天善智能
TIANSHAN SOFT
专注于商业智能BI和大数据的垂直社区平台

用户付费行为深度挖掘实战

谢佳标（Daniel.xie）

用户深度挖掘

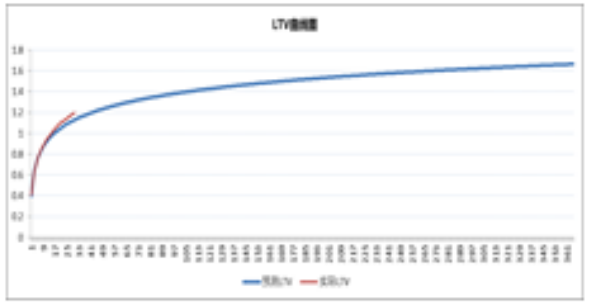
- 随着游戏市场竞争的日趋激烈，在如何获得更大收益延长游戏周期的问题上，越来越多的手机游戏开发公司开始选择借助大数据，以便挖掘更多更细的用户群、了解用户习惯来进行精细化、个性化的运营。游戏行业对用户的深度挖掘一般从两方面着手：

  - 一方面是用户游戏行为的深度分析，如玩家在游戏中的点击事件行为挖掘，譬如说新手教程中的点击事件，我们一般选择最关心的点击事件（即关键路径）进行转化率的分析（统计每个关键路径的点击人数或次数），通过漏斗图的展现形式就可以直接看出每个关键路径的流失和转化情况。漏斗图适合于单路径转化问题，如果涉及到多路径（点击完一个按钮后有多个按钮同时提供选择）情况时，可以使用路径分析的方法，路径分析更加基础、更加全面、更加丰富、更能真实再现玩家在游戏中的行为轨迹。
  - 另一方面是对用户付费行为的深度挖掘。付费用户是直接给公司创造价值的核心用户群，通过研究这批用户的付费数据，把脉其付费特征，可以实现精准推送，有效付费转化率。

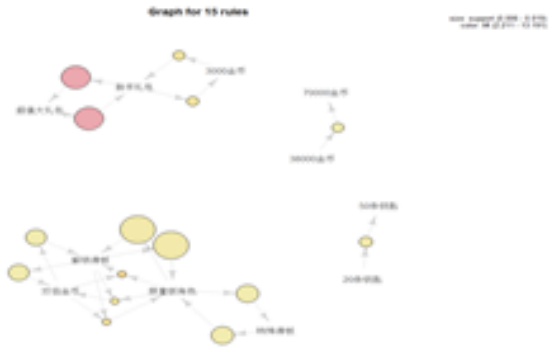
付费用户深度挖掘

付费玩家常用分析方法

LTV预测法



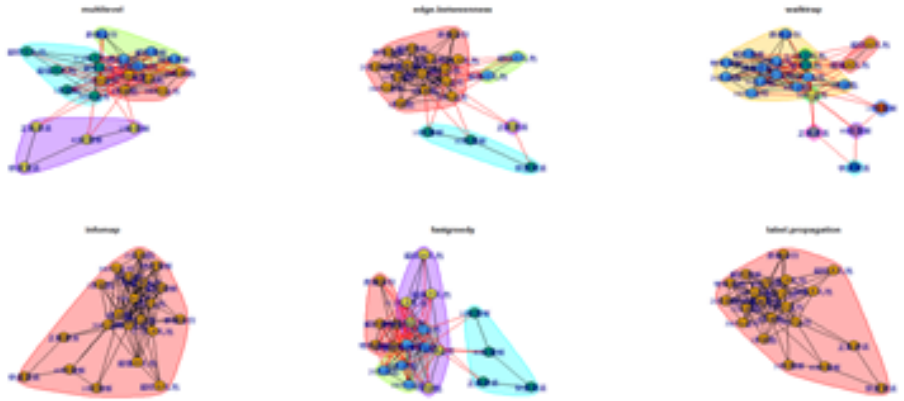
玩家物品购买关联分析



基于玩家物品的智能推荐

```
> as(recom3,"list")[1:5] #查看前五个玩家的 top3 推荐
$`107204535`
[1] "超值大礼包" "3000 金币"     "超级大礼包"
$`213666611`
[1] "15000 金币" "20 条钥匙"
$`226500629`
```

