Sales Forecast for Pharmacy with Dynamical Prices

汇报人：杨继琛、谢宇、徐佳杭、万芳彬

目录

CONTENTS

PART 1

描述性分析

Data Description

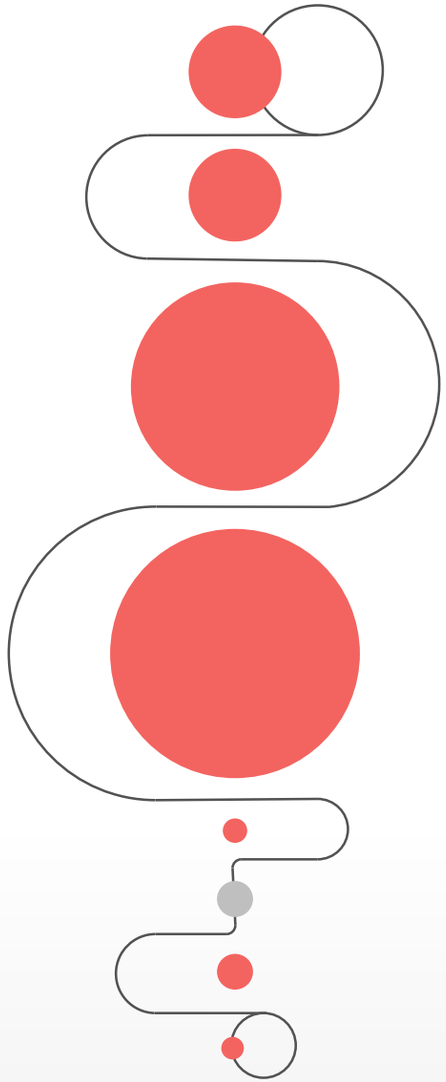
问题描述

通过建立预测模型，在动态价格下，针对每一次消费者登录网站的行为，预测消费者是否会购买药品，进而预测邮购药房的收入。

原始数据

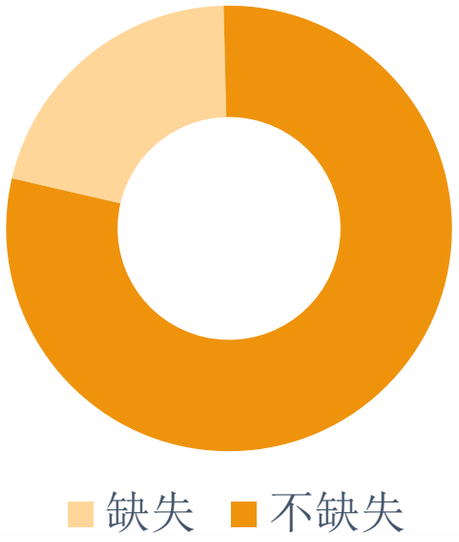
Items.csv 药品种类

- pid：药品唯一标识号
- manufacturer：药品制造商编号
- group： 药品分组
- category, pharmForm：药品剂型信息
- content, unit：药品剂量信息
- genericProduct：是否为基因药物
- salesIndex：配药规范代码
- campaignIndex：行为标签
- rrp：建议售价

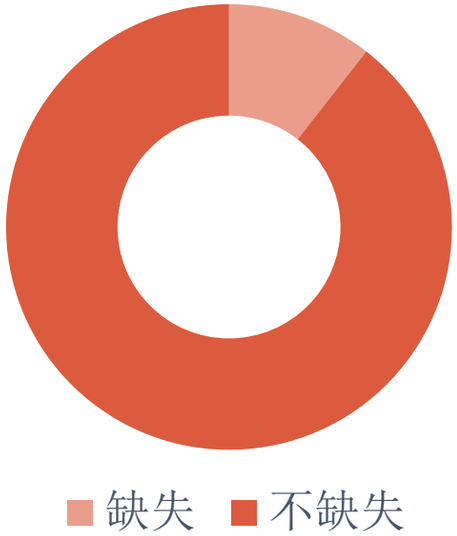


Items.csv

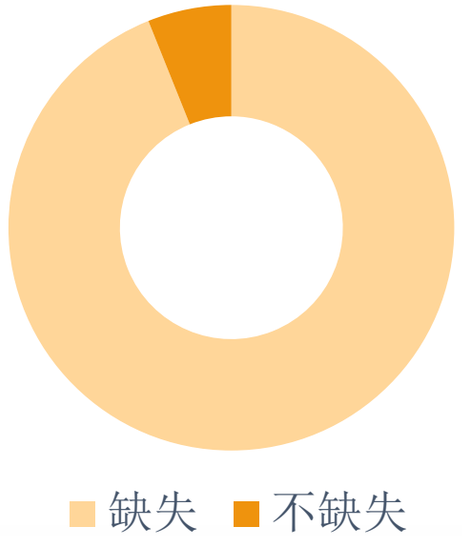
- 共有22035条数据，其中category，pharmForm，campaignIndex三列有缺失值



