”	“移动”	“上海市浦东区”	1	0	0	0
3.6	“不详”	NULL	“深圳市南山区”	0	1	1	0
2.7	“上海”	“联通”	“”	0	0	0	1
-1	“广州”	NULL	“广州市番禺区”	1	0	1	0
2	“深圳”	“电信”	“北京市朝阳区”	0	0	0	0

图1 缺失信息深入挖掘

7.Master_ThirdParty_Feature

根据数值型特征中的ThirdParty_*、UserInfo_*等5个类型的特征独立进行XGBoost模型构建预测。我们发现第三方（ThirdParty_*）特征对模型的AUC起到关键的作用。

```
Numeric各个类型的数据重要性
ThridParty 119维 XGB600 max_depth=4
[990] cv-test-auc:0.7187414+0.0087207494311 cv-train-auc:0.8940278+0.0010865868396
UserInfo 2维
[40] cv-test-auc:0.5779192+0.00705863852028 cv-train-auc:0.5995334+0.00442177179873
WeblogInfo 54维
[350] cv-test-auc:0.6270612+0.0128781315632 cv-train-auc:0.6874062+0.00290154375462
SocialNetwork 13维
[250] cv-test-auc:0.5932028+0.0120215542323 cv-train-auc:0.6412292+0.00292207641242
LinstingInfo_transform
[60] cv-test-auc:0.5780908+0.00581688230584 cv-train-auc:0.6092664+0.00245686553153
```

图 1 不同类型特征在 XGBoost 训练下的 CV 值

7.Master_ThirdParty_Feature

15

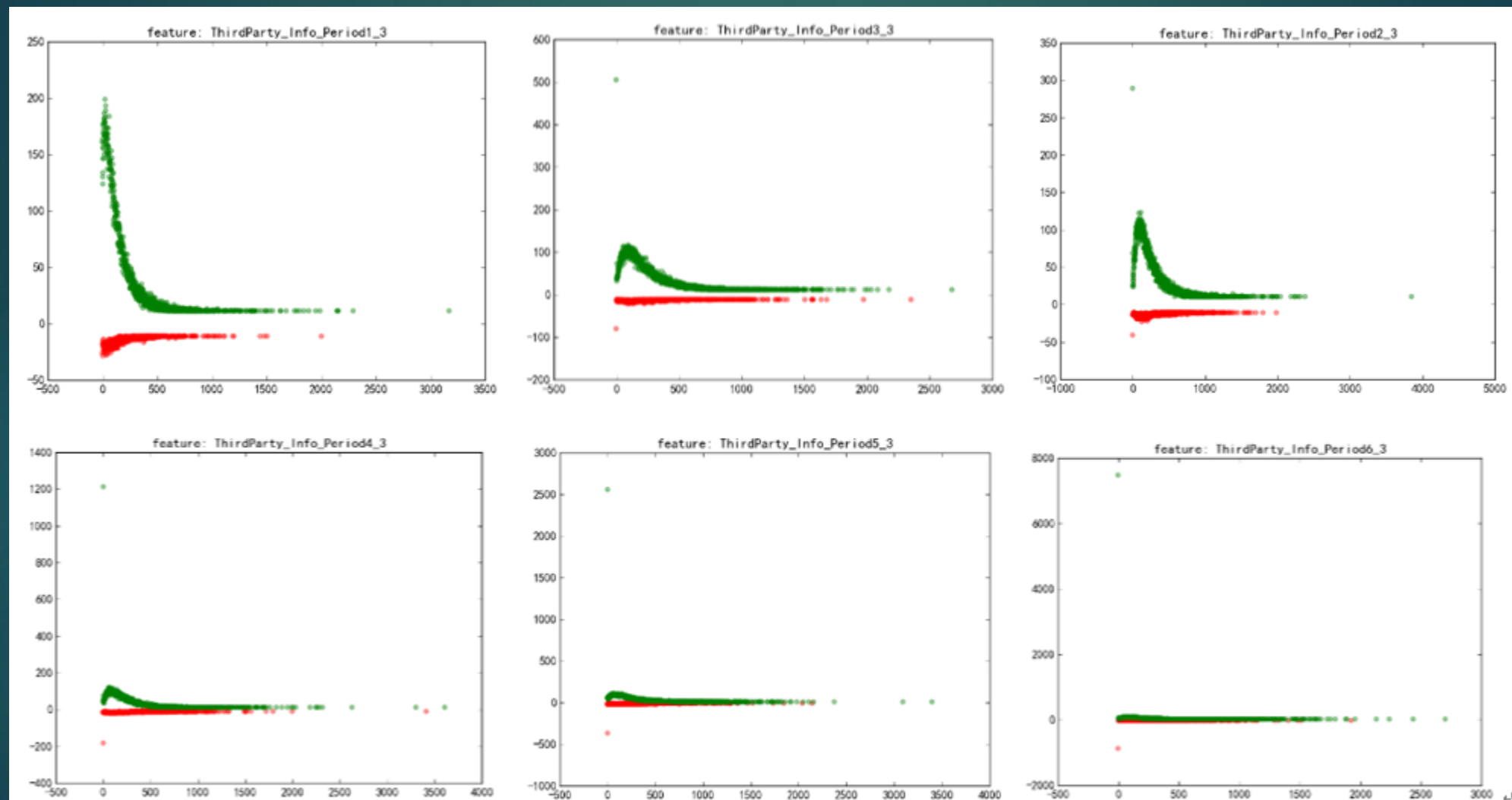
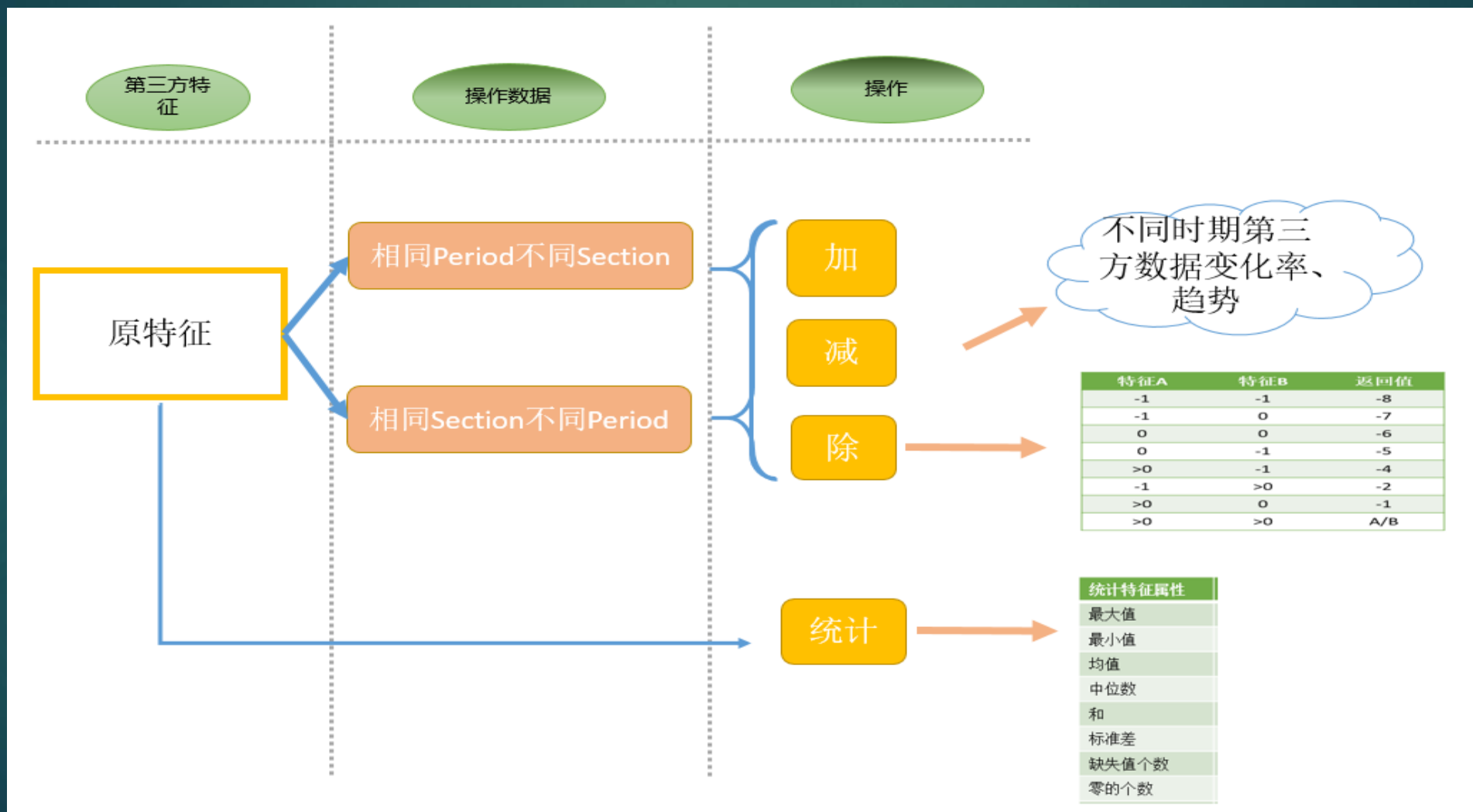


