第五章 大数据用户特征分析

方杰、李烜

福建江夏学院金融学院

本章内容

1. 用户特征的 RFM 分析
   - RFM 模型的基本原理
   - RFM 模型的实施步骤
   - RFM 的 Python 实现
2. 用户特征的 K-means 分析
   - 聚类算法
   - K-Means 聚类算法
   - K-Means 算法的实现

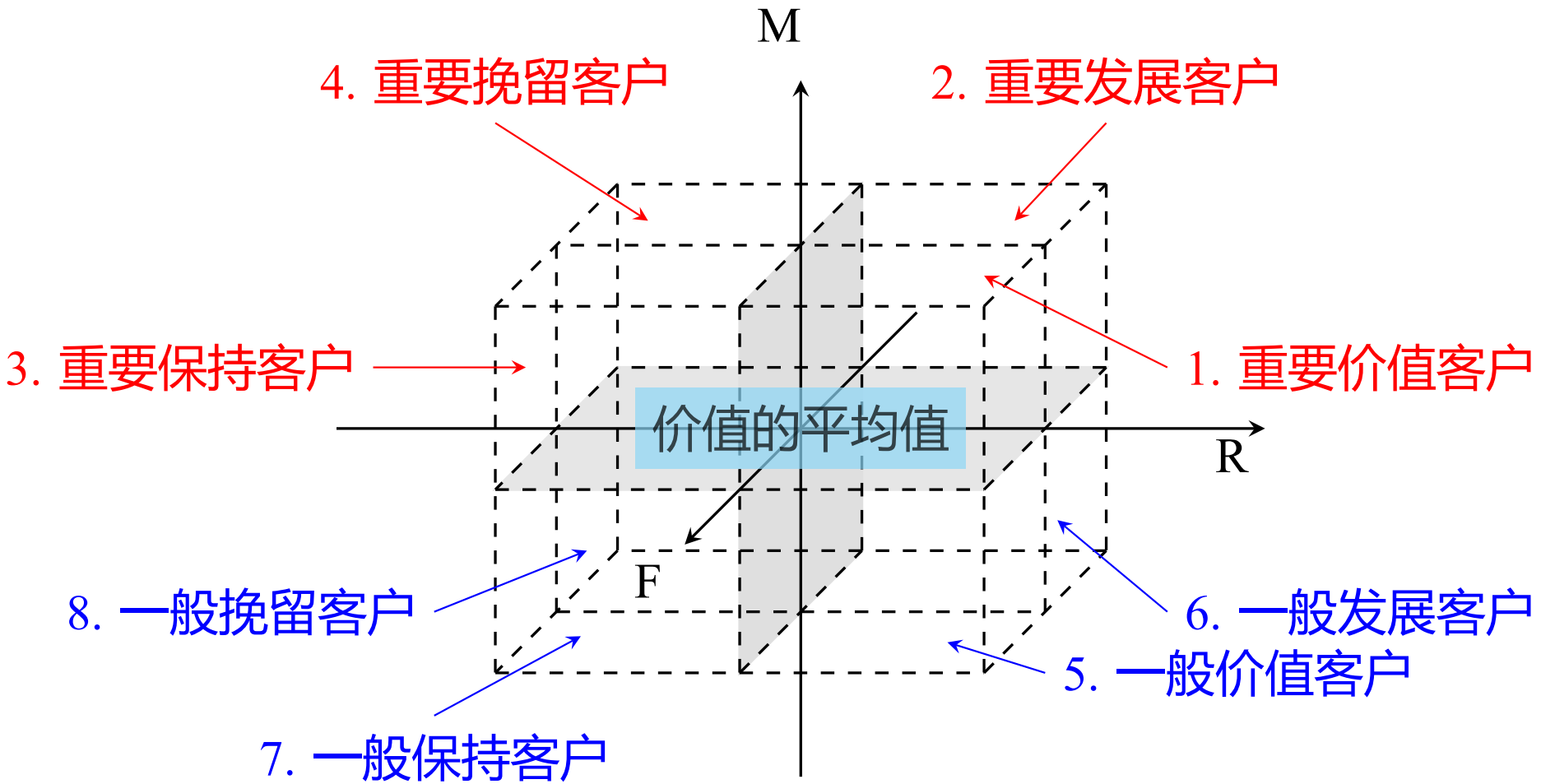
RFM 模型

如何将用户从一个整体拆分成特征明显的群体是精细化运营的关键。RFM 模型是衡量用户价值和用户创利能力的经典工具。

- R (Recency) 最近一次消费时间间隔
- F (Frequency) 消费频率
- M (Monetary) 消费金额

基于这三个维度，每个维度都可以取高或低两种值，可以构造出一个三维的坐标系。由此依托于用户最近一次消费时间、消费频次以及消费金额可以将客户划分为 8 类价值群体。

RFM 模型


M
4. 重要挽留客户
2. 重要发展客户
3. 重要保持客户
1. 重要价值客户
价值的平均值
R
F
8. 一般挽留客户
6. 一般发展客户
5. 一般价值客户
7. 一般保持客户