Category



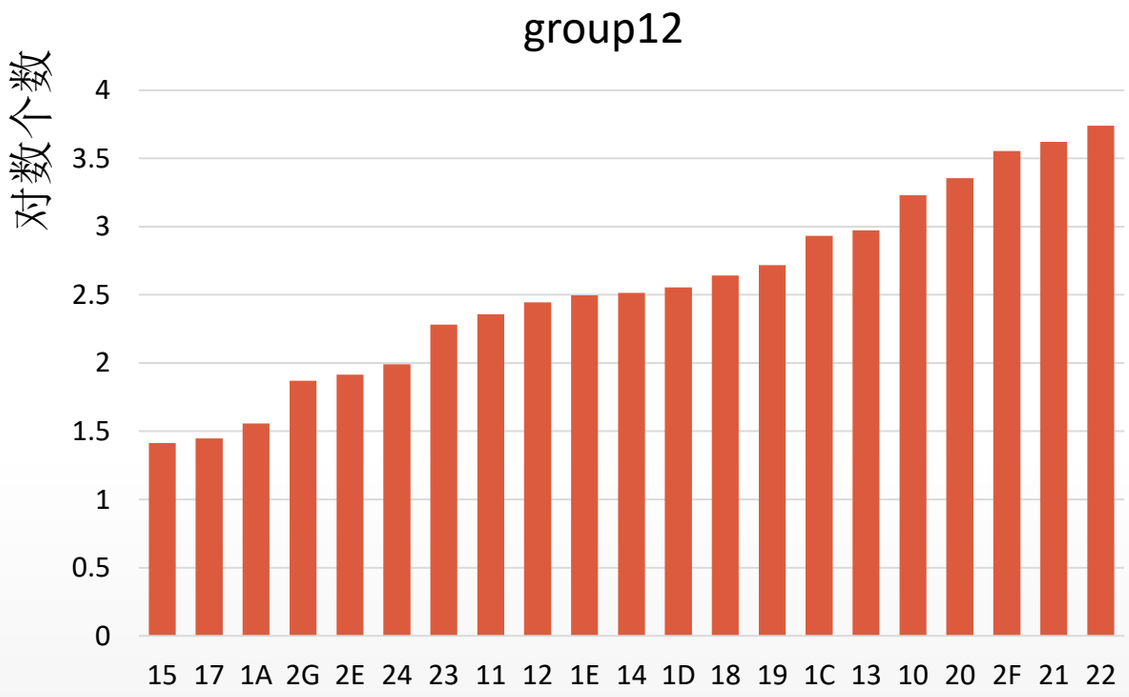
pharmForm

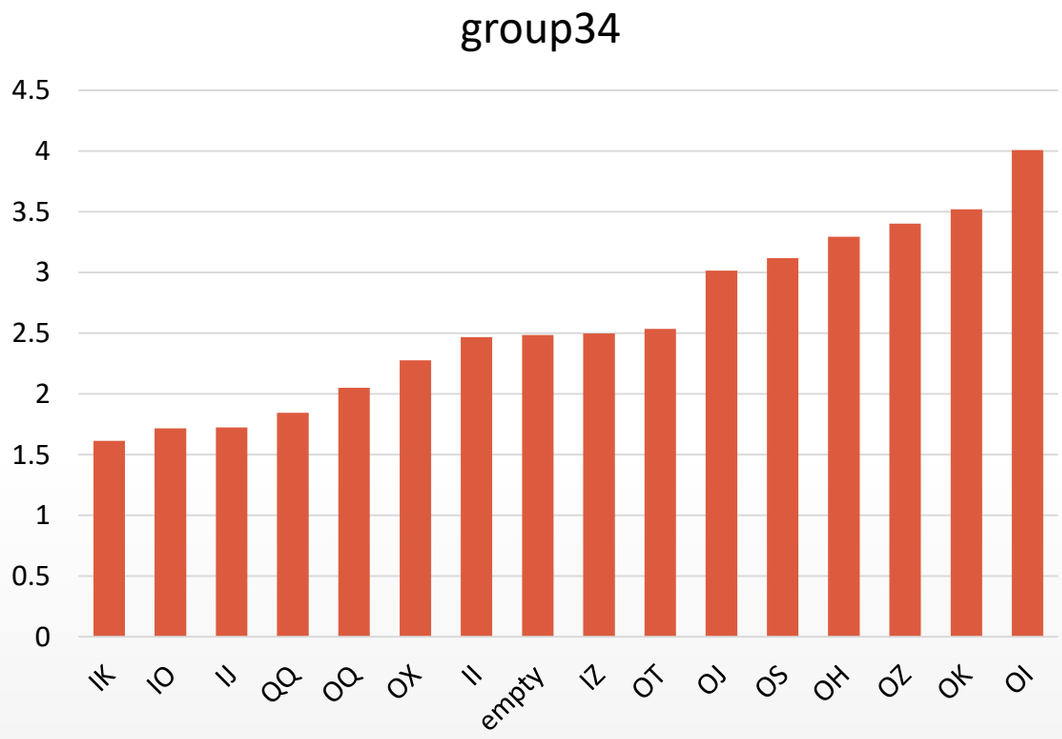


CampaignIndex

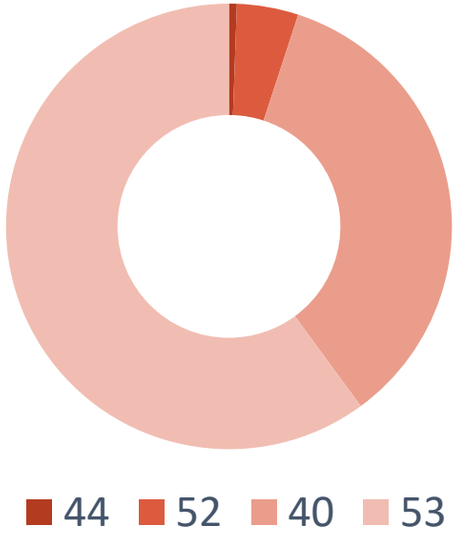
Items.csv

- 共有533种不同的group，我们推测group中的数字和字母的位数代表不同含义。
- 截取group的前两个变量，设为group12变量，截取group的中间两个变量，设为group34变量；