图 1 不同时期训练集中正样本与负样本在不同取值人数的变化举例

7.Master_ThirdParty_Feature

16



特征工程-LogInfo特征

17

1.LogInfo_Time_Info_Feature

2.LogInfo_Operational_Statistical_with_Time_Feature

1.LogInfo_Time_Info_Feature

我们对操作时间进行了以下十五维度的统计：

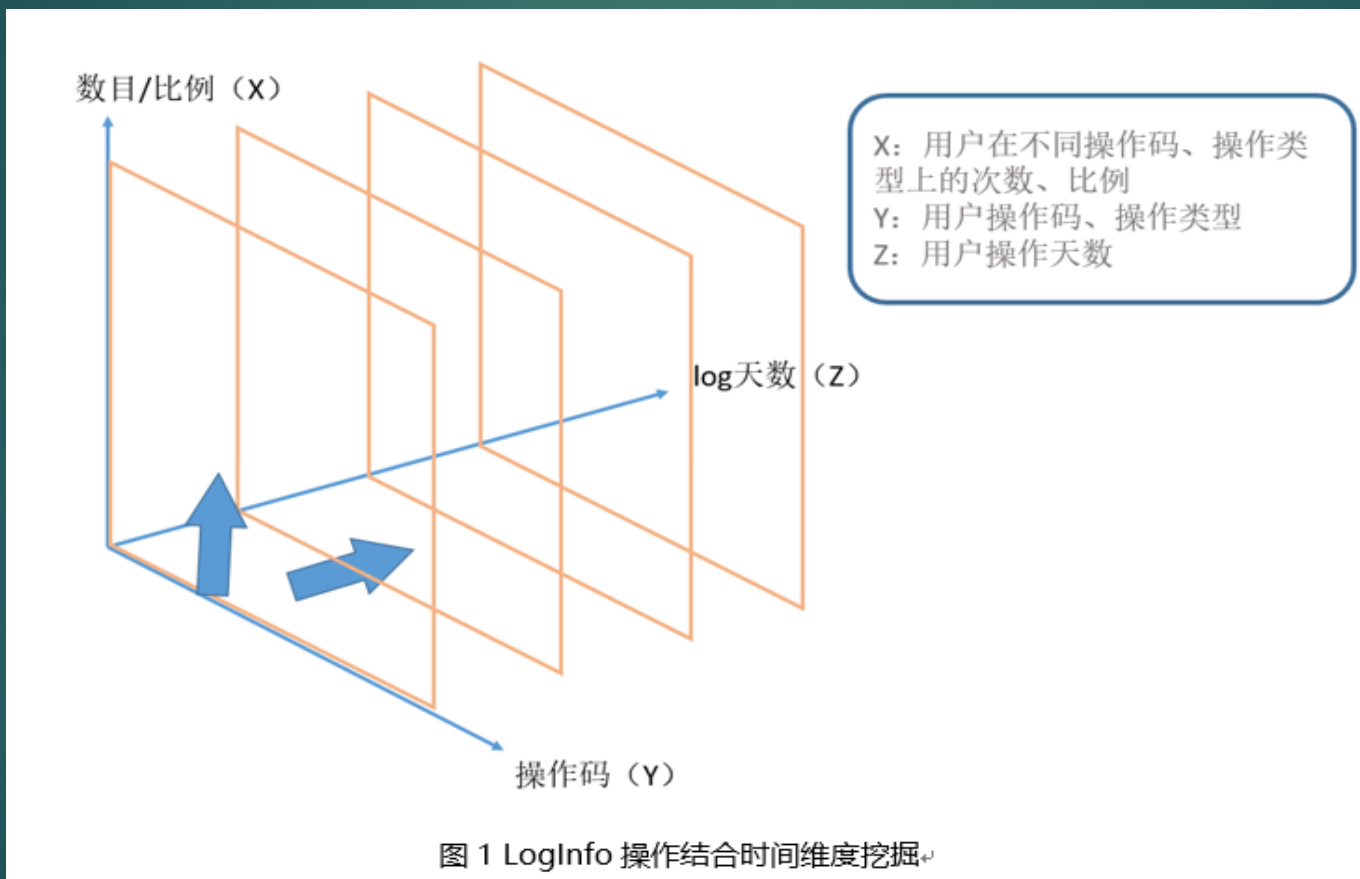
特征描述	维度
交易时间	1
交易时间-最后log时间	1
交易(当天, log最后1/3/5/7)数目、比例	10
用户活跃天数	1
用户log最后一周活跃天数、活跃比例	2

