

Broadview®
www.broadview.com.cn

行业畅销书升级版

★ 告诉你职场图表背后的故事 ★

谁说菜鸟不会 数据分析

方小敏 张文霖 著

(Python篇)



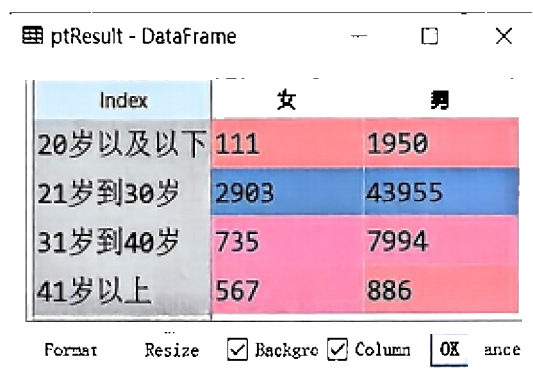
中国工信出版集团



电子工业出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

```
]
# 生成年龄分层列
data['年龄分层'] = pandas.cut(
    data.年龄,
    bins,
    labels=ageLabels
)
# 进行交叉统计，行为年龄分层，列为性别，对用户 ID 进行计数统计
ptResult = data.pivot_table(
    values='用户 ID',
    index='年龄分层',
    columns='性别',
    aggfunc='count'
)
```

执行代码，得到交叉分析的统计结果，如图 5-20 所示。可以看到，这份统计结果类似 Excel 数据透视表的统计结果。行为年龄分组，列为性别，这里只有一个统计函数 count，所以只有一个计数统计结果。



Index	女	男
20岁及以下	111	1950
21岁到30岁	2903	43955
31岁到40岁	735	7994
41岁以上	567	886

图 5-20 交叉分析的统计结果

在分布分析的案例中，已经得到“21 岁到 30 岁”年龄段的用户数最多这个结论，加入性别作为列进行交叉分析，可以看到，不仅不同年龄段之间用户数相差较大，同一个年龄段不同性别的用户数也相差较大，“21 岁到 30 岁”年龄段中，男性用户数比女性多十几倍。

5.7 RFM 分析

RFM 分析，是根据客户活跃程度和交易金额贡献，进行客户价值细分的一种客户细分方法。RFM 分析，主要由三个指标组成，分别为 R (Recency) 近度、F (Frequency) 频度、M (Monetary) 额度组成，如图 5-21 所示。

指标	解释	意义
R (Recency) 近度	客户最近一次交易的时间间隔	R越大, 表示客户越久未发生交易 R越小, 表示客户越近有交易发生
F (Frequency) 频度	客户在最近一段时间内交易的次数	F越大, 表示客户交易越频繁 F越小, 表示客户不够活跃
M (Monetary) 额度	客户在最近一段时间内交易的金额	M越大, 表示客户价值越高 M越小, 表示客户价值越低

图 5-21 RFM 指标意义

R 表示近度 (Recency), 也就是客户最近一次交易时间到现在的间隔, 注意, R 是最近一次交易时间到现在的间隔, 而不是最近一次的交易时间, R 越大, 表示客户未发生交易时间越长, R 越小, 表示客户未发生交易时间越短。

F 表示频度 (Frequency), 也就是客户在最近一段时间内交易的次数, F 越大, 表示客户交易越频繁, F 越小, 表示客户越不活跃。

M 表示额度 (Monetary), 也就是客户在最近一段时间内交易的金额, M 越大, 表示客户价值越高, M 越小, 表示客户价值越低。

这里有一张经典 RFM 客户细分模型图, 如图 5-22 所示, R 分值、F 分值和 M 分值三个指标构成了一个三维立方图, 在各自维度上, 根据得分值又可以分为高和低两个分类。最终三个指标, 每个指标分为高低两类, 两两组合, 就细分为八大客户群体。