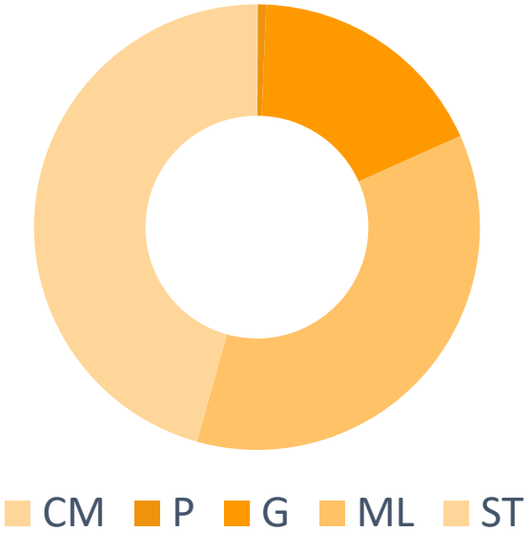




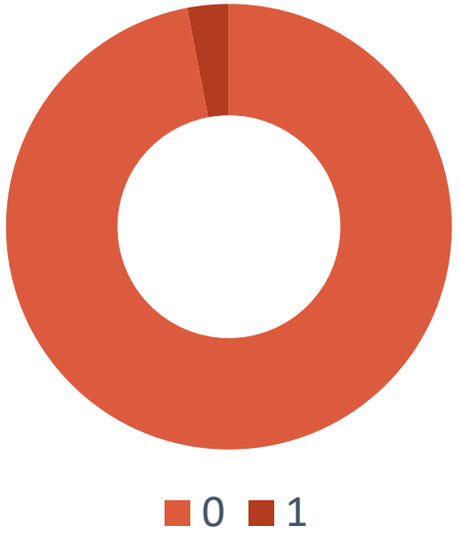
Item数据分析 - salesIndex



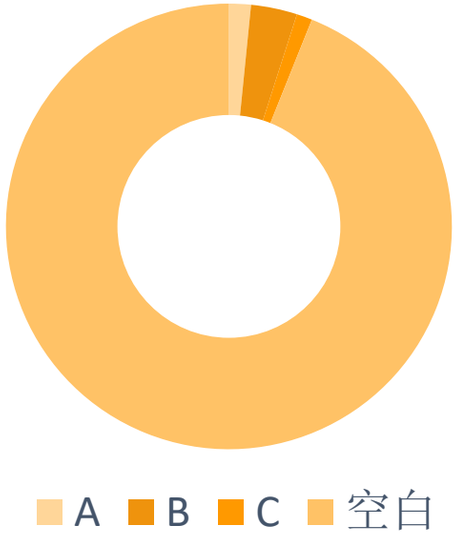
salesIndex:
共有四种类型



Unit:
共有五种类型



GenericProduct:
是否仿制药



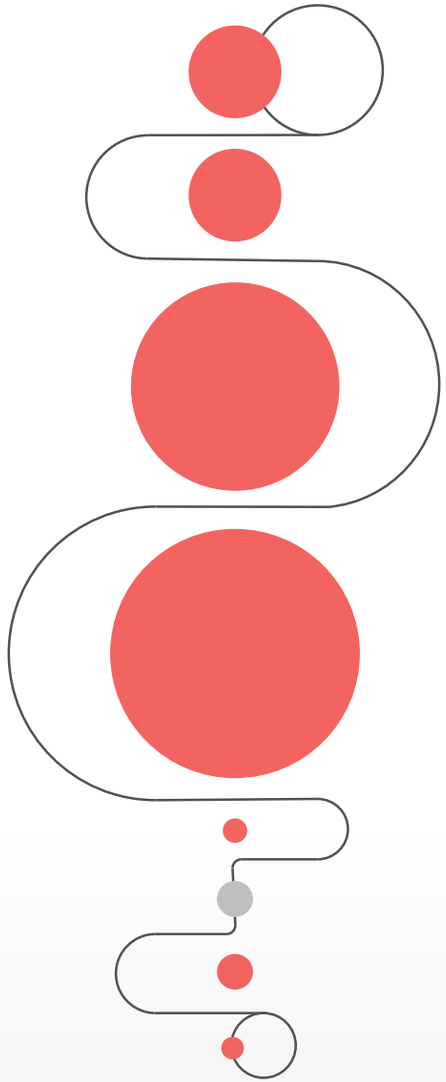
CampaignIndex:
大部分都是空白

原始数据

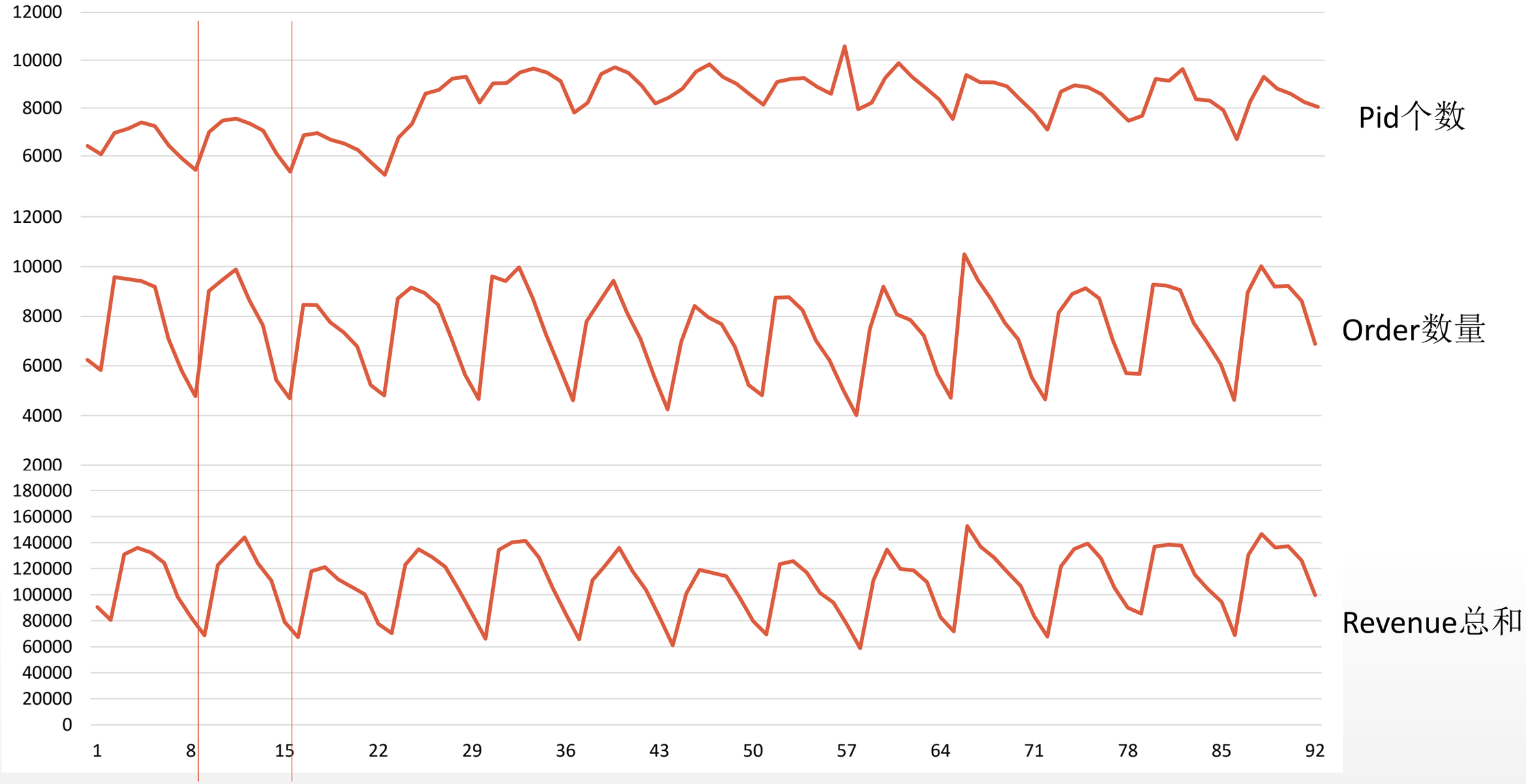
train.csv & class.csv

lineID,
Day
Pid
adFlag
Availability
CompetitorPrice
price

Click
Basket
Order
revenue

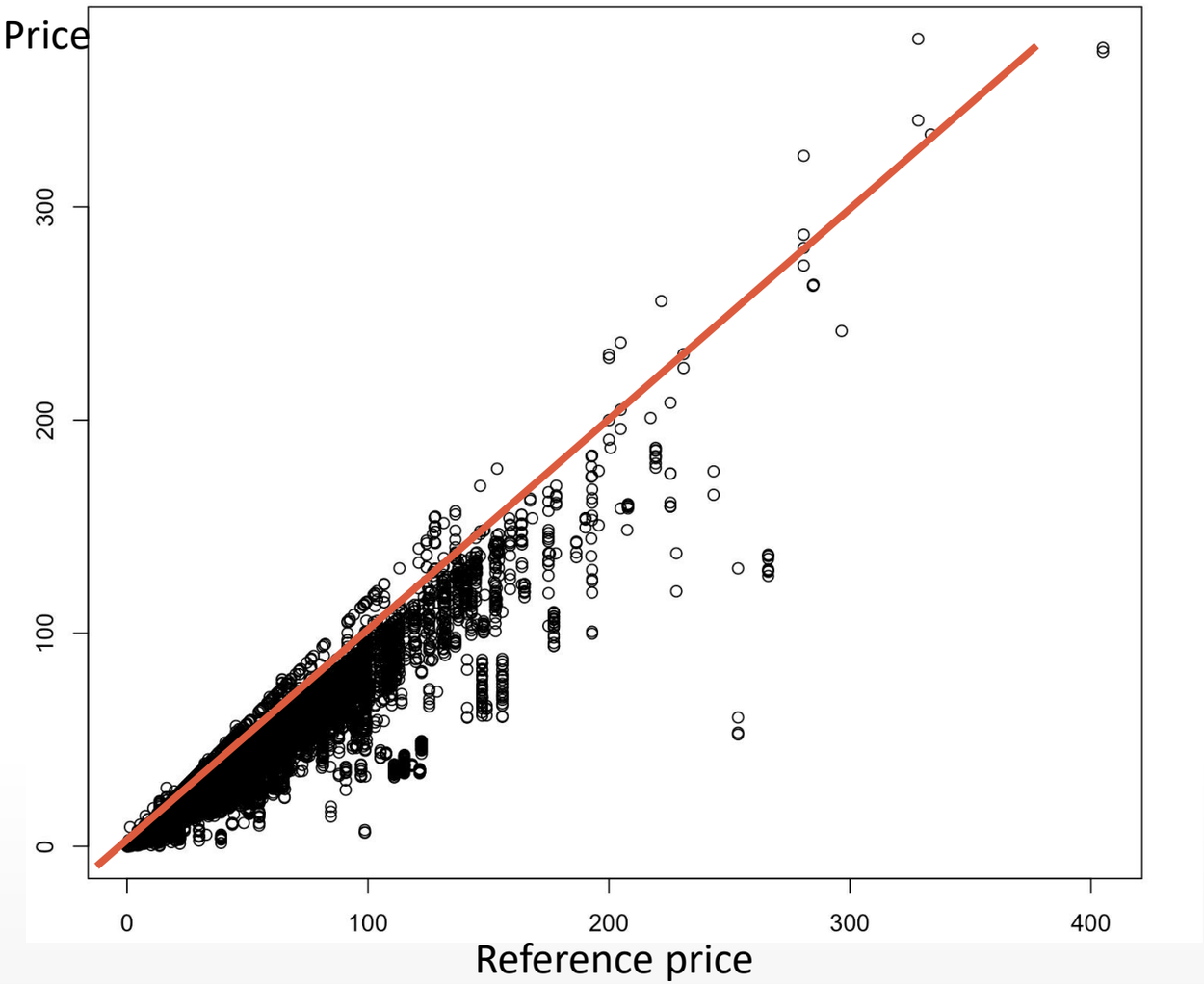


按照day进行总结

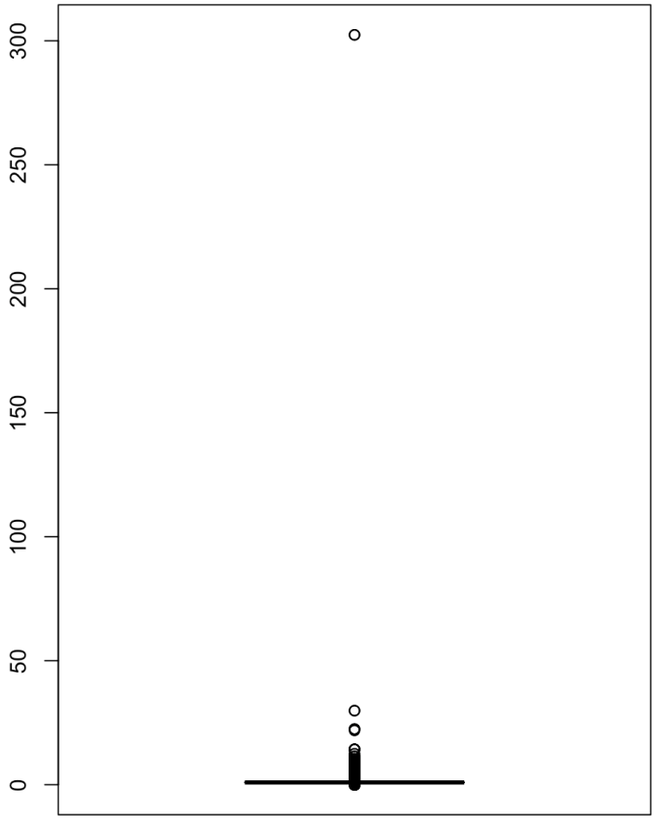

12000
10000
8000
6000
12000
10000
8000
6000
4000
2000
180000
160000
140000
120000
100000
80000
60000
40000
20000
0
1
8
15
22
29
36
43
50
57
64
71
78
85
92
Pid个数
Order数量
Revenue总和