图 1 LogInfo 时间维度挖掘

2.LogInfo_Operational_Statistical_with_Time_Feature

19

我们从操作类型/数目/时间三个维度挖掘信息：



特征工程-Update特征

20

1.UserUpdate_Time_Info_Feature

2.UserUpdate_Operational_Statistical_with_Time_Feature

1.UserUpdate_Time_Info_Feature

我们依旧对操作时间进行了以下十五维度的统计：

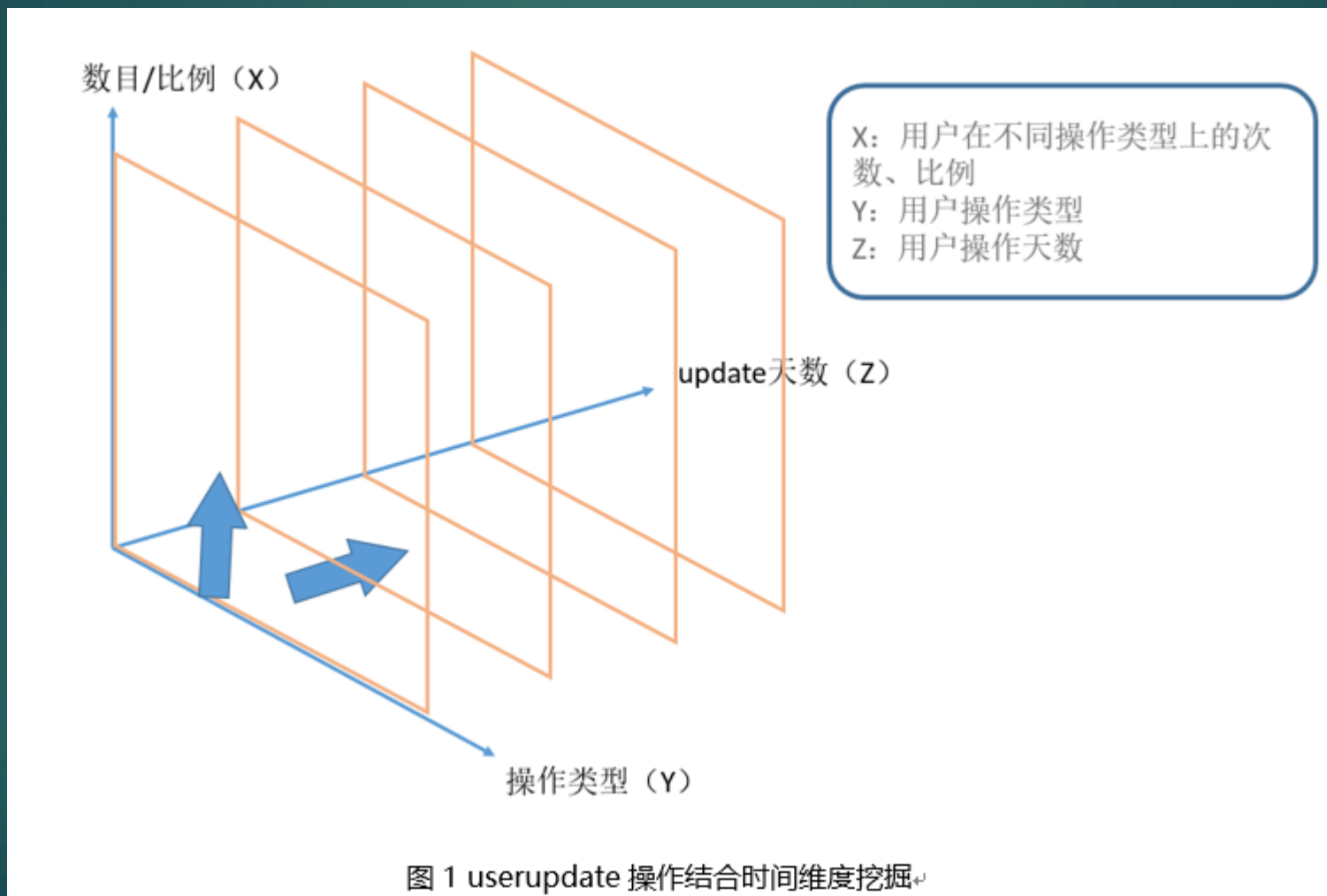
特征描述	维度
交易时间	1
交易时间-最后update时间	1
交易(当天, update最后1/3/5/7)数目、比例	10
用户活跃天数	1
用户update最后一周活跃天数、活跃比例	2