例如 R 分值高、F 分值高、M 分值高的客户为重要价值客户, R、F、M 三个分值都低的客户为潜在客户, 其他类型客户以此类推进行解读即可。

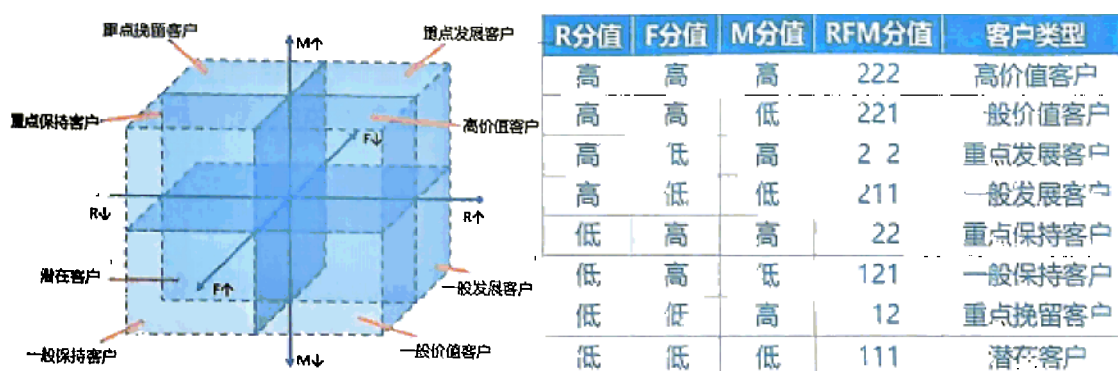


图 5-22 RFM 客户细分模型

使用 RFM 分析, 需要满足以下三点假设:

- 1) 假设最近有过交易行为的客户, 再次发生交易的可能性要高于最近没有交易行为的客户;
- 2) 假设交易频率较高的客户比交易频率较低的客户, 更有可能再次发生交易行为;

>> 谁说菜鸟不会数据分析（Python 篇）

3) 假设过去所有交易总金额较多的客户，比交易总金额较少的客户，更有消费积极性。

尽管大部分的场景都符合这三个假设条件，但在使用 RFM 分析之前，还是需要结合实际的业务场景，判断是否都符合以上三个假设条件。

RFM 分析步骤如图 5-23 所示。



图 5-23 RFM 分析步骤

在 Python 中，暂时还没有实现 RFM 分析的模块，我们可以按照 RFM 分析的步骤，自行编写代码实现。

STEP 01 数据准备

下面通过一个案例学习 RFM 分析的使用，首先将数据导入 data 变量，代码如下所示：

```
代码输入
import pandas
data = pandas.read_csv(
    'D:/PDABook/第五章/5.7 RFM 分析/RFM 分析.csv',
    engine='python'
)
```

执行代码，即可得到 data 数据框，如图 5-24 所示。可以看到，第一列为订单 ID，第二列为客户 ID，第三列为交易时间，第四列为交易金额。这个数据格式，也是 RFM 分析要求的基本数据格式。

Index	OrderID	CustomerID	DealDateTime	Sales
0	4529	34858	2014-05-14	807
1	4532	14597	2014-05-14	160
2	4533	24598	2014-05-14	418
3	4534	14600	2014-05-14	401
4	4535	24798	2014-05-14	234
5	4536	44856	2014-05-14	102
6	4558	34695	2014-05-14	130
7	4559	24764	2014-05-14	377
8	4566	34765	2014-05-15	466

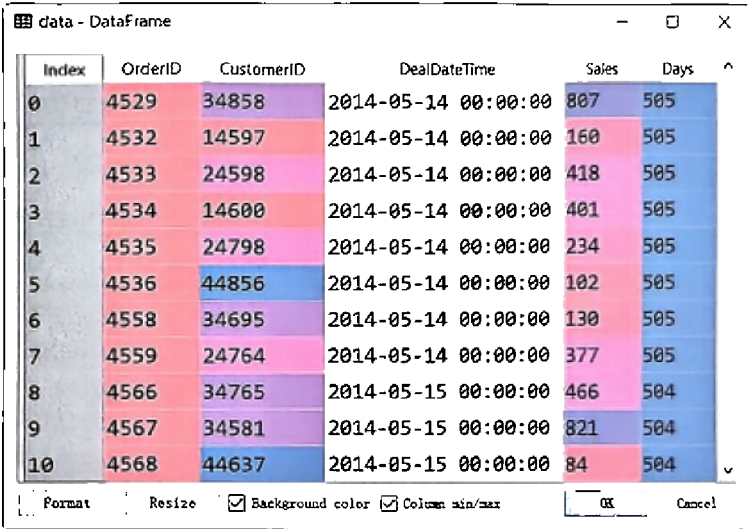
图 5-24 RFM 分析数据格式

根据交易日期，计算出交易日期距离指定日期的间隔天数，代码如下所示：

代码输入

```
# 将交易日期处理为日期数据类型
data['DealDateTime'] = pandas.to_datetime(
    data.DealDateTime,
    format='%Y/%m/%d'
)
# 假设 2015-10-1 是计算当天，求交易日期至计算当天的距离天数
data['Days'] = pandas.to_datetime('2015-10-1') - data['DealDateTime']
# 从时间距离中获取天数
data['Days'] = data['Days'].dt.days
```