Price

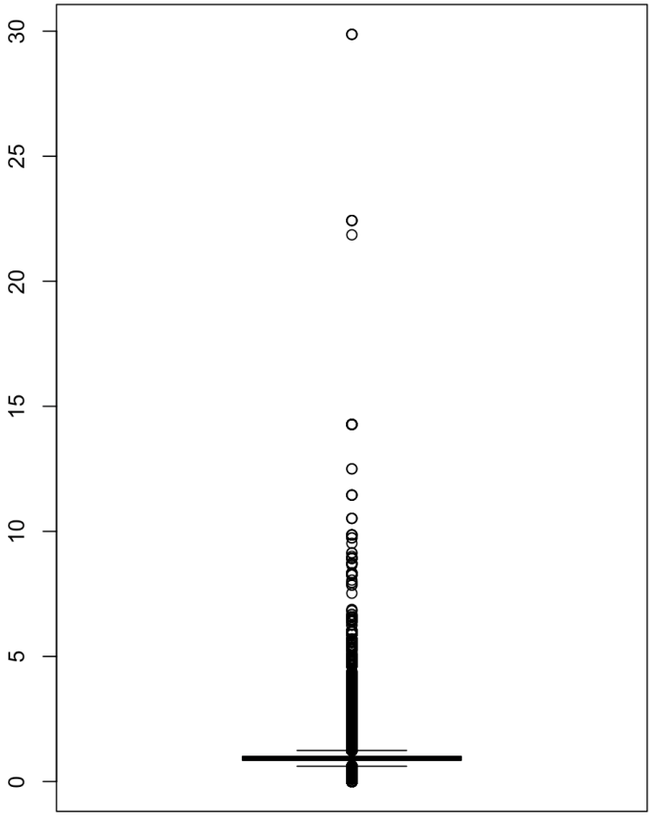
- Reference price和price有强相关关系，价格在参考售价的上下浮动。