图 1 LogInfo 时间维度挖掘

2.UserUpdate_Operational_Statistical_with_Time_Feature

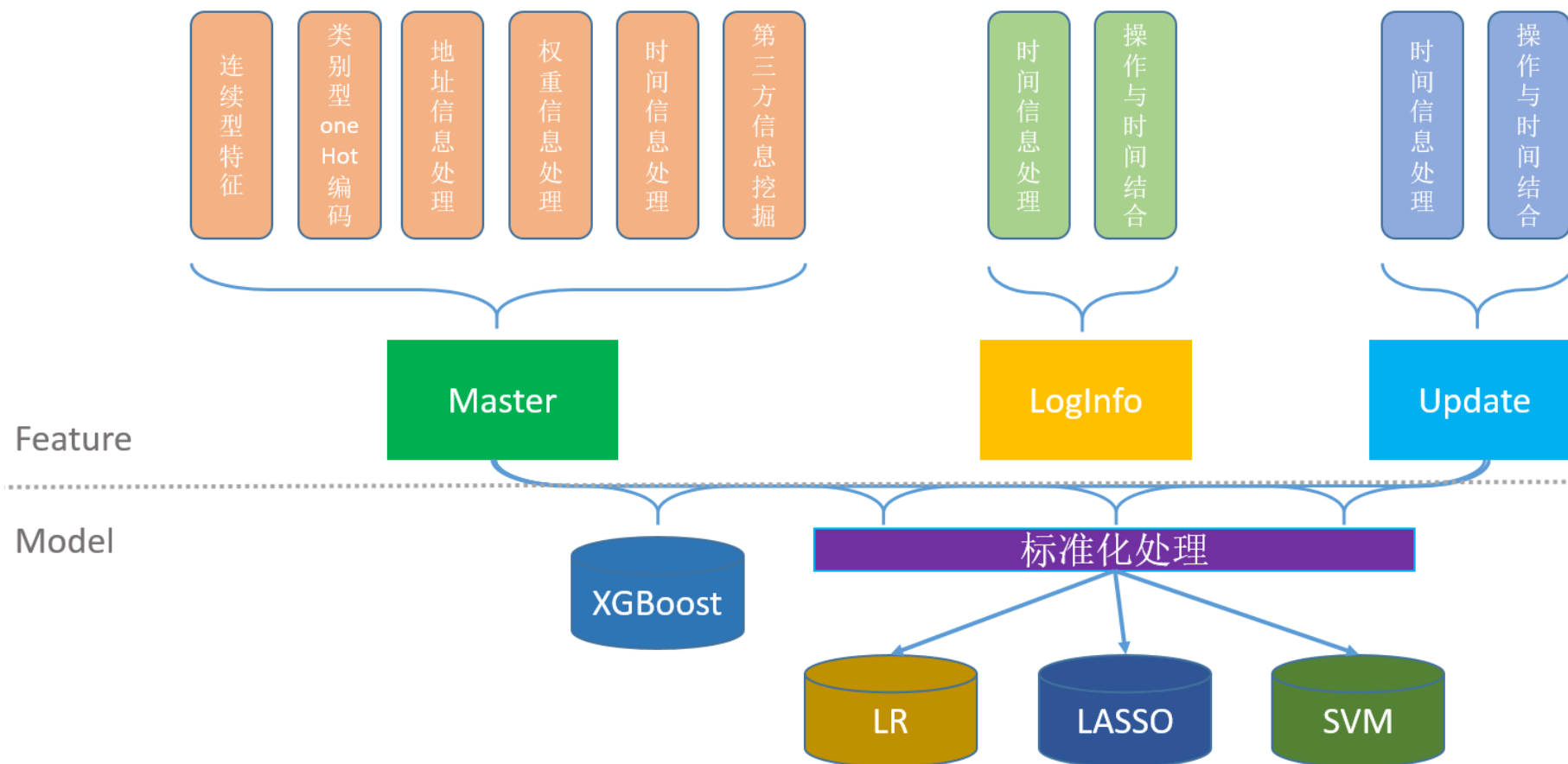
22

我们依旧从操作类型/数目/时间三个维度挖掘信息：



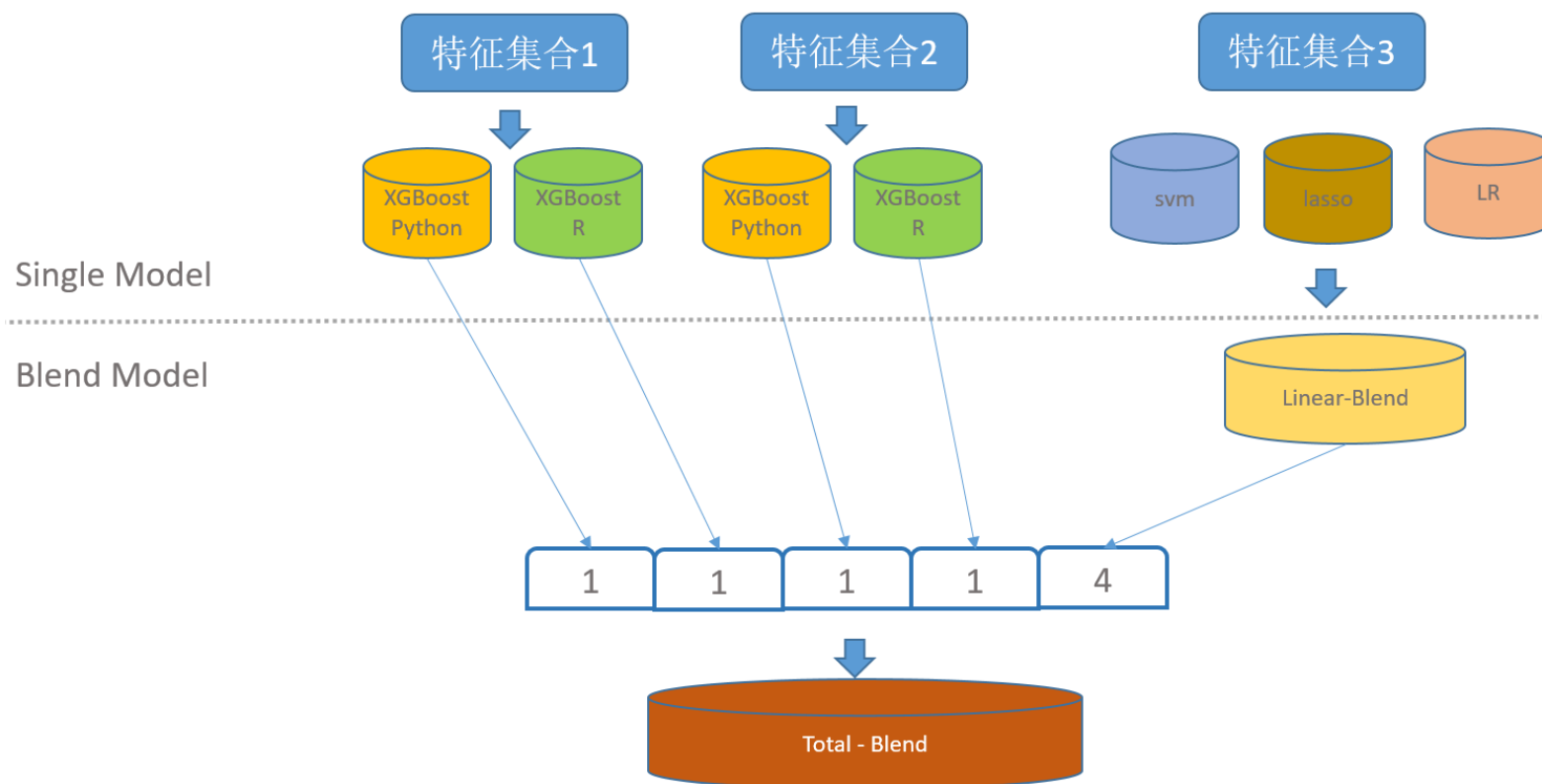
模型构建

23



模型融合

24



模型结果

25

模型	CV-9 折结果
XGB-4637 Python	0.7839
XGB-3952 Python	0.7857
XGB-3952 R	0.7852
XGB-4637 R	0.7845
LASSO	0.7745
LR	0.7752
Linear-SVM	0.7734
Linear-ensemble	0.7785
All-Blend-AUC	0.7896

模型结果

我们第一次的提交结果就突破了0.79，也是第一支破0.79的队伍

比赛阶段	作品名称	文件名称	提交时间	评审状态/得分
复赛	结果文件	4-16-1.csv	2016-04-17 01:43:18	0.790434595511
初赛	结果文件	3-31-max.csv	2016-03-31 22:48:59	0.773049188825
初赛	作品文件	src.zip	2016-03-31 19:36:06	已评审

Thanks

这次赛题主要按照下面步骤进行：

1. 数据预处理

数据预处理主要是对数据中出现的缺失值及异常值进行处理。

a. 缺失值

缺失值的信息广泛存在于数据集之中。缺失值的出现形式是数值型的-1 以及字符串的空字符串。首先我们统计了缺失值在 Master 数据集中每个特征占有比的分布。

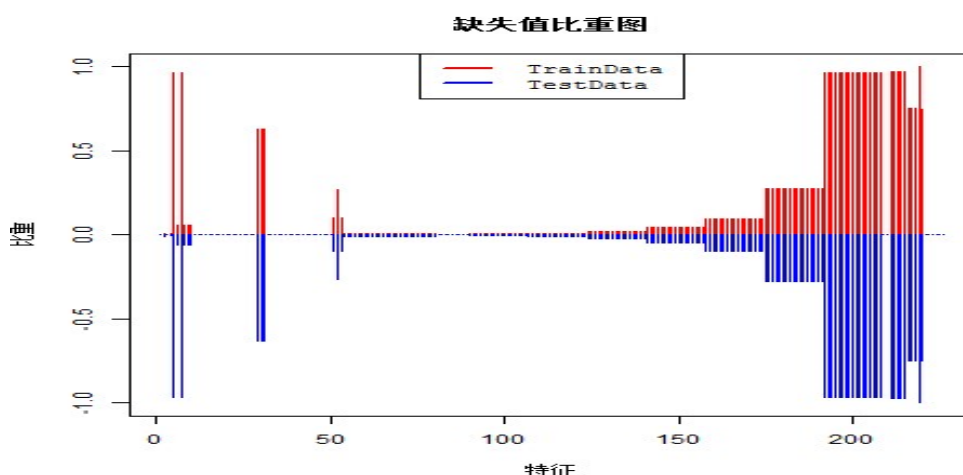


图 1 缺失比重信息

从上图我们可以看出来，每个特征的缺失值占有率不近相同，但是训练集与测试集的缺失值比重还是比较相近的。有些缺失占有率很高甚至接近于 1，有些缺失占有率很低接近于 0。我们采取的策略是缺失值比重低的进行特征填充，缺失值比重高的进行特征选择。另外为了保持其自身缺失的信息，我们又进行构造了对每个样本的特征缺失统计特征。

- (1) 缺失值占有率在 0.2 以下的进行特征填充，填充的值我们试过除缺失值后整体数据的均值、中位数、众数。最终我们使用的方法是：对于数值型特征，填充均值。对于类别型特征填充众数。发现这样的填充比例和填充方式是最优的。
- (2) 缺失值占有率在 97%以上的我们进行的是去除操作和 one-hot 编码转换操作，去除操作是去掉缺失占有率超过 97%的特征。我们发现缺失值占有率很多的情况下，该特征更倾向于离散化。所以，我们不仅进行了去除操作，还进行了 one-hot 编码。

b. 异常值

异常值的信息在数据之中分为两种类型，第一种是形式上的异常，第二种是数据表现上的异常。

1) 形式上的异常主要在类别型中出现：

比如说，“中国移动”与“中国移动 ”是一个性质的类别。但是后面的多出现了一个空格，就造成了两种类别。“山东省”和“山东”是一个地理位置表达。但是在形式上是两种类型。并且我们发现这种形式上的异常有不少，所以我们首先对这些形式上的异常进行了处理。将这些一个意思两种表达的变量给统一化。

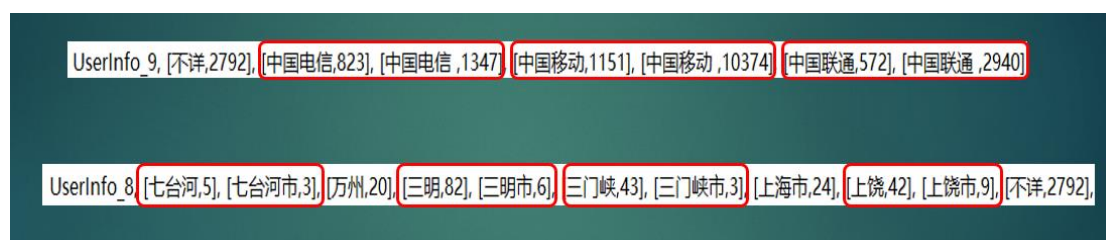


图 2 形式上的异常

2) 数据表现上的异常主要在数值型中出现：

比如说，在一个数值变量中分布大部分都在[10,300]区间内，突然出现了几个几万的数据值。这些值被我们称为异常值。当然，我们定义的异常值是指超过该特征总体分布的均值加 6 倍标准差以外的值(检验得出 6 倍的标准差是最优依据)。我们对均值加 6 倍标准差以外的值进行异常值处理。处理的方式有变为均值加 6 倍的标准差、均值、中位数的处理。最终，我们采用的替换值是均值加六倍的标准差。

