# RFM案例

# 一、会员价值度模型

## 1、RFM模型介绍

- 会员价值度用来评估用户的价值情况，是区分会员价值的重要模型和参考依据，也是衡量不同营销效果的关键指标之一。
- 价值度模型一般基于交易行为产生，衡量的是有实体转化价值的行为。常用的价值度模型是RFM
- RFM模型是根据会员
    - 最近一次购买时间R（Recency）
    - 购买频率F（Frequency）
    - 购买金额M（Monetary）
    - 计算得出RFM得分
- 通过这3个维度来评估客户的订单活跃价值，常用来做客户分群或价值区分
    - RFM模型基于一个固定时间点来做模型分析，不同时间计算的的RFM结果可能不一样

| R | F | M | 用户类别 |
|---|---|---|---|
| 高 | 高 | 高 | 重要价值用户 |
| 高 | 低 | 高 | 重要发展用户 |
| 低 | 高 | 高 | 重要保持用户 |
| 低 | 低 | 高 | 重要挽留用户 |
| 高 | 高 | 低 | 一般价值用户 |
| 高 | 低 | 低 | 一般发展用户 |
| 低 | 高 | 低 | 一般保持用户 |
| 低 | 低 | 低 | 一般挽留用户 |

## 2、RFM模型实现过程

**设置截止时间节点**：选择一个基准日期（如2024-12-29）作为计算Recency的参考点。

**获取原始数据集**：从会员数据库中提取过去一年内的订单数据，包括会员ID、订单时间和订单金额。

**数据预处理**：

- **Recency**：计算每个会员距离截止日期最近的订单时间。
- **Frequency**：统计每个会员的订单总次数。
- **Monetary**：计算每个会员的订单总金额。

**R、F、M 分区**：

- 使用五分位法（1-5分）对R、F、M三个指标进行分区。
- Recency分数越小，得分越高（接近截止日期），Frequency和Monetary则相反。

**计算RFM得分**：

- **加权得分**：根据业务需求赋予R、F、M不同权重并计算总分。
- **组合得分**：将R、F、M得分拼接成一个字符串（如312、555）。

**导出结果**：将RFM得分保存为CSV文件，以便后续分析。

三大类：最优价值 中等价值 低质用户

# 二、代码实现

## 1、导入所需的库

```
import numpy as np
import pandas as pd
```

## 2、导入数据

```
# 导入数据
df_raw = pd.DataFrame(pd.read_excel('./sales.xlsx',index_col='USERID'))
```

## 3、缺失值处理

```
# 缺失值处理
sales_data = df_raw.dropna()  # 丢失带有缺失值NA的行记录
# 丢弃订单金额<=1的记录
sales_data = sales_data[sales_data['AMOUNTINFO'] > 1]
```

## 4、数据转换 (按用户ID去重归总)

```
# 数据转换 (按用户ID去重归总)
recency_value = sales_data['ORDERDATE'].groupby(sales_data.index).max()  # 计算最近一次订单时间
frequency_value = sales_data['ORDERDATE'].groupby(sales_data.index).count()  # 计算订单频率
monetary_value = sales_data['AMOUNTINFO'].groupby(sales_data.index).sum()  # 计算订单总金额
```

## 5、分别计算 R, F, M 得分

```
pandas.cut() 方法用于将连续的数据分段（离散化）。
它通常用于将数值型数据分成不同的区间或类别。以下是 pandas.cut 方法的一些关键参数和返回值
    关键参数
        x: 要分段的数组或 Series。
        bins: 分段的数量或具体的分段边界。
        right: 是否包括右端点，默认为 True。
        labels: 每个分段的标签，可以是字符串或数字。
        ...
    返回值
        pd.cut 返回一个 Categorical 类型的 Series，其中每个元素对应于输入数据中的一个区间标签。
```

```
# 分别计算 R, F, M 得分
deadline_date = pd.to_datetime("2024-12-29")  # 指定一个时间节点，用来计算其他时间和该时间的距离
r_interval = (deadline_date - recency_value).dt.days  # 计算 R 间隔
r_score = pd.cut(r_interval, 5, labels=[5, 4, 3, 2, 1])  # 计算 R 得分，五分位倒序
f_score = pd.cut(frequency_value, 5, labels=[1, 2, 3, 4, 5])  # 计算 F 得分
m_score = pd.cut(monetary_value, 5, labels=[1, 2, 3, 4, 5])  # 计算 M 得分
```