Price & competitor price



竞争力箱线图



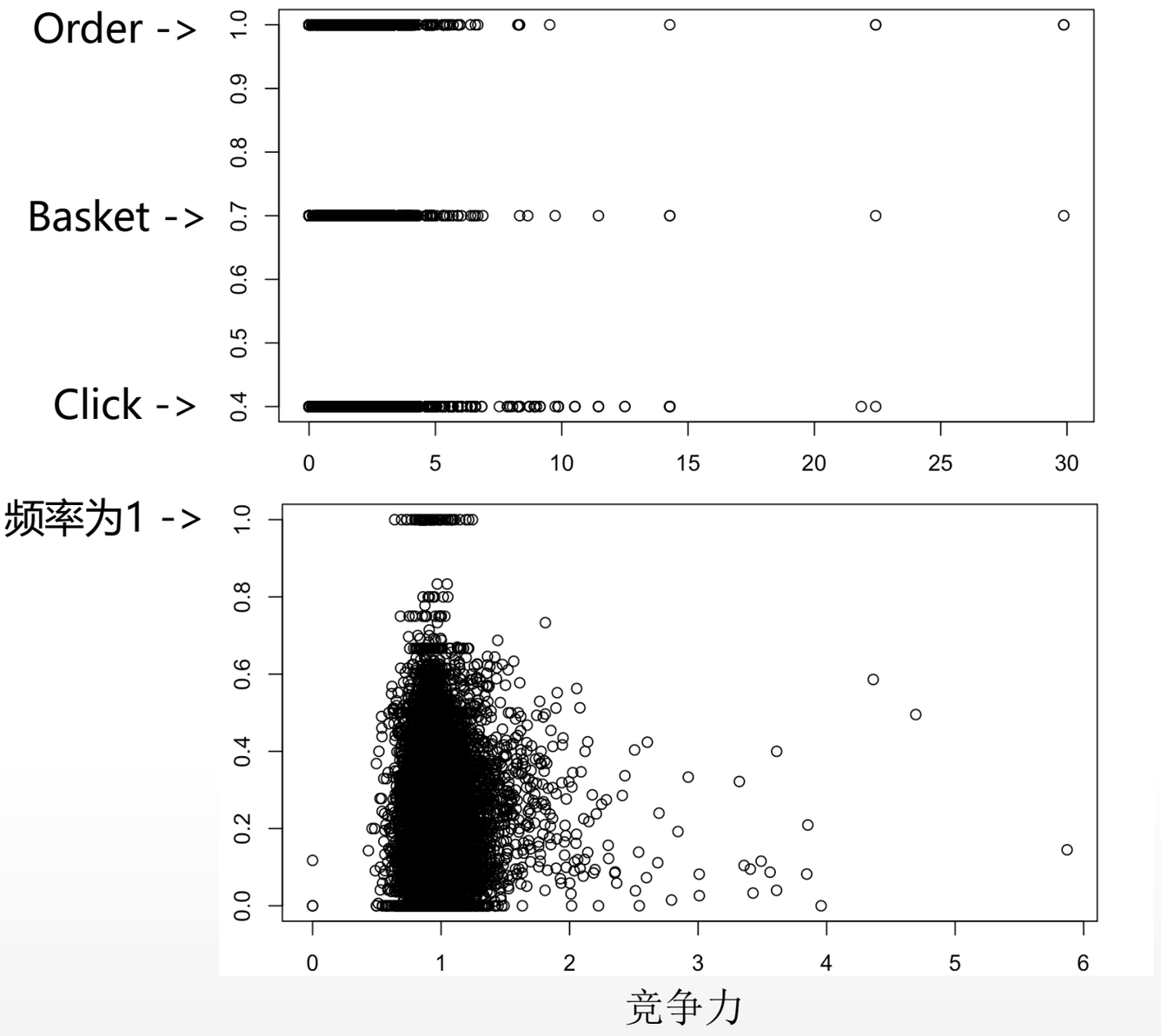
去除最高异常值后

- 计算价格和竞争者价格的比值，计算出我方的竞争力。
- 可以看出基本在1周围上下浮动，有明显的异常值

competitorPrice	price
24.19	0.08
24.19	0.08
24.19	0.08

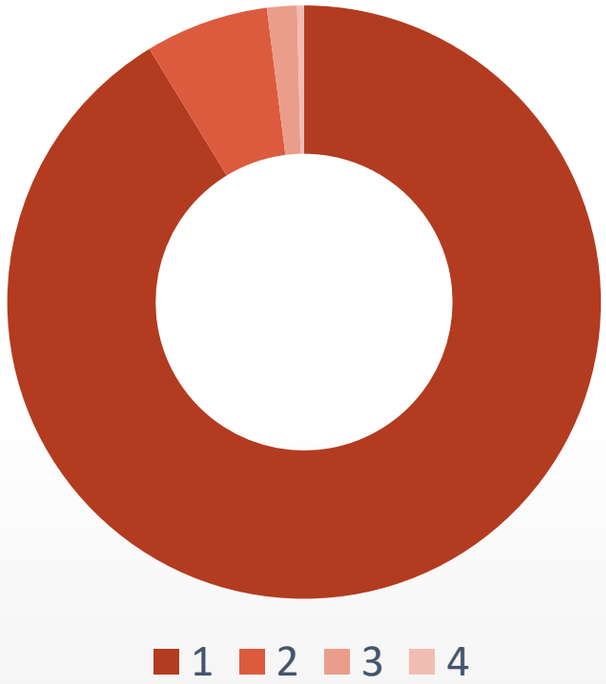
竞争力对行为的影响

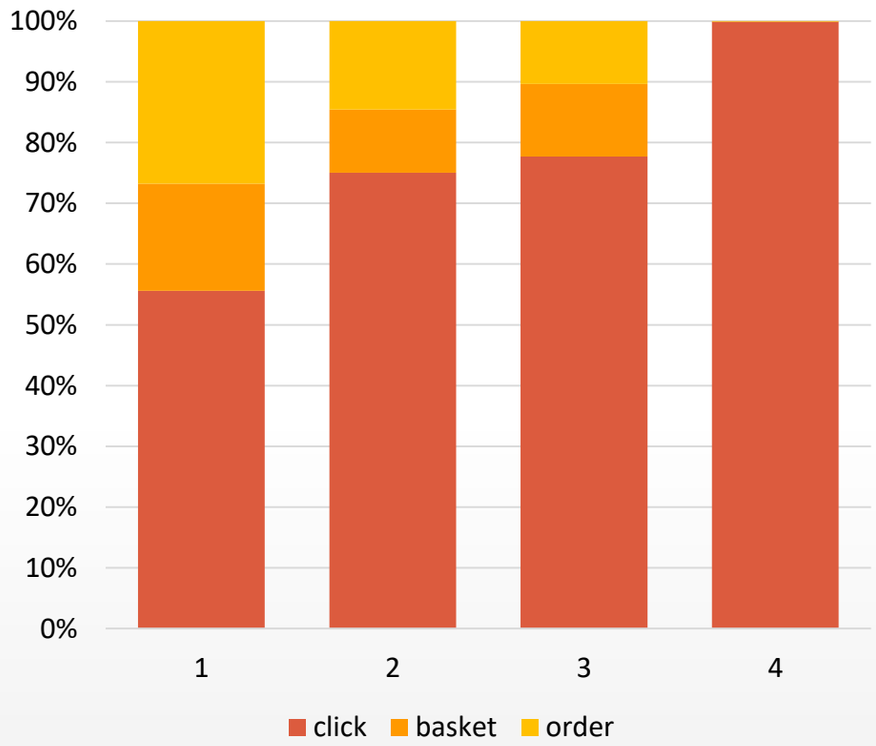
可以发现数据显示和我们的直观想象不同，我方价格竞争力越大，order的频率反而降低



Availability分析

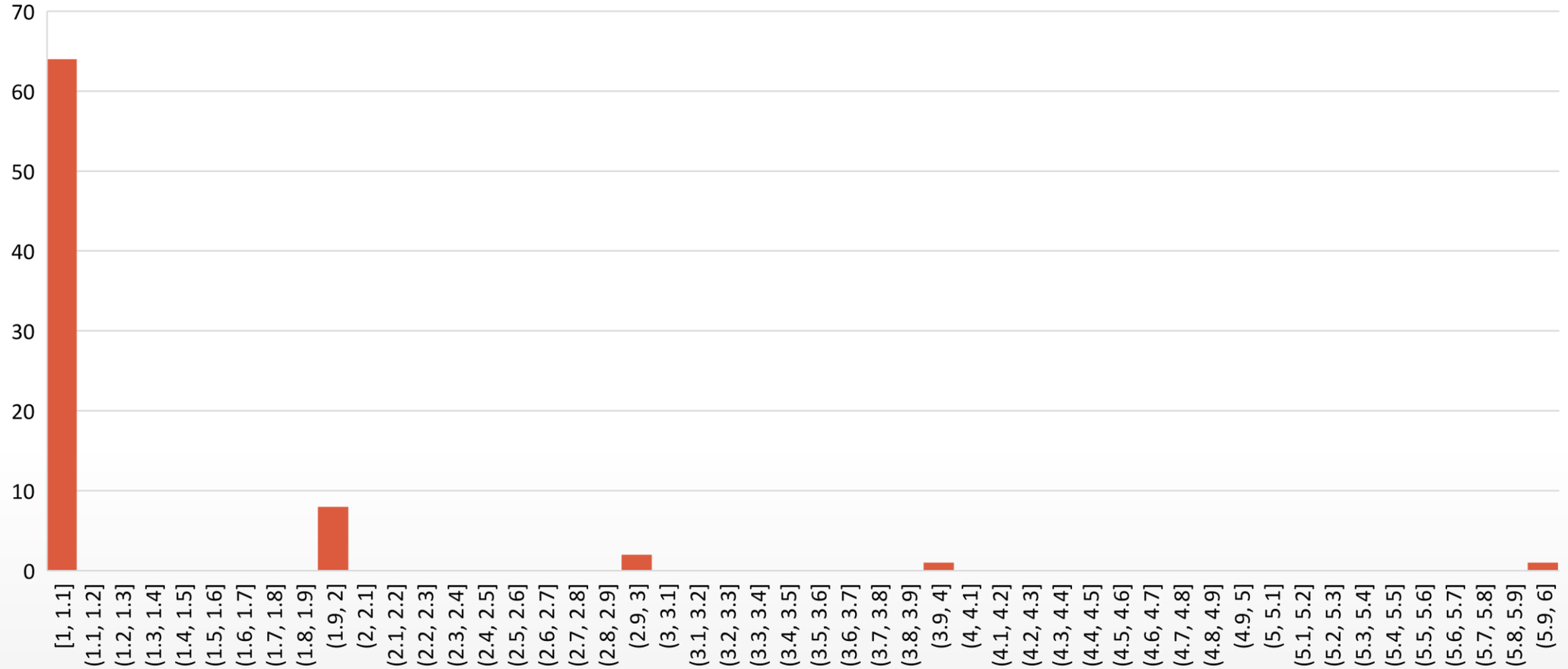
- Availability四种类型的个数占比
- 针对不同的availability，分别计算三种action的概率
- 可以发现1类型order的概率最高，说明发现有货时更容易倾向于购买





revenue

计算quantity = Revenue/price，可以发现有order的条目中，绝大多数都是1次，但是有quantity大于1的情形。



- 对train数据集进行Unique

	pid	day	adFlag	availability	competitorPrice	price	n()	n_distinct(click)	n_distinct(basket)	n_distinct(order)
755878	2655	90	1	1	14.26	15.35	622	2	2	2
755877	2655	88	1	1	13.74	15.35	615	2	2	2
755876	2655	83	1	1	15.37	15.35	610	2	2	2
755875	2655	24	1	1	14.26	15.35	609	2	2	2
755873	2655	82	1	1	13.74	15.35	608	2	2	2
755874	2655	85	1	1	13.74	15.35	608	2	2	2
755871	2655	25	1	1	13.74	15.35	606	2	2	2
755872	2655	54	1	1	13.74	15.35	606	2	2	2
755870	2655	92	1	1	13.74	15.35	605	2	2	2
755869	2655	81	1	1	13.74	15.35	603	2	2	2
755866	2655	42	1	1	13.74	15.35	602	2	2	2

PART 2

特征工程

FEATURE ENGINEERING

数据预处理—变量转换

DATA PROCESSING

content:

将其中类似 “ X * Y * Z” 的字符串转化为数值

group:

将前1/2位和3/4位数字拆分出来组成新变量：
group12，group34

unit:

我们认为可能代表药品单位，对单位进行统一

- ML and L > unified_ML,
- KG and G > unified_G
- CM and M > unified_CM
- ST > unified_ST
- P > unified_P

数据预处理--缺失值填补

DATA PROCESSING

基于**朴素贝叶斯**进行数据填补

A 数值型变量填补

1. 用相同商品的竞争者价格均值填补；
2. 用出现重复商品的竞争者价格均值进一步填补；
3. 用相似商品的竞争者价格均值填补；

(group, content, unit, day_7, salesIndex, adFlag等相同)

4. 最后用实际价格填补。

B 因子型变量填补

1. 用manufacturer, group12, group34, unit 相同的数据中占比最高的进行填补；
2. 用manufacturer, group12, group34相同的数据中占比最高的填补；
3. 用group12, group34相同的数据中占比最高的填补；
4. 将前三轮填补完成之后的300条数据用频数最高的值进行填补。

特征工程
FEATURE ENGINEERING

变量分类	变量序号	变量名	变量含义
时间特征	1	day_7	以7天为周期对时间进行划分
	2	day_14	以14天为周期对时间进行划分
	3	day_30	以30天为周期对时间进行划分
价格特征	4	rrp_per_unit	平均每个单位商品的参考价格
	5	price_per_unit	平均每个单位商品的实际销售价格
	6	competitorPrice_per_unit	平均每个单位商品的竞争者价格
	7	price_diff	实际销售价格和竞争者价格的差价
	8	price_discount	自实际销售价格相比于参考价格的折扣
	9	competitorPrice_discount	自身和竞争者的折扣差
	10	price_discount_diff	竞争者的折扣
	11	is_lower_price	是否比竞争者更低价
	12	is_discount	是否有折扣
	13	is_greater_discount	是否比竞争者更低折扣