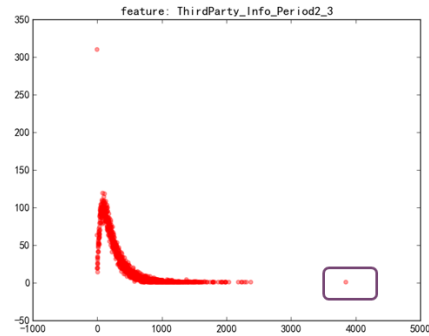


图 3 数据上的异常

2. 特征工程

基于上面数据的预处理之后 我们构建了丰富的特征工程。特征工程包含三大部分 ,它们分别是 Master 数据集的特征、LogInfo 数据集的特征及 UserUpdate 数据集的特征。

Master 数据集的特征

1.Master Numeric 类型特征：

Master 数值类型特征包含连续型的数值类型特征和类别型的数值类型特征。

```
UserInfo_11, [0.0,10091], [1.0,1000], [nan,18909]
UserInfo_12, [0.0,7187], [1.0,3904], [nan,18909]
UserInfo_13, [0.0,3953], [1.0,7138], [nan,18909]
UserInfo_14, [0,246], [1,275], [2,5756], [3,11423], [4,6628], [5,4083], [6,1589]
UserInfo_15, [0,246], [1,266], [2,5698], [3,11473], [4,6615], [5,4130], [6,1572]
UserInfo_16, [0,2], [1,18026], [2,9412], [3,797], [4,735], [6,1028]
UserInfo_17, [1,26120], [2,3880]
```

图 4 类别型特征中数值型取值的特征举例

从上图我们可以看出，一些类别型特征本身的取值为数值型的。如上图：类别型特征 UserInfo_11~UserInfo_17，这些特征在官方给出的文档中显示为类别型的特征。而其实其取值为数值型的。将这一类可转换为数值型的特征，统一转换为数值型的形式。而我们在之后的模型构建中发现，这类特征转换对模型拟合也是非常有用的。

2. Master One-Hot 编码类型特征：

Master One-Hot 编码类型特征包含类别型特征和数值型特征集中值的类别不超过 12 个的特征。对这些特征进行 one-hot 的编码。one-hot 编码常用于处理类别型数据，因为分类器不能很好地处理没有次序的特征，如地址、性别等信息，而 one-hot 编码使用 N 个状态维度来对 N 个状态进行编码，每个状态都由他独立的寄存器位，并在任意时候，只有一个位有效。而我们从下图 1 的数值型与图 2 类别型的每个维特征取值的个数可知。数值型特征中，有一些特征的取值个数非常少，这一类特征经观察知这类特征某个值的用户数量非常多，而其他取值的用户数量非常少。那么将这类特征作 one-hot 编码也可以增加分类器对特征的识别能力，但在作编码的同时也保留原来的次序特征。同样，在类别型特征中，我们发现一些地址特征（如用户的详细地址）取值非常多，达到 3000 个以上。如果对这些地址直接进行 one-hot 编码，则会使整个特征维度大大提升，同时编码后矩阵也极为稀疏，所以我们对这类的地址进行独立的处理。

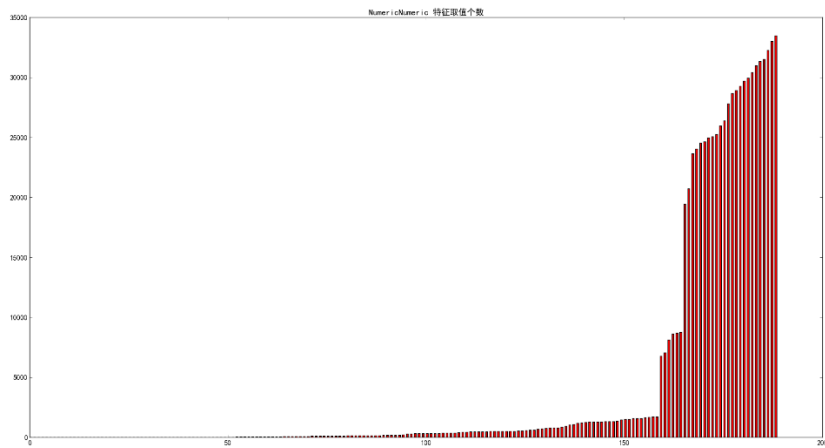


图 5 数值型（Numeric）特征每一维取值的个数

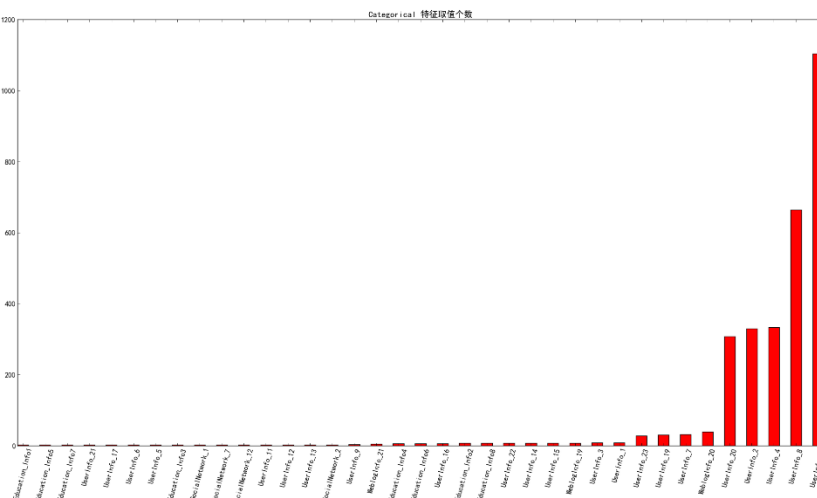


图 6 类别型（Categorical）特征每一维取值的个数

3. Master Location 类型特征：

Master 中含有 location 信息的特征，涉及到地址的特征维度有：UserInfo_2, UserInfo_7, UserInfo_4, UserInfo_8, UserInfo_9, UserInfo_24, UserInfo_20, UserInfo_22, UserInfo_19。由于地址一些地址信息为用户的详细地址，粒度太小，所以要将这些信息扩大为省份与城市的级别，去除不必要的噪声。我们采取百度地图 API，百度地图 API 可以将用户输入的地址转换的响应的字段信息，比如：国家、省份、城市、街区。我们在地址转换的过程中，只选取省份与城市两个级别的信息。将地址转换为相应的省份与城市后，再将这些信息进行 one-hot 编码。这样，进行编码的维度从 3000 维可以下降到 1000 维左右，这样可以减少模型的复杂度与噪声。根据《第一财经周刊》发布的 2015 城市等级数据，中国的地级以上城市被划分为 6 个等级。其中，等级越高，城市个数越少。第一等级只有北京、上海、广州和深圳 4 个城市。我们根据上面得到的地址对应的城市，将城市的信息转为为相应的城市。同时，我们对不同地址维度的城市等级进行两两相减，表示某个用户在不同维度的城市迁移。比如某个维度是代表用户的籍贯，另外一维代表用户的目前居住城市。与地址信息转换为省份与城市相对应，百度地图 API 可以将用户输入的地址转换为该地址的经纬度，我们利用这个便利的 API 将每一维的地址信息转为为 2 个维度的经度和纬度。