社会网络方法



目录

LTV概念

LTV

Life	Time	Value

用户生命周期的价值

玩家的充值金额	免费玩家的沙包价值

LTV作用

优化渠道配置

考量ROI

LTV计算方法一

注册当天

100人

100元

$$LTV(1) = \frac{100元}{100人} = 1元/人$$

第二天

200元

$$LTV(2) = \frac{100 + 200}{100} = 3$$

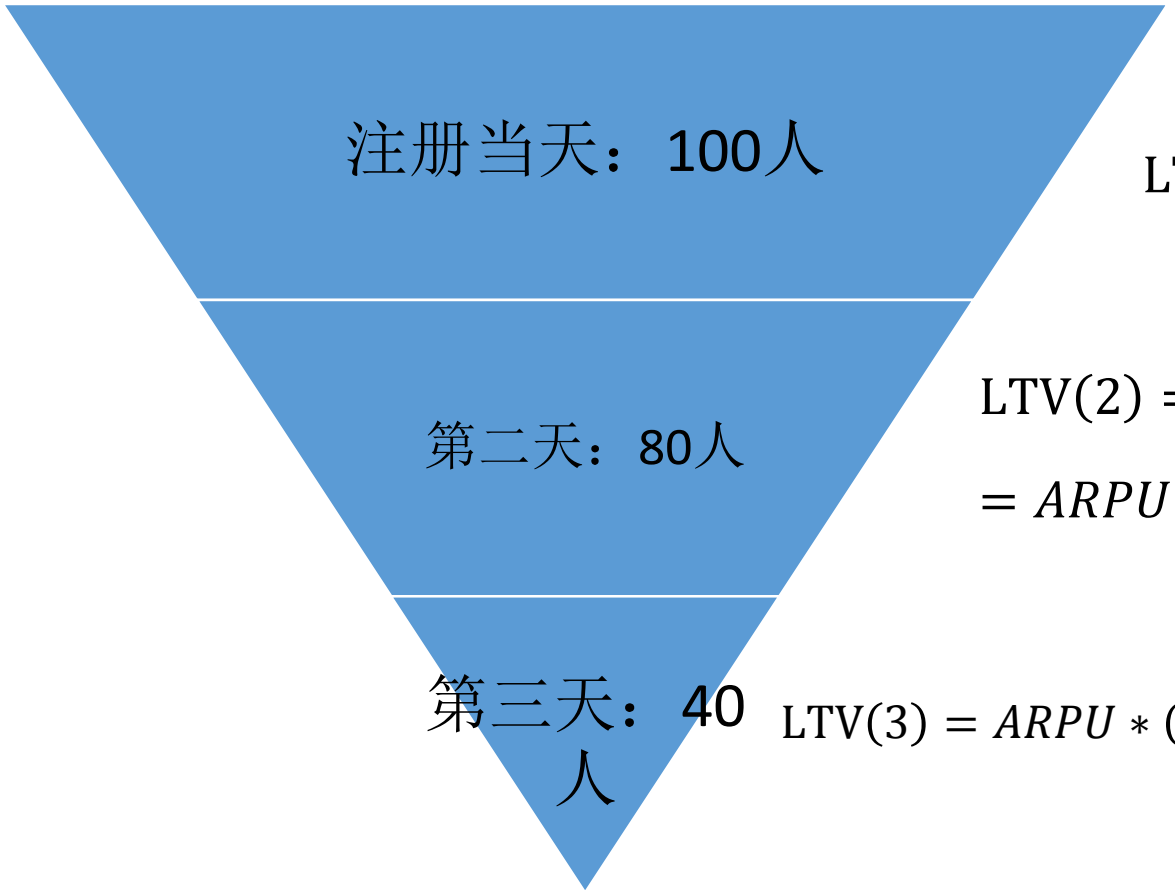
第三天

300元

$$LTV(3) = \frac{100 + 200 + 300}{100} = 6$$

LTV计算方法二

ARPU(活跃用户人均付费金额）:1.4元



$$\mathrm{LTV}(1) = 1.4 * 100\% = 1.4$$

$$\mathrm{LTV}(2) = ARPU * 100\% + ARPU * R1$$
$$= ARPU * (1 + R1) = 1.4 * \left(1 + \frac{80}{100}\right) = 2.52$$

$$\mathrm{LTV}(3) = ARPU * (1 + R1 + R2) = 1.4 * \left(1 + \frac{80}{100} + \frac{40}{100}\right) = 3.08$$