RFM 模型的基本原理

客户类型	R	F	M	客户场景
重要价值客户	高	高	高	RFM 都很大，优质客户，客户价值高
重要发展客户	高	低	高	交易金额大贡献度高，且最近有交易，需要重点识别
重要保持客户	低	高	高	交易金额和次数大，但最近无交易，需要保持联系
重要挽留客户	低	低	高	交易金额大，潜在的有价值客户，需要挽留
一般价值客户	高	高	低	交易次数大，最近有交易，需要挖掘
一般发展客户	高	低	低	最近有交易，是接触的新客户，有推广和发展的价值
一般保持客户	低	高	低	交易次数多，但是贡献不大，一般保持
一般挽留客户	低	低	低	RFM 都很低，相当于流失客户

RFM 模型的实施步骤

- 确定 RFM 的得分值：基于指标数据计算出 R、F、M 三个指标下可以用于建模计算的，统一量纲下的数值
- 确定 RFM 的评分标签：获得 RFM 的得分数值后，将 R、F、M 三个维度下的数据，使用“高”“低”两种取值进行表示，也就是“打标签”

确定 RFM 的得分值

R、F、M 的计算公式

$$\begin{cases} R = T_{\text{time}} - T_{\text{last}} \\ F = Count(\text{orderID}) \\ M = \sum \text{Price} \end{cases}$$

- R 计算的是现在的时间减去用户上一次消费的时间
- F 计算的是单个用户在一定时间内产生的所有订单的总数
- M 计算的是单个用户所有交易的总金额

注意：

RFM 得分值要注意数据间存在的量级差距，结合归一化（normalize）处理，使得得分处于同一尺度上，以利于评价

确定 RFM 的评分标签

- 均值 RFM 模型：通过与均值的比较进行 RFM 指标的打分
- 聚类 RFM 模型：先用聚类的算法对指标进行聚类，得到 m 个客户簇。再将每簇的 R、F、M 平均值和总 R、F、M 平均值作比较
- 动态 RFM 模型：结合业务人员的经验和大数据分析的结果来划分分界线，而不是采用静态的均值

打标签的 Python 实现

以浦发银行收益率数据为例，对股票收益率的正负情况“打标签”

```
# 浦发银行日收益率数据以dataframe格式存储在变量“df”中。
df['tag']= df['returns'].apply(lambda x: 'positive' if x >0
                                    else 'negative')
print(df)
```

lambda 匿名函数的格式

```
lambda 参数列表 : 表达式
```

lambda 匿名函数的使用，可以避免对函数进行定义和封装，使代码更为简洁

RFM 模型得分计算

使用数据透视表函数对数据源进行汇总计算

```
import pandas as pd
pd.pivot_table(data, values, index, aggfunc)
```

- data DataFrame 格式的数据源
- values 指定需要汇总计算的列的列索引值。若多个值，可以使用列表格式
- index 生成的数据透视表行索引
- aggfunc 聚合函数或聚合列表，可使用字典结构分别指定

可用的聚合函数

```
np.max, np.min, np.mean, np.size, np.sum
```

举例：以客户 ID 为条件进行筛选，并利用透视表功能计算

```
import numpy as np
import pandas as pd
# 读取数据
df = pd.read_excel('order_merge.xlsx')
# 计算R、F、M得分值
RFM = pd.pivot_table(df, values=['timestamp',
        'order_id', 'price'], index='customer_unique_id',
        aggfunc={'timestamp' : np.max,
        'order_id' : np.size, 'price' : np.sum})
```

举例：重命名及数据转换

```
# 为了便于理解和简化代码，先将各列名重命名为R、F、M
RFM = RFM.rename(columns={'timestamp' : 'R',
                          'order_id' : 'F', 'price': 'M'})
# 计算2023年1月1日与客户最后一次购买时间的间隔秒数
RFM['R'] = (pd.to_datetime('2023-01-01 00:00:00')
            - RFM['R']).dt.total_seconds()
```

说明

dt.total_seconds函数可用于将时间转化为秒数

举例：运用均值 RFM 模型打标签及结果分析

```
RFM_mean = RFM[['R', 'F', 'M']].mean()
RFM['R1'] = RFM['R'].apply(lambda x:
                           0 if x > RFM_mean['R'] else 1)
RFM['F1'] = RFM['F'].apply(lambda x:
                           1 if x > RFM_mean['F'] else 0)
RFM['M1'] = RFM['M'].apply(lambda x:
                           1 if x > RFM_mean['M'] else 0)
```

最后，根据消费者群体划分的分类要求，对客户群体特征进行分析。见 P263

聚类算法的概况

聚类算法是客户特征分析的一种经典模型，被广泛应用于客户画像、广告推荐、基于位置的商业推送、信用卡异常消费识别等领域

RFM 模型因其数据获取方便，模型成熟度好，被广泛运用在客户画像分析中。但 RFM 也存在一定的缺陷，比如其研究的变量固定，难以满足决策者因市场环境而变化的需求。此外，由于 RFM 数据间可能存在较强相关性，最终得出的分类结果可能不理想