# 6、R, F, M 数据合并

```
# R, F, M 数据合并
rfm_list = [r_score, f_score, m_score]  # 将 R, F, M 三个维度组成列表
rfm_cols = ['r_score', 'f_score', 'm_score']  # 设置 R, F, M 三个维度的列名
rfm_pd = pd.DataFrame(np.array(rfm_list).transpose(), dtype=np.int32,
columns=rfm_cols, index=frequency_value.index)  # 建立 R, F, M 数据框
```

注意: 其中transpose()类似于T属性,目的是行列转换

> np.array(rfm_list)效果是:
>
> [[5 5 5 5 5 1]
> [1 5 1 1 1 1]
> [1 3 3 4 5 1]]

> np.array(rfm_list).transpose()效果是:
>
> [[5 1 1]
> [5 5 3]
> [5 1 3]
> [5 1 4]
> [5 1 5]
> [1 1 1]]

> np.array(rfm_list).T 效果是:
>
> [[5 1 1]
> [5 5 3]
> [5 1 3]
> [5 1 4]
> [5 1 5]
> [1 1 1]]

> 最终rfm效果是:
>
> r_score f_score m_score
>
> USERID
> 1 5 1 1
> 2 5 5 3
> 3 5 1 3
> 4 5 1 4
> 5 5 1 5
> 6 1 1 1

# 7、策略1：加权得分 定义用户价值

```
# 策略1：加权得分 定义用户价值
rfm_pd['rfm_wscore'] = rfm_pd['r_score']*0.2 + rfm_pd['f_score']*0.2 +
rfm_pd['m_score']*0.6
```

根据加权得分 `rfm_wscore`，你可以将客户划分为不同的群体。划分标准可以根据得分范围来定义，以下是一个常见的划分方式：

- **高价值客户**：`rfm_wscore` 高于某个阈值，例如 4.0 到 5.0。这类客户在最近购买、频率高、消费金额大，属于最重要的客户群体。
- **中等价值客户**：`rfm_wscore` 介于中间范围，例如 2.0 到 4.0。这类客户表现较好，但在某些维度（如频率或金额）可能稍有不足。
- **低价值客户**：`rfm_wscore` 较低，例如低于 2.0。这类客户可能消费金额不高，购买频率低或最近很少购买，通常不是业务的主要目标群体。

# 8、策略2：RFM组合 直接输出三维度值

```
# 策略2：RFM组合 直接输出三维度值
rfm_pd_tmp = rfm_pd.copy()
rfm_pd_tmp['r_score'] = rfm_pd_tmp['r_score'].astype('str')
rfm_pd_tmp['f_score'] = rfm_pd_tmp['f_score'].astype('str')
rfm_pd_tmp['m_score'] = rfm_pd_tmp['m_score'].astype('str')
rfm_pd['rfm_comb'] =
rfm_pd_tmp['r_score'].str.cat(rfm_pd_tmp['f_score']).str.cat(rfm_pd_tmp['m_score
'])
```

后期划分客户类别

通过 `RFM` 组合得分，将客户划分为不同的群体。具体群体的划分取决于 `RFM` 组合得分的含义：

**高价值客户**：

- R 得分高：客户最近购买。
- F 得分高：客户购买频率高。
- M 得分高：客户消费金额高。
- 例如，RFM 组合是 "555"、"554"、"545" 等。

**中价值客户**：

- R 得分中等：客户购买时间适中，不是最近但也不是很久之前。
- F 得分中等：客户购买频率一般。
- M 得分中等或较高：客户消费金额中等。
- 例如，RFM 组合是 "343"、"442"、"453" 等。

**低价值客户**：

- R 得分低：客户很久没有购买。
- F 得分低：客户购买频率低。
- M 得分低或中等：客户消费金额较低。
- 例如，RFM 组合是 "111"、"221"、"132" 等。