LTV的预测必要性

- 前面讲的LTV是根据已知的留存率、付费金额等指标计算得到的，属于事后统计LTV值。有时候，我们可能需要去了解在未来一段时间的用户生命价值（LTV），通过对比LTV与CPA，计算单个用户的ROI，进一步辅助市场广告投放决策。

- 进行LTV预测常用的方法有两种:

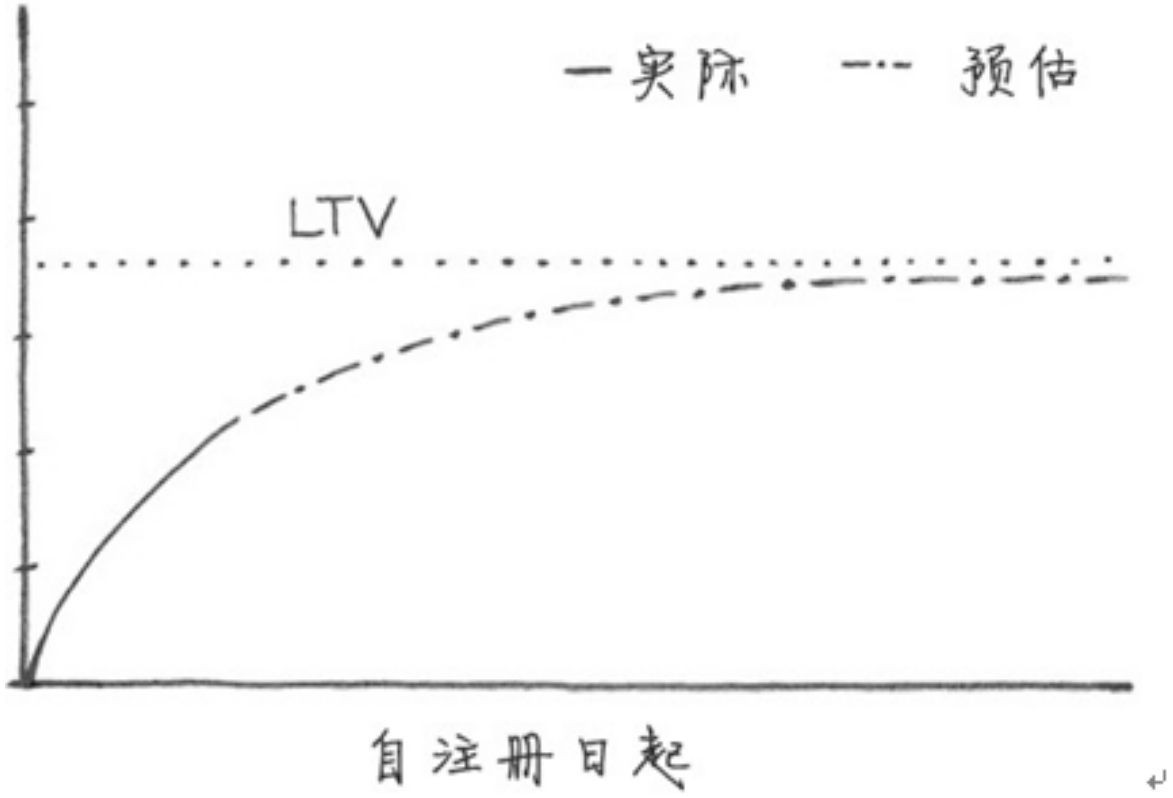
一种LTV的预测方法就是利用前期的LTV值对未来LTV进行预测

OR

一种LTV的预测是利用已知的留存率和ARPU，对未来LTV进行预测。

方法一：利用前期LTV进行预测

- 其基本原理是当我们对某日新用户的总收入跟踪时，可以发现，随着时间的增长，收入增长会逐渐放缓，可以想象，在未来足够长的时间里总收入会趋近一个估值，这个值可以被认为是某日新用户的总生命周期价值，当然我们也可以很容易得出其平均生命周期价值（LTV）。