执行代码，即可得到交易日期距离指定日期的天数，如图 5-25 所示。



Index	OrderID	CustomerID	DealDateTime	Sales	Days
0	4529	34858	2014-05-14 00:00:00	807	505
1	4532	14597	2014-05-14 00:00:00	160	505
2	4533	24598	2014-05-14 00:00:00	418	505
3	4534	14600	2014-05-14 00:00:00	401	505
4	4535	24798	2014-05-14 00:00:00	234	505
5	4536	44856	2014-05-14 00:00:00	102	505
6	4558	34695	2014-05-14 00:00:00	130	505
7	4559	24764	2014-05-14 00:00:00	377	505
8	4566	34765	2014-05-15 00:00:00	466	504
9	4567	34581	2014-05-15 00:00:00	821	504
10	4568	44637	2014-05-15 00:00:00	84	504

图 5-25 天数计算结果

STEP 02 计算 R、F、M

数据准备好后，接下来就可以计算每个客户的最近消费距离 R、消费频率 F 及消费总额 M，计算方法如下：

- ★ 最近消费距离 R：使用 CustomerID 作为分组列，距离指定日期间隔天数 Days 作为统计列，统计函数使用最小值函数 min，即可得到每个客户的最近消费距离 R。
- ★ 消费频率 F：使用 CustomerID 作为分组列，OrderID 作为统计列，统计函数使用计数函数 count。
- ★ 消费总额 M：使用 CustomerID 作为分组列，订单金额 Sales 作为统计列，统计函数使用求和函数 sum。

代码如下所示：

>> 谁说菜鸟不会数据分析（Python 篇）

代码输入

统计每个客户距离指定日期有多久没有消费了，即找出最小的最近消费距离

```
R = data.groupby(  
    by=['CustomerID'],  
    as_index=False
```

```
)['Days'].agg('min')
```

统计每个客户交易的总次数，即对订单 ID 计数

```
F = data.groupby(  
    by=['CustomerID'],  
    as_index=False  
)['OrderID'].agg('count')
```

统计每个客户交易的总额，即对每次的交易金额求和

```
M = data.groupby(  
    by=['CustomerID'],  
    as_index=False  
)['Sales'].agg('sum')
```

执行代码，得到的结果如图 5-26 所示。

Index	CustomerID	Days
0	14568	14
1	14569	104
2	14570	34
3	14571	74
4	14572	98
5	14573	84
6	14574	9
7	14575	9
8	14576	59

Index	CustomerID	OrderID
0	14568	15
1	14569	12
2	14570	15
3	14571	15
4	14572	8
5	14573	10
6	14574	15
7	14575	17
8	14576	12

Index	CustomerID	Sales
0	14568	6255
1	14569	5420
2	14570	8261
3	14571	8124
4	14572	3334
5	14573	4358
6	14574	8506
7	14575	7432
8	14576	6484

图 5-26 R、F、M 统计量

接下来使用 `merge` 方法，将 R、F、M 三个数据框关联起来，因为它们拥有共同的列名 `CustomerID`，并且 `CustomerID` 就是连接条件，在这种情况下，`on` 参数可以省略不写，代码如下所示：

代码输入

将 R、F、M 三个数据框关联，`merge` 默认内连接，可省略

```
RFMDData = R.merge(F).merge(M)
```

修改列名

```
RFMDData.columns = ['CustomerID', 'R', 'F', 'M']
```

执行代码，得到的结果如图 5-27 所示。

Inile	CustomerID	R	F	M
0	14568	14	15	6255
1	14569	104	12	5420
2	14570	34	15	8261
3	14571	74	15	8124
4	14572	98	8	3334
5	14573	84	10	4358
6	14574	9	15	8506
7	14575	9	17	7432

图 5-27 整合后的 R、F、M 统计量

STEP 03 将 R、F、M 分组打分赋值

各个客户的 R、F、M 数据计算好后，接下来就可以对 R、F、M 这三个列进行分组打分赋值得到对应的 R 分值、F 分值、M 分值。分组标准可以按照平均值、业务经验等标准进行划分。如果没有特别的标准，通常采用平均值进行划分。本例将 R、F、M 三列分别按照各自的平均值划分为 2 个组，并赋值 1 分、2 分。

- ★ R 分值 (R_S)：定义为距离指定日期越近，R_S 越大， $R \geq$ 平均值，R_S 为 1， $R <$ 平均值，R_S 为 2。
- ★ F 分值 (F_S)：定义为交易频率越高，F_S 越大， $F \leq$ 平均值，F_S 为 1， $F >$ 平均值，F_S 为 2。
- ★ M 分值 (M_S)：定义为交易金额越高，M_S 越大， $M \leq$ 平均值，M_S 为 1， $M >$ 平均值，M_S 为 2。