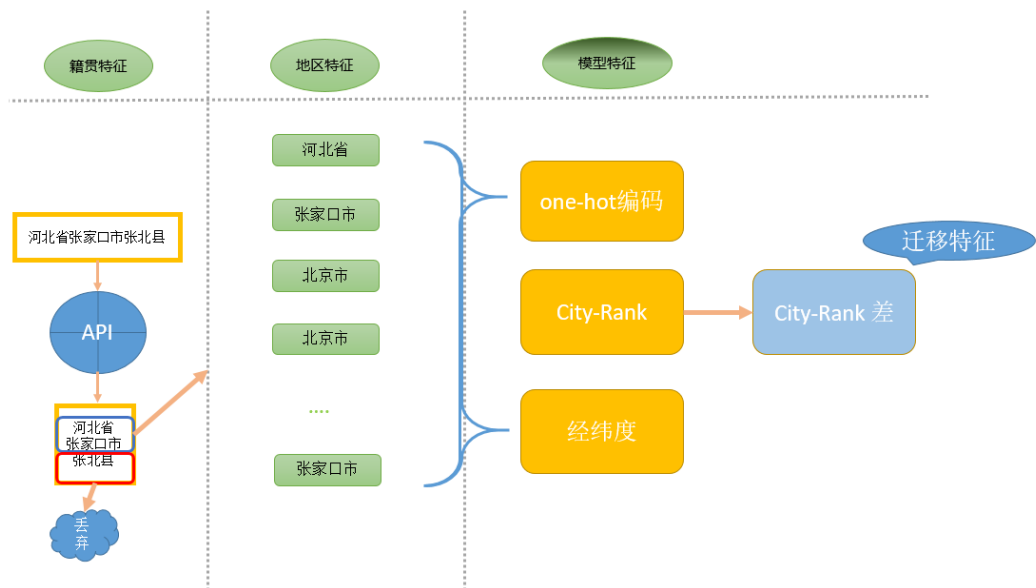


图 7 地址特征深入挖掘

4. Master Weight 类型特征：

Master 中我们对类别型和部分 UserInfo 的数值型特征进行了 Weight 类型特征提取，以一个特征为单位进行特征处理，计算出该列特征的每个取值的个数在此列特征中的比例，然后用比例信息代替特征原来的取值。这样做特征处理的意义在于表明特征某个取值的大众性或独有性。用户信息权重特征，是挑选出与“UserInfo_*”开头的特征作权重特征处理。类别型权重特征是基于挑选出官方给出的类别型特征做权重特征处理

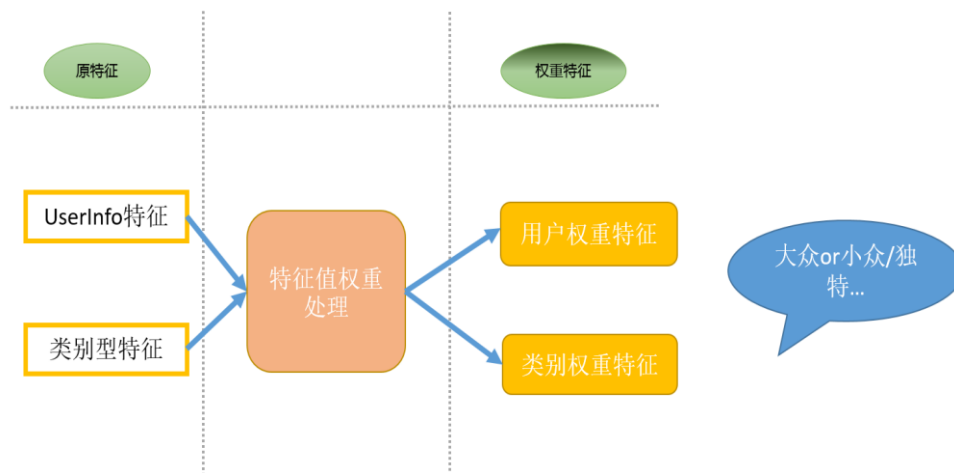


图 8 权重信息特征挖掘

5. Master Time Info Transform 类型特征：

Master 中含有交易时间信息的特征，我们对交易时间做了丰富的特征映射。这样增强了模型对时间的识别能力。

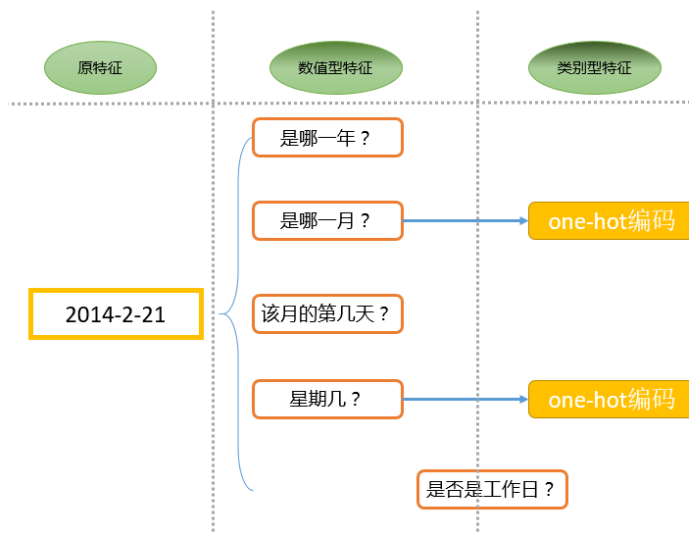


图 9 交易时间特征深入挖掘

从上图我们可以看出，原先的时间特征几乎没办法去利用，只有对其进行转换切分才能深入挖掘其中的信息。我们做的主要有两种类型的转换特征。数值型特征：这一交易时间发生的年份/月份/第几天/星期几/节假日。类别型特征：这一交易时间月份/星期的 one-hot 编码特征。

6. Master Miss 类型特征：

比赛论坛中说明“空白”、“-1”、“NULL”都表示缺失值。而在 master 原始特征统计中发现，类别型信息中包含“不详”等值，我们也视为缺失值。不同维度的特征缺失个数不同。所以统计用户在某个特征维度中的取值是否为缺失值是非常重要的。我对每一维特征取值为“空白”、“-1”、“NULL”和“不详”的值置为 1，其他的取值置为 0，获得用户的缺失值信息。

-1	“北京”	“移动”	“上海市浦东区”	1	0	0	0
3.6	“不详”	NULL	“深圳市南山区”	0	1	1	0
2.7	“上海”	“联通”	“”	0	0	0	1
-1	“广州”	NULL	“广州市番禺区”	1	0	1	0
2	“深圳”	“电信”	“北京市朝阳区”	0	0	0	0

图 10 缺失信息深入挖掘

7. Master ThirdParty Period 类型特征：

Master 中含有第三方信息的特征，根据数值型特征中的 ThirdParty_*、UserInfo_*等 5 个类型的特征独立进行 XGBoost 模型构建预测。在图 CV-10 折结果中可以看到，第三方 (ThirdParty_*) 特征对模型的 AUC 起到关键的作用。

Numeric各个类型的数据重要性
ThridParty 119维 XGB600 max_depth=4
[990] cv-test-auc:0.7187414+0.0087207494311 cv-train-auc:0.8940278+0.0010865868396
UserInfo 2维
[40] cv-test-auc:0.5779192+0.00705863852028 cv-train-auc:0.5995334+0.00442177179873
WeblogInfo 54维
[350] cv-test-auc:0.6270612+0.0128781315632 cv-train-auc:0.6874062+0.00290154375462
SocialNetwork 13维
[250] cv-test-auc:0.5932028+0.0120215542323 cv-train-auc:0.6412292+0.00292207641242
LinstingInfo_transform
[60] cv-test-auc:0.5780908+0.00581688230584 cv-train-auc:0.6092664+0.00245686553153