方法一：利用前期LTV进行预测

- 如上图所示，总收入逐渐逼近某个值，这样对于LTV（平均玩家收入），可以近似的用一个log函数曲线来反应LTV的变化趋势，我们用以下函数来表示：

$$y = c * LN(x) + b$$

- 接下来，利用前期LTV值求出系数c、b值，利用excel轻松实现：
  - c= INDEX(LINEST(Known Ys, LN(Known Xs)), 1)
  - b= INDEX(LINEST(Known Ys, LN(Known Xs)), 2)

方法一：利用前期LTV进行预测

- 下表某款游戏新增用户在前7日的LTV实际值：

$LTV_{(1)}$	0.412	$LTV_{(2)}$	0.543
$LTV_{(3)}$	0.603	$LTV_{(4)}$	0.694
$LTV_{(5)}$	0.743	$LTV_{(6)}$	0.792
$LTV_{(7)}$	0.826		

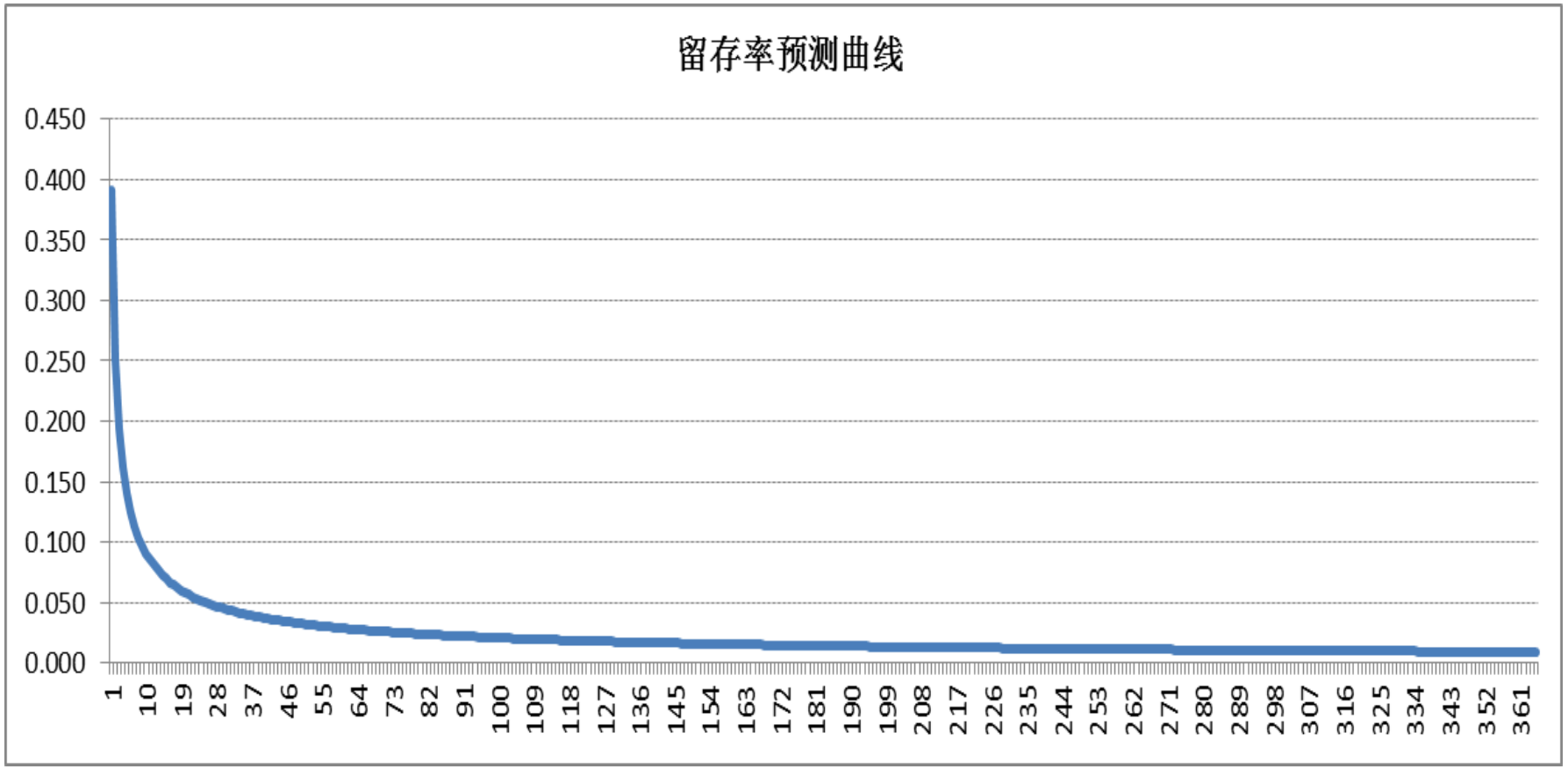
- 根据上面的公式求出c=0.214，b=0.403，所以LTV的预测函数为 y=0.214*LN(x)+0.403