特征工程
FEATURE ENGINEERING

变量分类	变量序号	变量名	变量含义
价格特征	14	price_discount_min	同类商品的最低价
	15	price_discount_p25	同类商品价格的四分位数
	16	price_discount_med	同类商品价格的中位数
	17	price_discount_p75	同类商品价格的四分之三位数
	18	price_discount_max	同类商品价格的最高价格
	19	price_discount_min	同类商品的最低价
	20	price_discount_p25	同类商品价格的四分位数
	21	price_discount_med	同类商品价格的中位数
次数特征	22	click_time	同一pid商品被点击次数
	23	basket_time	同一pid商品被收藏次数
	24	order_time	同一pid商品被购买次数
购买特征	25	group12_order	group12中购买总量
	26	group34_order	group34中购买总量
	27	week_order	一周中每一天购买总量

特征工程
FEATURE ENGINEERING

变量分类	变量序号	变量名	变量含义
单位价格特征	28	price_per_ML	平均每ML价格
	29	price_per_G	平均每G价格
	30	price_per_CM	平均每CM价格
	31	price_per_ST	平均每ST价格
	32	price_per_P	平均每P价格
	33	Cprice_per_ML	平均每ML竞争者价格
	34	Cprice_per_G	平均每G竞争者价格
	35	Cprice_per_CM	平均每CM竞争者价格
	36	Cprice_per_ST	平均每ST竞争者价格
	37	Cprice_per_P	平均每P竞争者价格
	38	rrp_per_ML	平均每ML参考价格
	39	rrp_per_G	平均每G参考价格
	40	rrp_per_CM	平均每CM参考价格
	41	rrp_per_ST	平均每ST参考价格
	42	rrp_per_P	平均每P参考价格

特征工程
FEATURE ENGINEERING

变量分类	变量序号	变量名	变量含义
概率特征	43	num_pid_order	每一个pid曾经的购买量（先验）
	44	pid_prob	每一个pid被再次购买的概率
	45	availability_likelihood	借助order进行likelihood编码
	46	pid_likelihood	借助order进行likelihood编码
	47	day_7_likelihood	借助order进行likelihood编码

总共**47**个变量，

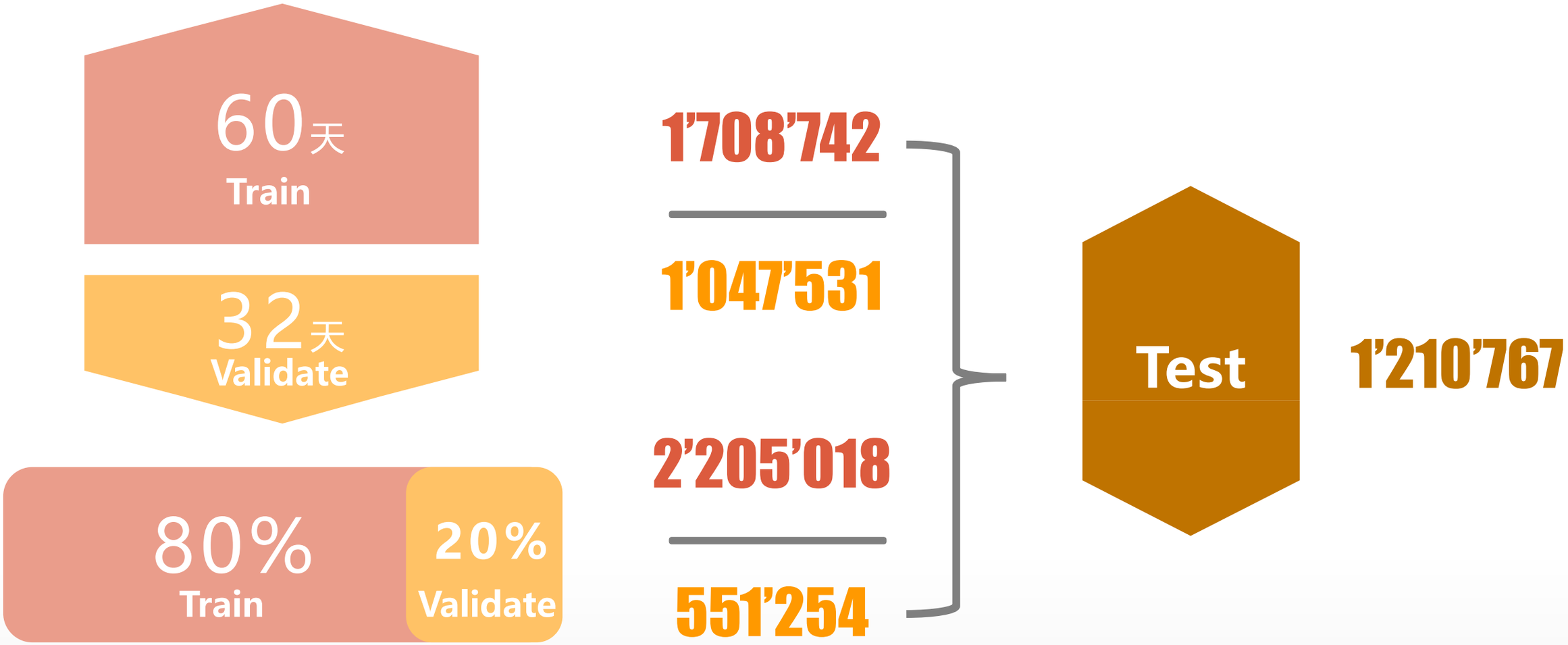
涵盖了：**时间特征、价格特征、次数特征、次数特征、单位价格特征、概率特征**

PART 3

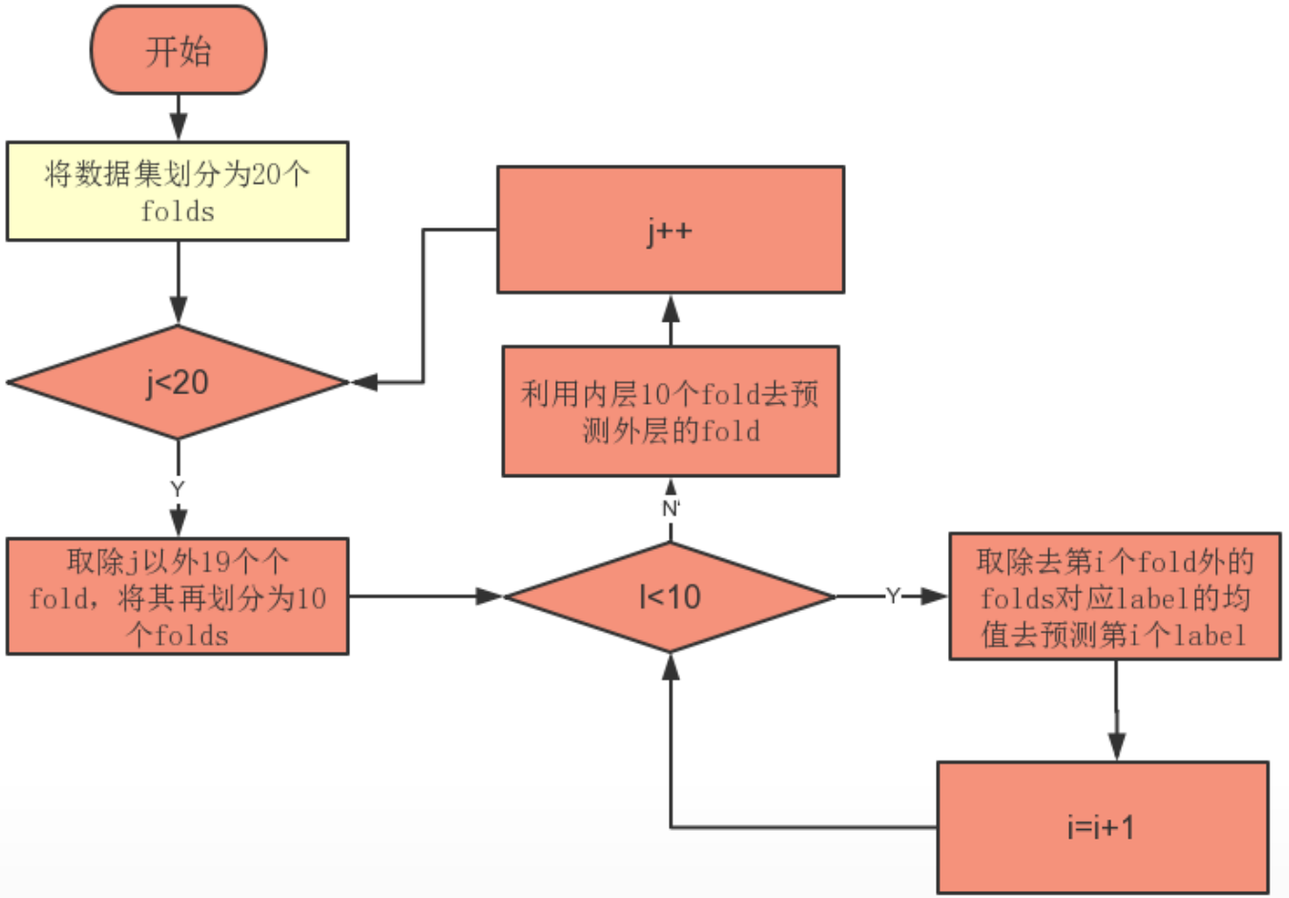
模型拟合

MODELING

数据集划分


60天
Train
32天
Validate
80%
Train
20%
Validate
1’708’742
1’047’531
2’205’018
551’254
Test
1’210’767