图 11 不同类型特征在 XGBoost 训练下的 CV 值

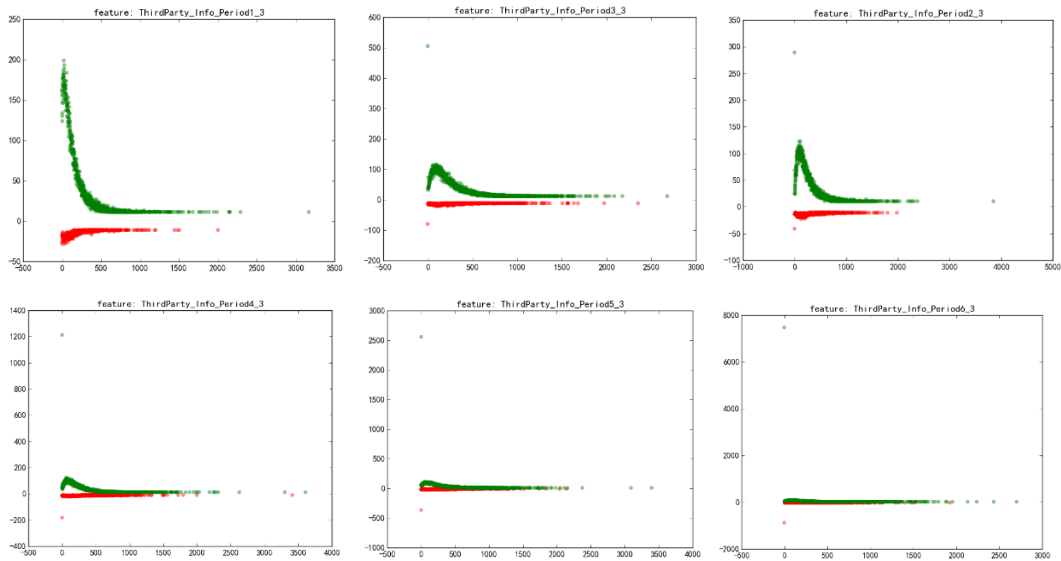


图 12 不同时期训练集中正样本与负样本在不同取值人数的变化举例

根据官方给出第三方特征共有 11 个特征。其中，特征分为 7 个时期（Period），每个时期中有 17 个时间段（Section）的取值。可以分别对比：

不同时期（Period）、相同时间段（Section）的特征进行对比；

相同时期（Period）、不同时间段（Section）的特征进行对比；

对比的形式有：

1. 对两两特征列进行相减处理，表示不同时期或不同时间段的特征差值变换；
2. 对两两特征列进行相除处理，表示不同时期或不同时间段特征的环比（斜率）变换；
3. 对相同时期或者相同阶段的特征进行累积处理，表示相同时期或阶段的和值变化；
4. 对相同时期或者相同阶段的特征进行统计处理，其中，统计的量有：最小值，最大值，中位数，均值，缺失值个数，值为 0 的个数，值大于 0 的个数，所有值相加的个数。

其中，对于除法运算，由于一些列有缺失值，不能对该列特征直接除。我们采取以下方式对不能直接除的情况进行处理：

特征A	特征B	返回值
-1	-1	-8
-1	0	-7
0	0	-6
0	-1	-5
>0	-1	-4
-1	>0	-2
>0	0	-1
>0	>0	A/B

图 13 特征相除条件判断

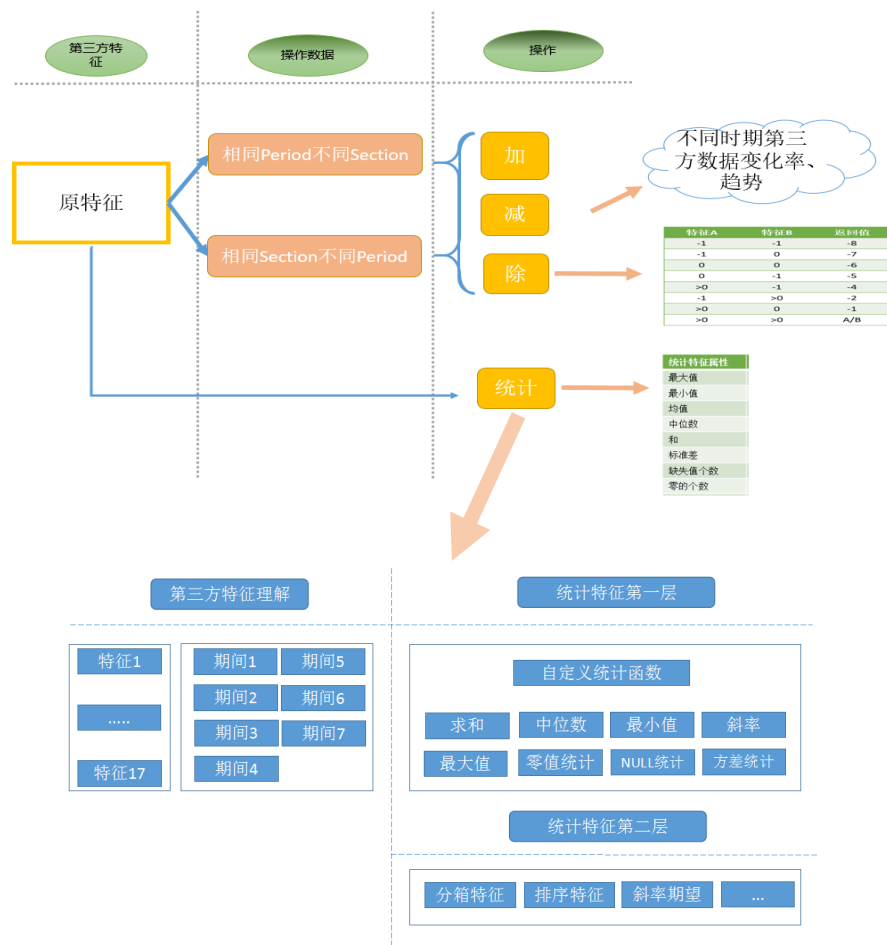


图 14 第三方特征深度挖掘

LogInfo 数据集的特征

LogInfo 中共有 5 个字段，其中 LogInfo1 表示操作代码，LogInfo2 表示操作类别，LogInfo3 表示操作时间。我们团队对 LogInfo 的特征提取分为三种类型。

1. LogInfo Time Info 类型特征：

我们对操作时间进行了以下十五维度的统计：

特征描述	维度
交易时间	1
交易时间-最后log时间	1
交易(当天, log最后1/3/5/7)数目、比例	10
用户活跃天数	1
用户log最后一周活跃天数、活跃比例	2

图 15 LogInfo 时间维度挖掘

2. LogInfo Operational Statistical with Time 类型特征：

由统计可知，所有用户的操作代码共有 36 种类型，而操作类别有 16 种类型。计算每个用户在不同类别操作代码的次数、操作比例；计算每

个用户在不同操作类别的次数、操作比例；用户 UserInfo log 的数目。同时，由于观察到一些用户产生 log 的数据跨越的时间较长，比如用户第一条 log 在 2013 年 5 月，而最后的交易时间是在 2014 年 2 月，一些年代久远的 log 对用户当前的交易影响不大。所以，我们还对用户的 log 进行时间维度的扩展，也就是加上时间限制。对上面得到的统计特征，分别用户最后 1 天，最后 3 天，最后 7 天进行相同的统计，从而描述用户 UserInfo 在不同时间维度的信息。

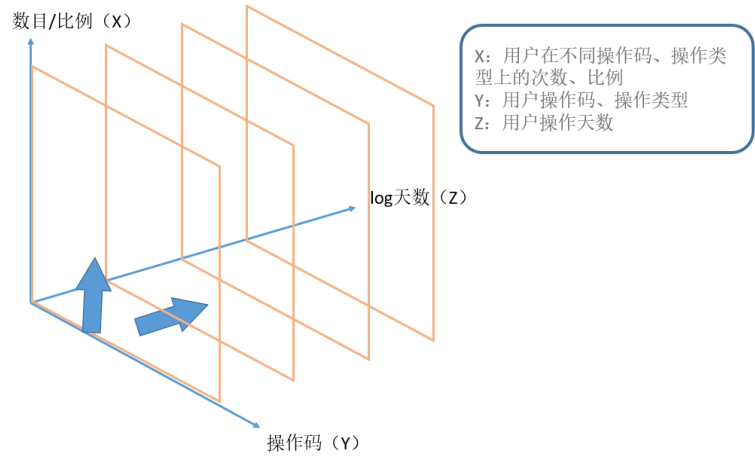


图 16 LogInfo 操作结合时间维度挖掘

UserUpdate 数据集的特征

UserUpdate 中共有 4 个字段，其中 UserupdateInfo1 表示操作代码，UserupdateInfo2 操作时间。我们团队对 Userupdate 的特征提取同样分为两种类型。

1. UserUpdate Time Info 类型特征：

我们对操作时间进行了以下十五维度的统计：

特征描述	维度
交易时间	1
交易时间-最后update时间	1
交易(当天, update最后1/3/5/7)数目、比例	10
用户活跃天数	1
用户update最后一周活跃天数、活跃比例	2