方法二：利用前期留存率和APRU进行预测

- 具体步骤如下：
  - 利用指数函数对未来的留存率进行预测；
  - 计算用户生命周期LT值：LT(n)=R(1)+R(2)+…+R(n)；
  - 计算出玩家的价值ARPU，由于玩家不同日期的ARPU值可能不同，可以求出一段时间内每日APRU值得平均值，例如取玩家14天的平均ARPU，即ARPU=（ARPU(1)+ARPU(2)+…+ARPU(14))/14；
  - 从而计算出玩家的生命周期价值LTV=LT*ARPU=(R(1)+R(2)+…+R(n))*ARPU。

方法二：利用前期留存率和APRU进行预测

- 可见，根据前期已经留存率对给定自然时间内的留存率进行预测是问题的关键。经研究方发现，一批新增玩家的留存率随自然时间的增长必然无限趋近于0，用几何图形表示就是一条幂减的指数曲线。



方法二：利用前期留存率和APRU进行预测

- 假设指数曲线的函数如下所示：y=a*x^b

- 接下来，利用前期留存率求出系数a、b值，利用excel轻松实现
  - a =EXP(INDEX(LINEST(LN(Known Ys), LN(Known Xs)), 2))
  - b=INDEX(LINEST(LN(Known Ys), LN(Known Xs)), 1)

目录

关联规则的定义

- 从事务数据库，关系数据库和其他信息存储中的大量数据的项集之间发现有趣的、频繁出现的模式、关联和相关性。
- 更确切的说，关联规则通过量化的数字描述物品甲的出现对物品乙的出现有多大的影响。
- 关联规则模式属于描述型模式，发现关联规则的算法属于无监督学习的方法。
- 应用： 购物篮分析、交叉销售、产品目录设计、聚集、分类等
- 两种策略：
  - 商品放近， 增加销量
  - 商品放远， 增加其他商品的销量

关联规则的基本概念

- 在关联规则所使用的数据中，把一个样本称为一个“事务”(Transaction)；
- 每个事务由多个属性来确定，这里的属性称为**“项”(Item)**，多个项组成的集合称为**“项集”(Itemset);**
- 根据项集中的包含项的数量，项集可以是1-项集，2-项集或者k-项集。
- 用X表示一个项或者项集，Y表示与X没有交的另一个项或项集，那么记号 X => Y表示X和Y同时出现一个规则(rule)。在 X => Y中，称X为前项(也称为条件项或左项），而Y称为后项（也称为结果项或右项）。

关联规则的基本概念

- 在进行关联规则挖掘之前，由用户预先定义**最小支持度阈值(min_sup)和最小置信度阈值(min_conf)**。我们一般并不是关心所有的规则，而只是对那些重要的规则感兴趣，它们都是支持度大于等于min_sup,置信度大于等于min_conf的规则，这些规则称为“强规则”。

- 如果某个项集的支持度大于等于设定的最小支持度阈值min_sup，称这个项集为“**频繁项集**”(也称为“大项集”，LargeItemsets)，所有的“频繁k-项集”组成的集合通常记为Lk。

支持度、置信度、提升度

- 假设共有1000人购买了物品（事务集W），其中购买了5000金币的有100人，购买了8000金币的有150人，同时购买了5000金币和8000金币的有80人。