在 Python 中，可以使用数据框的 loc 属性将符合条件的数据行进行打分赋值，代码如下所示：

代码输入

```
# 判断 R 列是否大于等于 R 列的平均值，使用 loc 将符合条件 R_S 列的值赋值为 1
RFMDData.loc[RFMDData['R'] >= RFMDData.R.mean(), 'R_S'] = 1
# 判断 R 列是否小于 R 列的平均值，使用 loc 将符合条件 R_S 列的值赋值为 2
RFMDData.loc[RFMDData['R'] < RFMDData.R.mean(), 'R_S'] = 2
# 同 R_S 赋值方法，对 F_S、M_S 进行赋值，但与 R 相反，F、M 均为越大越好
RFMDData.loc[RFMDData['F'] <= RFMDData.F.mean(), 'F_S'] = 1
RFMDData.loc[RFMDData['F'] > RFMDData.F.mean(), 'F_S'] = 2
RFMDData.loc[RFMDData['M'] <= RFMDData.M.mean(), 'M_S'] = 1
RFMDData.loc[RFMDData['M'] > RFMDData.M.mean(), 'M_S'] = 2
```

执行代码，R_S、F_S、M_S 的分组分值就计算出来了，如图 5-28 所示。

Index	CustomerID	R	F	M	R_S	F_S	M_S
0	14568	14	15	6255	2	2	1
1	14569	104	12	5420	1	1	1
2	14570	34	15	8261	2	2	2
3	14571	74	15	8124	1	2	2
4	14572	98	8	3334	1	1	1
5	14573	84	10	4358	1	1	1
6	14574	9	15	8506	2	2	2
7	14575	9	17	7432	2	2	2
8	14576	59	13	6481	1	1	1
9	14577	20	8	2737	2	1	1
10	14578	62	16	7137	1	2	2

图 5-28 R_S、F_S、M_S 分组分值

STEP 04 计算 RFM 综合分值

得到 R_S、F_S、M_S 的分组分值后，接下来就可以计算 RFM 综合分值。RFM 综合分值计算公式如下所示：

$$RFM = 100 \times R_S + 10 \times F_S + 1 \times M_S$$

为什么设置 R_S 的权重为 100，F_S 的权重为 10，M_S 的权重为 1 呢？这样设置相当分别为百位、十位、个位的组合，以确保 RFM 综合分值顺序与 RFM 客户细分模型的分类顺序一致。

RFM 综合分值计算的代码如下所示：

代码输入

```
# 计算 RFM 综合分值
```

```
RFMData['RFM'] = 100*RFMData.R_S+10*RFMData.F_S+1*RFMData.M_S
```

执行代码，得到的 RFM 综合分值如图 5-29 所示。

Index	CustomerID	R	F	M	R_S	F_S	M_S	RFM
0	14568	14	15	6255	2	2	1	221
1	14569	104	12	5420	1	1	1	111
2	14570	34	15	8261	2	2	2	222
3	14571	74	15	8124	1	2	2	122
4	14572	98	8	3334	1	1	1	111
5	14573	84	10	4358	1	1	1	111
6	14574	9	15	8506	2	2	2	222
7	14575	9	17	7432	2	2	2	222
8	14576	59	13	6481	1	1	1	111
9	14577	20	8	2737	2	1	1	211
10	14578	62	16	7137	1	2	2	122

图 5-29 RFM 综合分值

STEP 05 客户分类

接下来根据 RFM 客户细分模型，将客户细分为八种不同的类型。本例采用与 RFM 综合分值客户类型的对应关系表匹配合并的方式实现客户分类。

首先将各个 RFM 综合分值与客户类型的对应关系定义为一个数据框。然后再使用 merge 中的内连接 inner 方法，将 RFMData 数据框与刚定义的 RFM 综合分值客户类型的对应关系表，根据关联列名 RFM 匹配合并为一个数据框，这样就完成了客户分类的操作，代码如下所示：

代码输入

```
# 定义 RFM 综合分值与客户类型的对应关系表
```

```
CustomerType = pandas.DataFrame(  
    data={  
        'RFM': [111,112,121,122,211,212,221,222]  
        'Type': ['潜在客户', '重点挽留客户', '一般保持客户', '重点保持客户',  
        '一般发展客户', '重点发展客户', '一般价值客户', '高价值客户']  
    }  
)
```

```
# 将 RFMData 与 RFM 综合分值客户类型的对应关系表合并为一个数据框
```

```
# merge 默认内连接，可省略，两表 on 条件的关联列名均为 RFM，同样可省略
```

```
RFMData = RFMData.merge(CustomerType)
```