在大数据实际应用中，许多领域会采用聚类算法对客户特征进行分析

聚类算法

聚类 (clustering) 算法是一种典型的无监督学习 (unsupervised learning) 算法，主要用于将相似的样本自动归到一个类别中。

聚类和分类的区别

- 聚类 (clustering) 是指把相似的数据划分到二起，聚类的时候并不关心这一类的标签，目标就是把相似的数据聚合到一起。二个聚类算法通常只需要知道如何计算相似度就可以开始工作了，并不需要使用训练集数据进行学习，这在机器学习中被称为无监督学习。
- 分类 (classification) 是把不同的数据划分开，其过程是通过训练数据集获得个分类器，再通过分类器去预测未知数据，分类是一种监督学习方法。

聚类算法的分类

- 划分式聚类算法：大部分划分方法是基于距离的
  代表算法有：K-means 及其变体 K-means++，K-medoids，kernel K-means 等
- 基于密度的聚类：核心思想在于只要一个区域中点的密度大于某个阈值，就把它加到与之相近的聚类中去
  代表算法有：DBSCAN，OPTICS，DENCLUE，WaveCluster 等。
- 基于分层的聚类：能降低前两者的可能存在的链式效应，只适合在数据量小的时候使用，数据量大的时候速度会较慢

K-Means 算法的原理

K-Means 算法是典型的基于距离的非层次聚类算法

基本步骤

1. 创建 K 个点作为初始各类簇的中心，通常是随机选择;
2. 计算每个聚类对象到聚类中心的距离来划分类簇;
3. 根据重新划分的类簇，更新每个类簇的中心;
4. 计算标准测度函数，直到达到最大迭代次数则停止，否则继续操作。

K-means 算法的优缺点

优点

1. 简单、易用
2. 收敛速度快，聚类效果优
3. 算法可解释性强

缺点

1. K 值的选取不好把握
2. 算法对噪音和异常点比较的敏感
3. 由于选择了距离作为相似度划分依据，因此只能发现球型类簇

K-means 聚类个数选择

- 通过观测数据特性进行聚类个数的判定过于主观且有很大不确定性，通常通过手肘法则 (The Elbow Method) 寻找最佳聚类个数 K
- 手肘法的核心思想在于通过误差平方和 SSE 降低速度的减缓判定出真实聚类数。聚合效果会降低，反映为 SSE 降低的速度减缓，因此可以通过 SSE 降低速度的减缓判定出真实聚类数
- Python 的`Sklearn`库中的`KMeans`函数能够计算给定聚类个数的误差项，故通过循环计算不同聚类个数下模型的误差项并作图判断最佳聚类个数

K-mean 模型构建

```
from sklearn.cluster import KMeans
algorithm=(KMeans(n_clusters, init, n_init, max_iter,
                  random_state=None, algorithm='auto'))
```

- n_clusters 要形成的聚类数以及要生成的质心数，默认值为 8
- init 初始化方法，默认为'k-means++'
- n_init 使用不同质心种子运行的次数，默认为 10
- max_iter 单次运行算法的最大迭代次数，默认为 300
- random_state 确定质心初始化的随机数生成方式
- algorithm 确定算法风格

例：鸢尾花 (iris) 数据导入

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.cluster import KMeans
from sklearn.datasets import load_iris
iris = load_iris()
X = iris.data[:]
df = pd.DataFrame(X)
```

确定类簇个数

```
from sklearn.cluster import KMeans
# 剥离出鸢尾花数据的前3个特征，构成数据X3
X3 = df[[0,1,2]].values
inertia = []
for n in range(1, 11):
    algorithm=(KMeans(n_clusters=n, init='k-means++',
                      random_state=111, algorithm='elkan'))
    algorithm.fit(X3) # 按样本数据X3拟合聚类函数
    inertia.append(algorithm.inertia_)
```

确定类簇个数 (cont.)

通过作图观察寻找最佳聚类个数 K

```
import matplotlib.pyplot as plt
import seaborn as sns
plt.figure(1,figsize = (15,6))
plt.plot(np.arange(1,11), inertia, 'o')
plt.plot(np.arange(1,11), inertia, '-', alpha=0.5)
plt.xlabel('Number of Clusters')
plt.ylabel('Inertia')
plt.show()
```

K-means 模型构建

```
algorithm = (KMeans(n_clusters=3, init='k-means++',
             n_init=10, max_iter=300, tol=0.0001,
             random_state=111, algorithm='elkan') )
algorithm.fit(X3) # 将鸢尾花数据X3代入
labels3 = algorithm.labels_ # 获得每个客户所在的类别
centroids3 = algorithm.cluster_centers_ # 获得各类别的中心点
df['label3'] = labels3
```