1. 支持度是交易集同时包含 X 和 Y 的交易数与事务集 W 之比。

   support(X⇒Y)=count(X∩Y)/W=80/1000=8%

2. 可信度是指包含 X 和 Y 的交易数与包含 X 的交易数之比。

   confidence(X⇒Y)=support(X⇒Y)/support(X)=80/100=80%

   也就是说有 80%的用户在购买了 5000 金币之后还会购买 8000 金币。

3. 提升度就是在购买 X 物品的前提下购买 Y 物品的可能性与没有购买 X 物品的情况下购买 Y 物品的可能性之比。

   lift(X⇒Y)= confidence(X⇒Y)/support(Y)=80%/15%=5.34

关联规则的R语言实现

- 应用arules包添加包中的函数apriori( )

apriori(data, parameter = NULL, appearance = NULL, control = NULL)

参数	解释
data	含有交易数据的稀疏矩阵
parameter	设置参数，默认情况下 parameter=list(supp=0.1,conf=0.8,maxlen=10,minlen=1,target=”rules”)

supp：支持度（support）
conf：置信度（confidence）
maxlen,minlen：每个项集所含项数的最大最小值
target：“rules”或“frequentitemsets”（输出关联规则或者频繁项集）

其他主要函数

- 检验关联规则：
  inspect(x,...)
  x:是由apriori()函数给出的一组关联规则。
  它将输出关联规则到屏幕。可以用x运用向量运算来选择查看一个或者多个特定的规则。
- 对关联规则集合排序：
  根据(市场)购物篮分析的目标，最有用的规则或许是那些具有最高支持度、置信度和提升度的规则。arules添加包包含一个sort()函数，可以用来对规则列表重新排序，从而那些具有最高或者最低度量值的规则将会排在第一位。
- 提取关联规则的子集：
  subet(x,subset,...)函数提供了一种用来寻找交易、商品(项)或者规则子集的方法。
  x:要取子集的对象；subset:条件逻辑表达式。
- transactions：事务型数据集：
  将数据转换成arules关联规则方法apriori可以处理的数据形式。
- 画频繁项的图：
  itemFrequencyPlot( )函数可以描绘所包含的特定商品的交易比例的柱状图。

业务案例：对玩家物品购买关联分析

原始数据

ver_id	product_name	qty
7204535	感恩大礼包	1
107204535	新手礼包	1
213666611	8条钥匙	1
278620434	15000金币	1
278620434	70000金币	1
278620434	快速复活	2
...	...	...

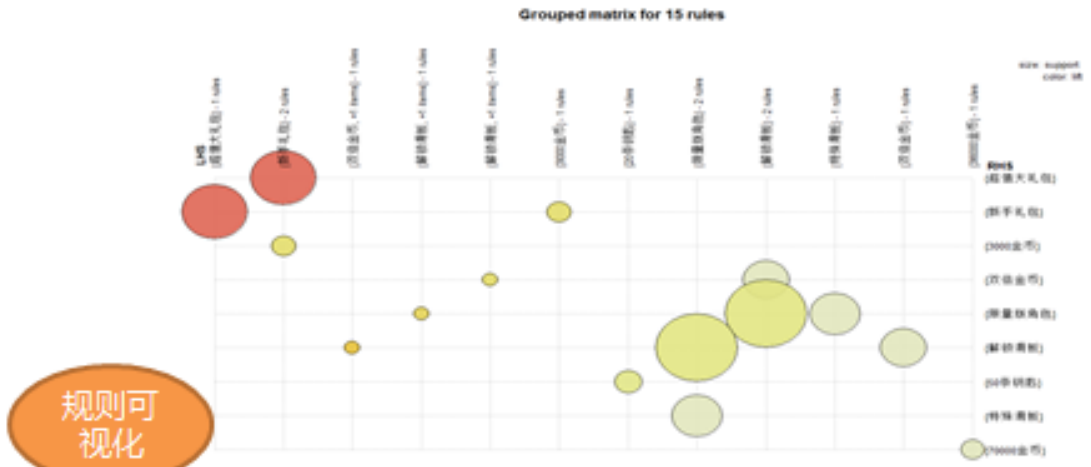
转换数据

	items	transactionID
1	{感恩大礼包,新手礼包}	107204535
2	{8 条钥匙}	213666611
3	{0.1 元大礼包,8 条钥匙,限量版角色}	226500629
4	{38000 金币,限量版角色,新手礼包}	230329140
5	{50 条钥匙}	264162836
6	{15000 金币,70000 金币,快速复活}	278620434