Likelihood Encoding



主要思想：

将category类变量用对应的label变量取值的均值来替代，同时为了避免变量的信息没能传递给label，使用K折交叉法进行取值。

利用train数据集中的order变量train/validate/test数据集对group_34、group_12、content、unit、group、pharmForm等变量进行了likelihood encoding。

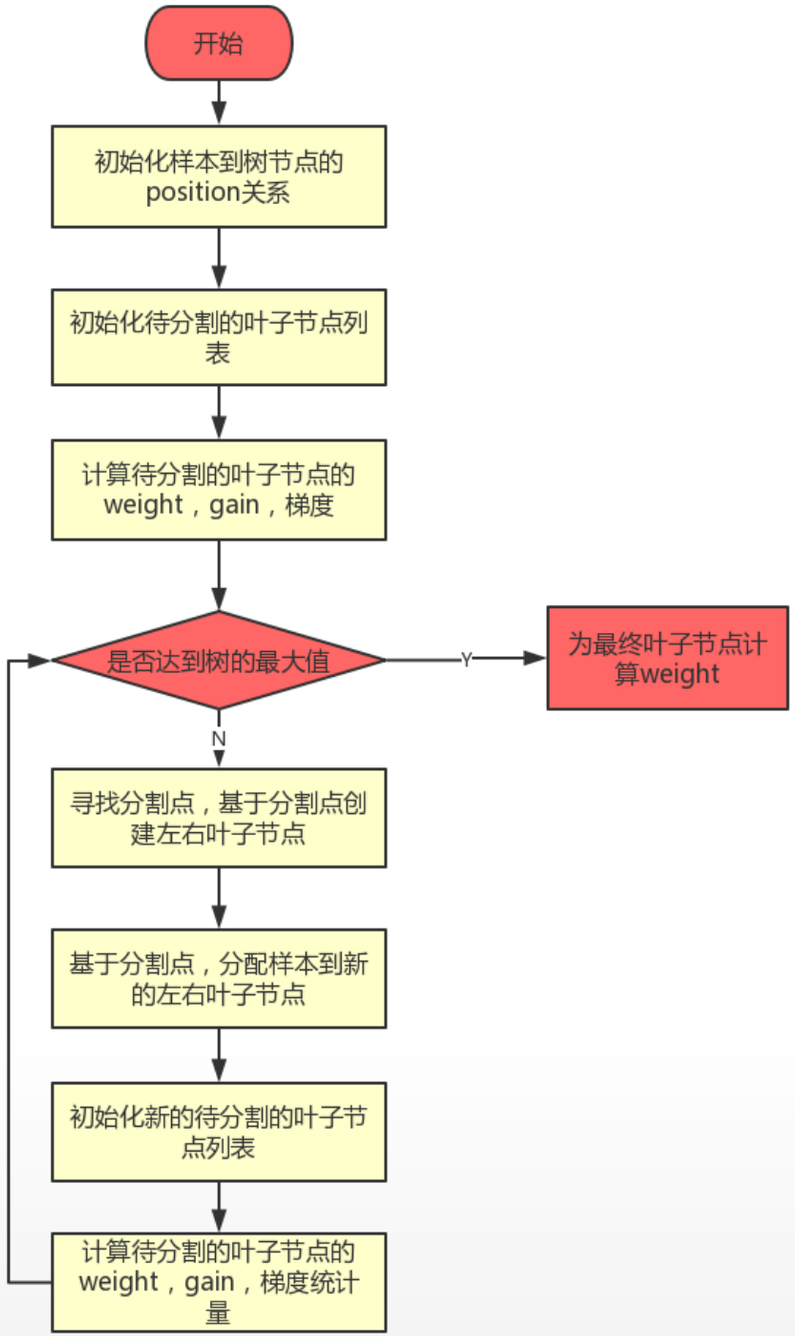
XGBOOST

Algorithm 1: Exact Greedy Algorithm for Split Finding

Input: I, instance set of current node
Input: d, feature dimension
$gain \leftarrow 0$
$G \leftarrow \sum_{i \in I} g_i$, $H \leftarrow \sum_{i \in I} h_i$
for $k = 1$ ***to*** m **do**
  $G_L \leftarrow 0$, $H_L \leftarrow 0$
  for j *in sorted(I, by* $\mathbf{x}_{jk}$*)* **do**
    $G_L \leftarrow G_L + g_j$, $H_L \leftarrow H_L + h_j$
    $G_R \leftarrow G - G_L$, $H_R \leftarrow H - H_L$
    $score \leftarrow \max(score, \frac{G_L^2}{H_L+\lambda} + \frac{G_R^2}{H_R+\lambda} - \frac{G^2}{H+\lambda})$
  end
end
Output: Split with max score

Algorithm 2: Approximate Algorithm for Split Finding

for $k = 1$ ***to*** m **do**
  Propose $S_k = \{s_{k1}, s_{k2}, \cdots s_{kl}\}$ by percentiles on feature k.
  Proposal can be done per tree (global), or per split(local).
end
for $k = 1$ ***to*** m **do**
  $G_{kv} \leftarrow= \sum_{j \in \{j | s_{k,v} \geq \mathbf{x}_{jk} > s_{k,v-1}\}} g_j$
  $H_{kv} \leftarrow= \sum_{j \in \{j | s_{k,v} \geq \mathbf{x}_{jk} > s_{k,v-1}\}} h_j$
end
Follow same step as in previous section to find max score only among proposed splits.



LGBM

为了解决在大样本高纬度数据的环境下耗时的问题，Lightgbm使用了如下解决办法：

1. GOSS（基于梯度的单边采样），不是使用所用的样本点来计算梯度，而是对样本进行采样来计算梯度；
2. EFB（互斥特征捆绑），这里不是使用所有的特征来进行扫描获得最佳的切分点，而是将某些特征进行捆绑在一起来降低特征的维度，是寻找最佳切分点的消耗减少。

这样大大的降低的处理样本的时间复杂度，甚至有时还会提升精度。

Algorithm 2: Gradient-based One-Side Sampling

Input: I: training data, d: iterations
Input: a: sampling ratio of large gradient data
Input: b: sampling ratio of small gradient data
Input: $loss$: loss function, L: weak learner
models ← {}, fact ← $\frac{1-a}{b}$
topN ← a × len(I) , randN ← b × len(I)
for i = 1 ***to*** d **do**
  preds ← models.predict(I)
  g ← $loss$(I, preds), w ← {1,1,...}
  sorted ← GetSortedIndices(abs(g))
  topSet ← sorted[1:topN]
  randSet ← RandomPick(sorted[topN:len(I)], randN)
  usedSet ← topSet + randSet
  w[randSet] × = fact ▷ Assign weight $fact$ to the small gradient data.
  newModel ← L(I[usedSet], − g[usedSet], w[usedSet])
  models.append(newModel)

Algorithm 3: Greedy Bundling

Input: F: features, K: max conflict count
Construct graph G
searchOrder ← G.sortByDegree()
bundles ← {}, bundlesConflict ← {}
for i ***in*** *searchOrder* **do**
  needNew ← True
  for j = 1 ***to*** *len(bundles)* **do**
    cnt ← ConflictCnt(bundles[j],F[i])
    if *cnt + bundlesConflict[i]* $\leq K$ **then**
      bundles[j].add(F[i]), needNew ← False
      break
  if *needNew* **then**
    Add $F[i]$ as a new bundle to *bundles*