# 9、导出数据

```
# 导出数据
rfm_pd.to_csv('rfm_result.csv')
```

# 最终完整代码

```
import numpy as np
import pandas as pd

# 导入数据
df_raw = pd.DataFrame(pd.read_excel('./dataset/sales.xlsx',index_col='USERID'))

# 缺失值处理
sales_data = df_raw.dropna() # 丢失带有缺失值NA的行记录
sales_data = sales_data[sales_data['AMOUNTINFO'] > 1] # 丢弃订单金额<=1的记录

# 数据转换 (按用户id去重归总)
recency_value = sales_data['ORDERDATE'].groupby(sales_data.index).max() #计算最近一次订单时间
frequency_value = sales_data['ORDERDATE'].groupby(sales_data.index).count() #计算订单频率
monetary_value = sales_data['AMOUNTINFO'].groupby(sales_data.index).sum() #计算订单总金额

# 分别计算R,F,M得分
deadline_date = pd.to_datetime("2020-05-01") #指定一个时间节点，用来计算其他时间和改时间的距离
r_interval = (deadline_date - recency_value).dt.days #计算r间隔
r_score = pd.cut(r_interval, 5, labels=[5,4,3,2,1]) # 计算r得分 五分位倒序
f_score = pd.cut(frequency_value, 5, labels=[1,2,3,4,5]) # 计算f得分
m_score = pd.cut(monetary_value, 5, labels=[1,2,3,4,5]) # 计算m得分

# R,F,M数据合并
rfm_list = [r_score, f_score, m_score] # 将R,F,M三个维度组成列表
rfm_cols = ['r_score', 'f_score', 'm_score'] # 设置R,F,M三个维度的列名
rfm_pd = pd.DataFrame(np.array(rfm_list).transpose(), dtype=np.int32,
columns=rfm_cols, index=frequency_value.index) #建立R,F,M数据框

#策略1：加权得分 定义用户价值
rfm_pd['rfm_wscore'] = rfm_pd['r_score']*0.2 + rfm_pd['f_score']*0.2 +
rfm_pd['m_score']*0.6

#策略2：RFM组合 直接输出三维度值
rfm_pd_tmp = rfm_pd.copy()
rfm_pd_tmp['r_score'] = rfm_pd_tmp['r_score'].astype('str')
rfm_pd_tmp['f_score'] = rfm_pd_tmp['f_score'].astype('str')
rfm_pd_tmp['m_score'] = rfm_pd_tmp['m_score'].astype('str')
rfm_pd['rfm_comb'] =
rfm_pd_tmp['r_score'].str.cat(rfm_pd_tmp['f_score']).str.cat(rfm_pd_tmp['m_score
'])

# 导出数据
rfm_pd.to_csv('rfm_result.csv')
```

# 三、图形可视化

区间 `[0, 2)` 表示从 `0` 开始到 `2` 之前，`0` 包含在内，而 `2` 不包含在内。

区间 `[2, 4)` 表示从 `2` 开始到 `4` 之前，`2` 包含在内，而 `4` 不包含在内。

区间 `[4, 5]` 表示从 `4` 开始到 `5` 结束，`4` 和 `5` 都包含在内。

# 1、柱形图

```
import matplotlib.pyplot as plt

# 根据加权得分划分客户群体
bins = [0, 2, 4, 5]  # 自定义分层阈值
labels = ['Low', 'Medium', 'High']  # 分层标签
rfm_pd['Customer Segment'] = pd.cut(rfm_pd['rfm_wscore'], bins=bins,
labels=labels)

# 绘制柱状图
rfm_pd['Customer Segment'].value_counts().plot(kind='bar', color=['red',
'orange', 'green'])
plt.title('Customer Segments based on RFM Score')
plt.xlabel('Segment')
plt.ylabel('Number of Customers')
plt.show()
```

# 2、饼状图

```
import matplotlib.pyplot as plt

# 根据加权得分划分客户群体
bins = [0, 2, 4, 5]  # 自定义分层阈值
labels = ['Low', 'Medium', 'High']  # 分层标签
rfm_pd['Customer Segment'] = pd.cut(rfm_pd['rfm_wscore'], bins=bins,
labels=labels)

# 绘制饼状图
rfm_pd['Customer Segment'].value_counts().plot(kind='pie',
                                               colors=['red', 'orange',
'green'],
                                               autopct='%1.1f%%',  # 显示百分比
                                               startangle=90,      # 旋转起始角度
                                               counterclock=False) # 顺时针方向
plt.title('Customer Segments based on RFM Score')
plt.ylabel('')  # 隐藏Y轴标签
plt.show()
```