生成规则

```
> # 对规则按照提升度排序，并输出提升度最大的前六条规则
> inspect(sort(rules,by="lift")[1:6])
    lhs                    rhs                  support     confidence lift
6   {超值大礼包}           => {新手礼包}        0.015994882 0.9009009  13.190708
7   {新手礼包}             => {超值大礼包}      0.015994882 0.2341920  13.190708
50  {双倍金币,限量版角色} => {解锁滑板}        0.005918106 0.3523810  5.737202
48  {解锁滑板,双倍金币}    => {限量版角色}      0.005918106 0.4933333  4.213552
49  {解锁滑板,限量版角色} => {双倍金币}        0.005918106 0.3083333  3.524132
8   {3000 金币}            => {新手礼包}        0.007677543 0.2330097  3.411655
```

规则可视化



天善智能 TIANSHAN SOFT

目录

基于玩家物品的智能推荐

- 关联规则只能反映一个事物与其他事物之间的关联性，有时候，我们想根据玩家的兴趣特点和购买行为，向玩家推荐他们感兴趣的信息和道具。

- 一个好的推荐系统能够为玩家提供个性化服务，增强用户黏性。

- 智能推荐的方法有很多，包括基于内容推荐、协同过滤推荐、基于规则推荐、基于效用推荐和基于知识推荐。各种推荐算法都有其优缺点。

推荐算法其优缺点

推荐算法	优点	缺点
基于内容推荐	推荐结果直观，容易解释； 不需要领域知识	新用户问题； 复杂属性不好处理； 要有足够数据构造分类器
协同过滤推荐	新异兴趣发现、不需要领域知识； 随着时间推移性能提高； 推荐个性化、自动化程度高； 能处理复杂的非结构化对象	可扩展性问题； 新用户问题； 质量取决于历史数据集； 系统开始时推荐质量差
基于规则推荐	能发现新兴趣点； 不需要领域知识	规则抽取难、耗时； 产品名同义性问题； 个性化程度低
基于效用推荐	无冷启动和稀疏问题； 对用户偏好变化敏感； 能考虑非产品特性	用户必须输入效用函数； 推荐是静态的，灵活性差； 属性重叠问题
基于知识推荐	能把用户需求映射到产品上； 能考虑非产品属性	知识难获得； 推荐是静态的

由于各种推荐算法都有优缺点，所以在实践中，组合推荐经常被采用。研究应用最多的是内容推荐和协同过滤推荐的组合。

推荐算法R语言实现

- 在R语言中，常使用recommenderlab包中的函数构建和评估智能推荐模型。

- Recommender函数：构建推荐模型

  Recommender(data, method, parameter=NULL)

  其中data为一个ratingMatrix，method的选项包括包括IBCF(基于物品的协同过滤推荐)、UBCF(基于　用户的协同过滤推荐)、SVD(矩阵因子化)、PCA(主成分分析)、RANDOM(随机推荐)、POPULAR(基于流行度的推荐)

- predict函数：预测推荐模型，得到模型的topN列表或者用户的预测评分。

  predict(object, newdata, n = 10, data=NULL, type="topNList", ...)

  其中object为recommender函数生成的推荐模型；newdata为待预测的数据；n为topN的值，默认为10，表示top10推荐；type的参数有"topNList"、"ratings"，当type="topNList"时，predict函数直接返回用户评分最高的前N个item，当type="ratings"时，predict函数预测用户对所有未评分的item打分，返回一个RatingMatrix对象。

推荐算法评估

- evaluationScheme ()函数

  evaluationScheme(data, method="split", train=0.9, k=NULL, given, goodRating = NA)

  功能：创建一个数据集的评价方案。该方案可以简单地分为训练数据和测试数据，n折交叉验证或bootstrap重复抽样。

- evaluate()函数

  evaluate(x, method, type="topNList",n=1:10, parameter=NULL,...)