执行代码，得到的数据如图 5-30 所示，可以看到，最后一列数据，就是对每个客户细分的客户类型。

Index	CustomerID	R	F	M	R_S	F_S	M_S	RFM	Type
0	14568	14	15	6255	2	2	1	221	一般价值客户
1	14583	24	15	6054	2	2	1	221	一般价值客户
2	14622	25	16	5530	2	2	1	221	一般价值客户
3	14629	7	14	5968	2	2	1	221	一般价值客户
4	14630	18	15	5840	2	2	1	221	一般价值客户
5	14635	12	14	5941	2	2	1	221	一般价值客户
6	14651	28	16	6368	2	2	1	221	一般价值客户
7	14668	37	14	6013	2	2	1	221	一般价值客户
8	14687	15	14	4908	2	2	1	221	一般价值客户
9	14725	16	14	5704	2	2	1	221	一般价值客户
10	14727	8	14	5146	2	2	1	221	一般价值客户

图 5-30 RFM 分析的结果

最后，我们来看看，每个类别的客户数是多少，代码如下所示：

代码输入	结果输出																											
<pre># 按 RFM、Type 进行分组统计客户数 rfmData.groupby(by=['RFM', 'Type'])['CustomerID'].agg('count')</pre>	<table><tr><th>RFM</th><th>Type</th><th></th></tr><tr><td>111</td><td>潜在客户</td><td>261</td></tr><tr><td>112</td><td>重点挽留客户</td><td>58</td></tr><tr><td>121</td><td>一般保持客户</td><td>34</td></tr><tr><td>122</td><td>重点保持客户</td><td>138</td></tr><tr><td>211</td><td>一般发展客户</td><td>257</td></tr><tr><td>212</td><td>重点发展客户</td><td>70</td></tr><tr><td>221</td><td>一般价值客户</td><td>67</td></tr><tr><td>222</td><td>高价值客户</td><td>315</td></tr></table> <p>Name: CustomerID, dtype: int64</p>	RFM	Type		111	潜在客户	261	112	重点挽留客户	58	121	一般保持客户	34	122	重点保持客户	138	211	一般发展客户	257	212	重点发展客户	70	221	一般价值客户	67	222	高价值客户	315
RFM	Type																											
111	潜在客户	261																										
112	重点挽留客户	58																										
121	一般保持客户	34																										
122	重点保持客户	138																										
211	一般发展客户	257																										
212	重点发展客户	70																										
221	一般价值客户	67																										
222	高价值客户	315																										

执行代码，就可以得到各个客户类型的客户数了。后续就可以对不同的客户群体，有针对性地采取相应运营策略进行推广、管理，进而提升客户价值和营收水平。

5.8 矩阵分析

矩阵分析，是指根据事物的两个重要属性（指标）作为分析的依据进行关联分析，找出解决问题的一种分析方法，也称为矩阵关联分析，简称矩阵分析法。

以属性 A 为横轴，属性 B 为纵轴，组成一个坐标系，在两坐标轴上分别按某一标准（可取平均值、经验值、行业水平等）进行象限划分，构成四个象限，将要分析的每个对象对应投射到这四个象限内，进行交叉分类分析，直观地将两个属性的关联性表现出来，进而分析每一个对象在这两个属性上的表现，如图 5-31 所示，因此它也称为象限图分析法。

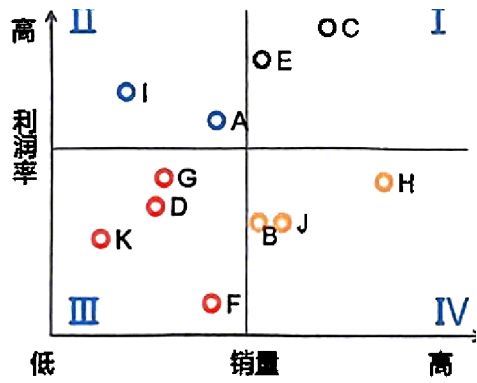


图 5-31 矩阵分析

矩阵关联分析法在解决问题和资源分配时，可为决策者提供重要参考依据。先解决主要矛盾，再解决次要矛盾，有利于提高工作效率，并将资源分配到最能产生绩效的部门、工作中，有利于决策者进行资源优化配置。

下面通过一个案例学习在 Python 中如何进行矩阵分析，将案例数据导入 data 变量，