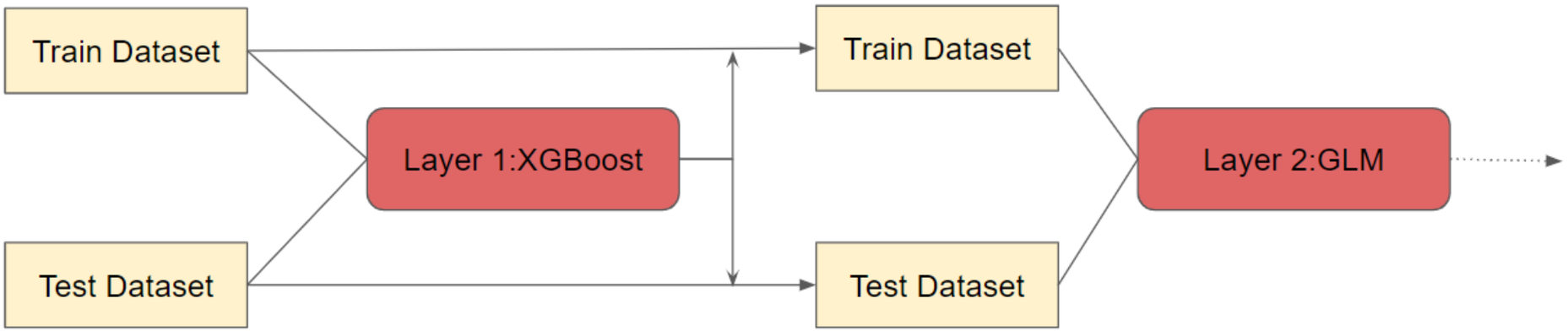
Output: *bundles*

模型架构

两层模型：

第一层模型用XGBoost预测是否购买行为；

第二层模型使用GLM预测购买数量的多少。



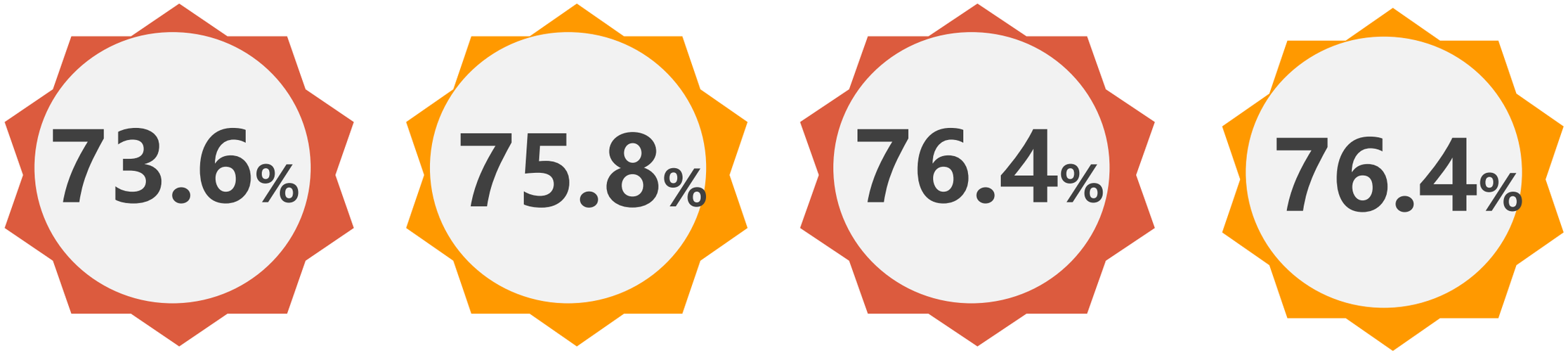
PART 4

结果分析

RESULT ANALYSIS

WORKFLOW

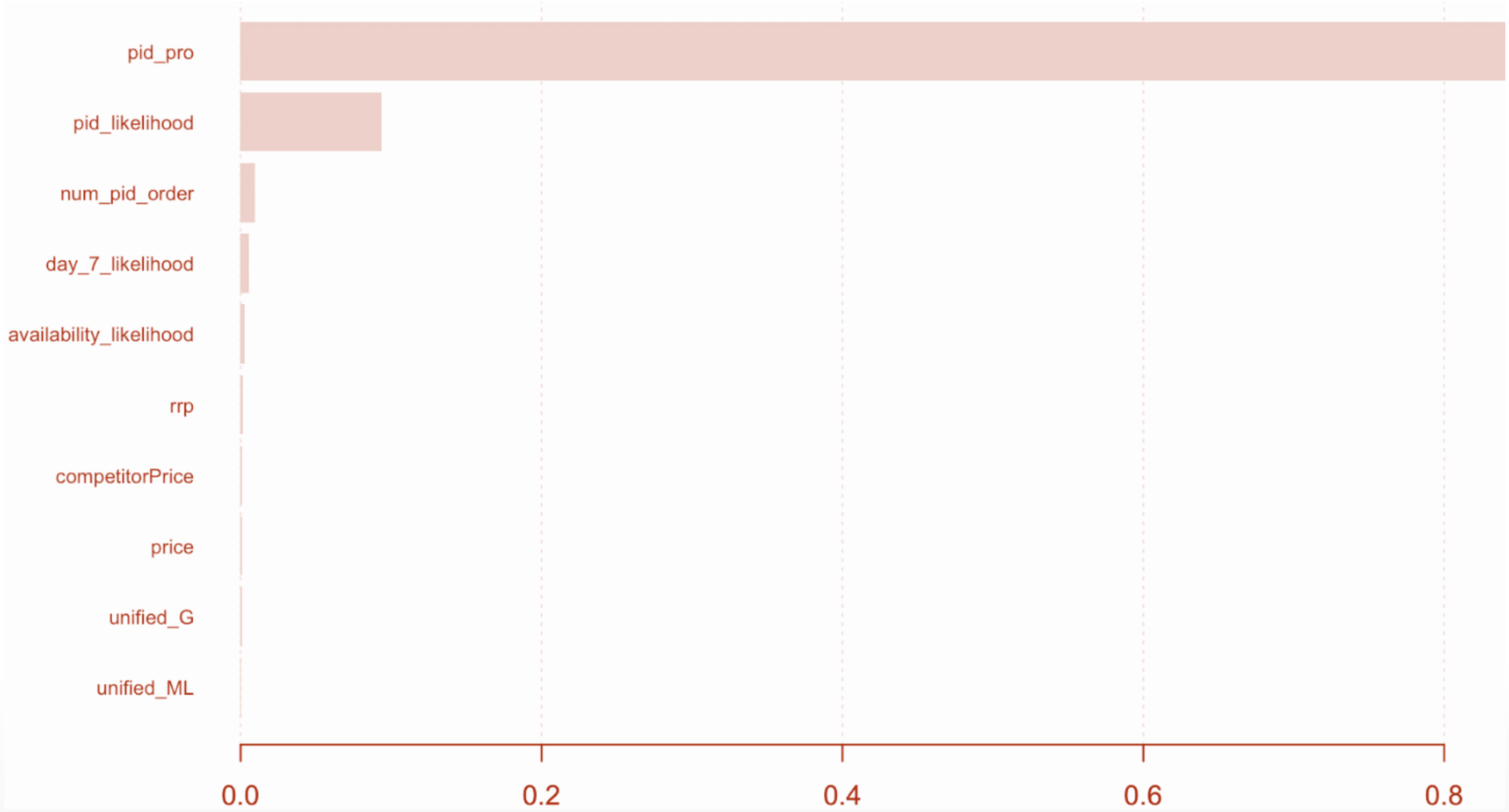
模型准确率



Logistice回归　　LGBM　　XGBoost　　XGBoost_unique

特征重要性

FEATURE INMPORTANCE



PART 5

思考与总结

THINKING AND CONCLUSION

思考与总结

Conclusion

1. 缺少user数据：个体是否购买药品实际与个体特征的相关性最高。所以预测order准确率的上限不会太高。

2. 数据相关性较差，可以在train上达到100%正确率，但是在test上效果很差，所以需要提取强相关变量。

3. 正负样本不均衡，训练时需要给正样本（order=1）加权。

4. 数据特征完全一样，但order的结果不同的这些数据需要重视。

我们学到的

What We Learned

1. 对数据集的理解十分重要，需要每个人都去做“重复”的探索性分析。
2. 有了想法就一定要去实现，然后再评价好坏。
3. 构建模型和特征工程同步，随时修正和调整，随着模型的深入会有更多的启发。
4. 留出时间调参，充分调用可使用的计算资源。

THANK YOU

感谢聆听，批评指导