  功能：评价一个或一系列的推荐模型，给出一个评价方案。

- plot()函数

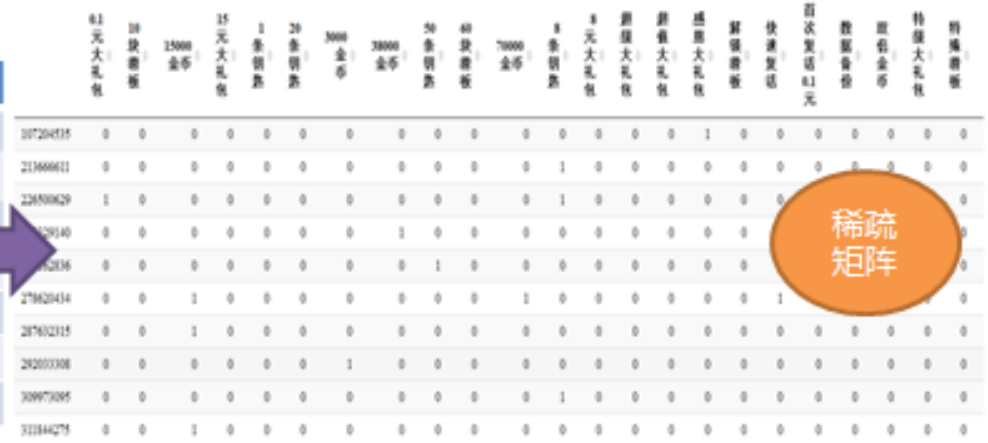
  plot(x, y,xlim=NULL, ylim=NULL, col = NULL, pch = NULL, lty = 1,avg = TRUE, type = "b", annotate= 0, legend="bottomright", ...)

功能：画出ROC曲线和PR曲线，用于推荐模型的评价。

业务案例：对玩家物品购买进行智能推荐

原始数据

layer_id	product_name	qty
107204535	感恩大礼包	1
107204535	新手礼包	1
213666611	8条钥匙	1
278620434	15000金币	1
278620434	70000金币	1
278620434	快速复活	2
...	...	...



稀疏矩阵

推荐结果

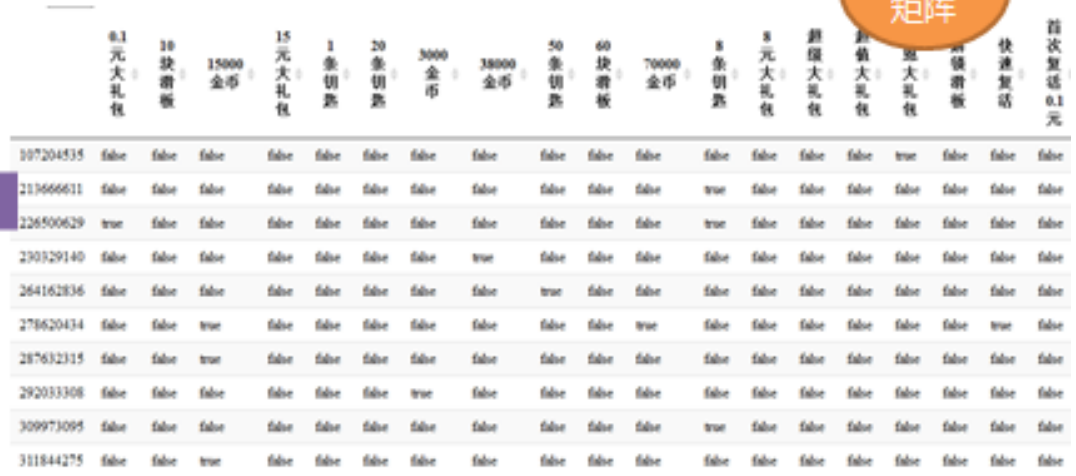
```
model.best <- Recommender(data_class,method="IBCF")
# 使用precict函数，对玩家进行Top3推荐
> data.predict <- predict(model.best,newdata=data_class,n=5)
> recom3 <- bestN(data.predict,3)
> as(recom3,"list")[1:5] #查看前五个玩家的top3推荐
$`107204535`
[1] "超值大礼包" "3000金币"    "超级大礼包"

$`213666611`
[1] "15000金币" "20条钥匙"

$`226500629`
[1] "解锁滑板" "双倍金币" "特殊滑板"

$`230329140`
[1] "超值大礼包" "解锁滑板"    "双倍金币"

$`264162836`
[1] "70000金币" "20条钥匙"   "15000金币"
```



评分矩阵