图 17 LogInfo 时间维度挖掘

2. UserUpdate Operational Statistical with Time 类型特征：

由统计可知，所有用户的 update 的操作类别有 55 种类型。计算每个用户在不同类别操作类别的次数、操作比例；计算每个用户在不同操作类别的次数、操作比例；用户 update 的数目。同时相同于 loginfo 的特征提取，我们还对用户的 update 进行时间维度的扩展，也就是加上时间限制。对上面得到的统计特征，分别用户最后 1 天，最后 3 天，最后 7 天进行相同的统计，从而描述用户 update 在不同时间维度的信息。

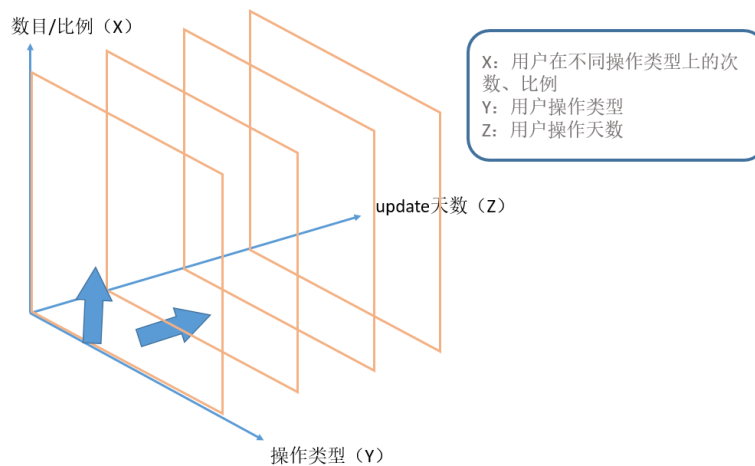


图 18 userupdate 操作结合时间维度挖掘

3. 模型构建

1. XGBoost

XGBoost 模型是一个非线性分类器，是渐进树模型的一种。Boosting 分类器属于集成学习模型，它基本思想是把成百上千的分类准确率较低的树模型组合起来，成为一个准确率很高的模型。这个模型会不断地迭代，每次迭代就生成一颗新的树。传统的渐进树代表是 GBDT（梯度泊松渐进树），GBDT 是一个加性回归模型，通过 boosting 迭代并构造一组弱分类器，相对 LR 的优势如不需要去做关于特征的归一化，自动进行特征选择，模型可解释性较好，可以适应多种损失函数如 Square Loss, Log Loss 等等。不同于传统的 GBDT 方式，只利用了一阶的导数信息。XGBoost 对 Loss function 做了二阶的泰勒展开，并在目标函数之外加入了正则项约束就能整体求最优解，用以权衡目标函数的下降和模型的复杂程度，避免过拟合。除了上述优点以外，XGBoost 还有速度快，可移植，少写代码，兼容错的优点。

2. Logistics Regression

逻辑回归（Logistic Regression）是机器学习中的一种线性分类模型，也是一种广义的线性回归分析模型。由于算法的简单和高效，在实际中应用非常广泛、快速、可以承载大数据量、可以有效处理离散化过的连续值数据、特征自我解释性等。还有线下训练出来权重词表，线上加载即可。有些模型可以很好的拟合数据，但解释度几乎为 0。有些模型可以很好的解释模型的输出，但拟合度较低。而 LR 是在数据的拟合度和模型解释度都能兼顾，而且兼顾得比较好的算法。

3. LASSO

Lasso 全称（least absolute shrinkage and selection operator）是 Robert Tibshirani 基于 Leo Breiman's Nonnegative Garrote 提出的改进算法，本质也是线性分类器的一种，不过它集成了特征选择以及正则化的功能，提高了统计模型的准确率和可解释性。类似的还有 Ridge 回归，它们用于解决不同的问题，之后被整合进 Elastic net 正则化框架。

4. Linear-SVM

Linear-SVM 是 NTU 提出的一种 SVM 加速方法，本质也是线性分类器。它没有使用 kernel matrix，所以它比 LIBSVM 快速很多，NTU 提到如果 with/without（non-linear mappings）的准确率很接近，推荐使用 linear-SVM。也就是说 LIBSVM 多用于训练集是低维的情况，用 kernel 增加线性可分的程度。相反，如果训练集做了大量特征工程，维数很高，用 linear-SVM 更合适，同时也减少过拟合风险。

利用上述模型对之前所构成的特征进行训练。

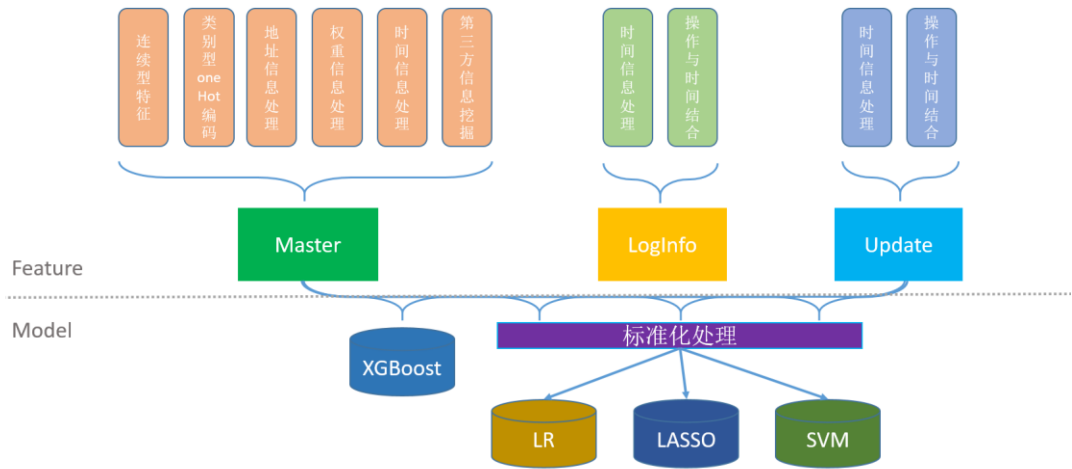


图 19 单模型训练

4. 模型融合

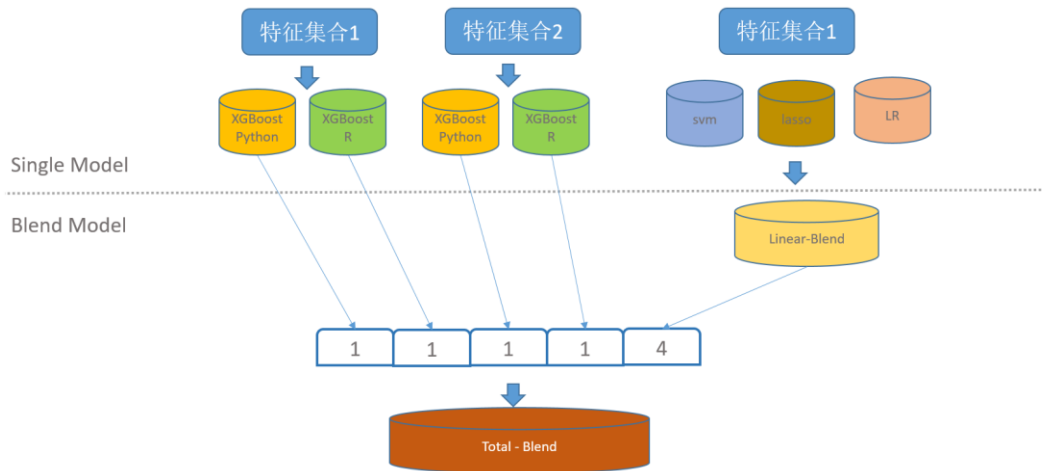


图 20 模型融合

模型	CV-9 折结果
XGB-4637 Python	0.7839
XGB-3952 Python	0.7857
XGB-3952 R	0.7852
XGB-4637 R	0.7845
LASSO	0.7745
LR	0.7752
Linear-SVM	0.7734
Linear-ensemble	0.7785
All-Blend-AUC	0.7896

