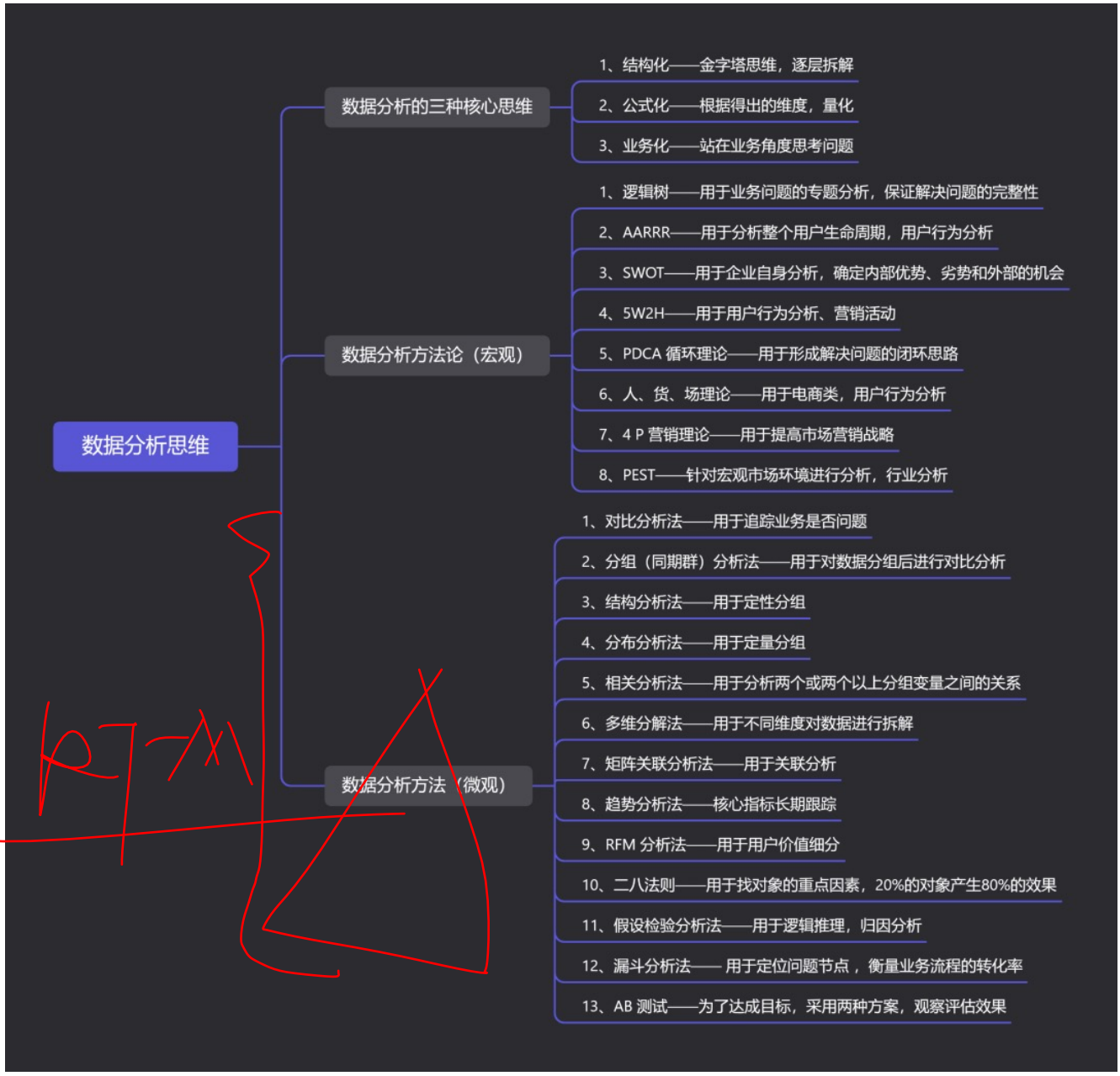
做数据分析的小伙伴可能多多少少都知道一些分析方法，但是谈到分析思维却没有底气或者遇到业务问题，不知道如何下手。

如果你有上述困惑，那么本篇文章可以作为参考。

下图是整理的分析方法论及方法。如果能够灵活运用，将能够解决工作中 80% 以上问题。

注意的是，方法论是思维层面，方法是执行层面。那么，重点是我们如何将其应用到实际业务中。本文将以 RFM 模型 为例，运用到实际案例中。（本文以 Python 实现，Excel 也可以）



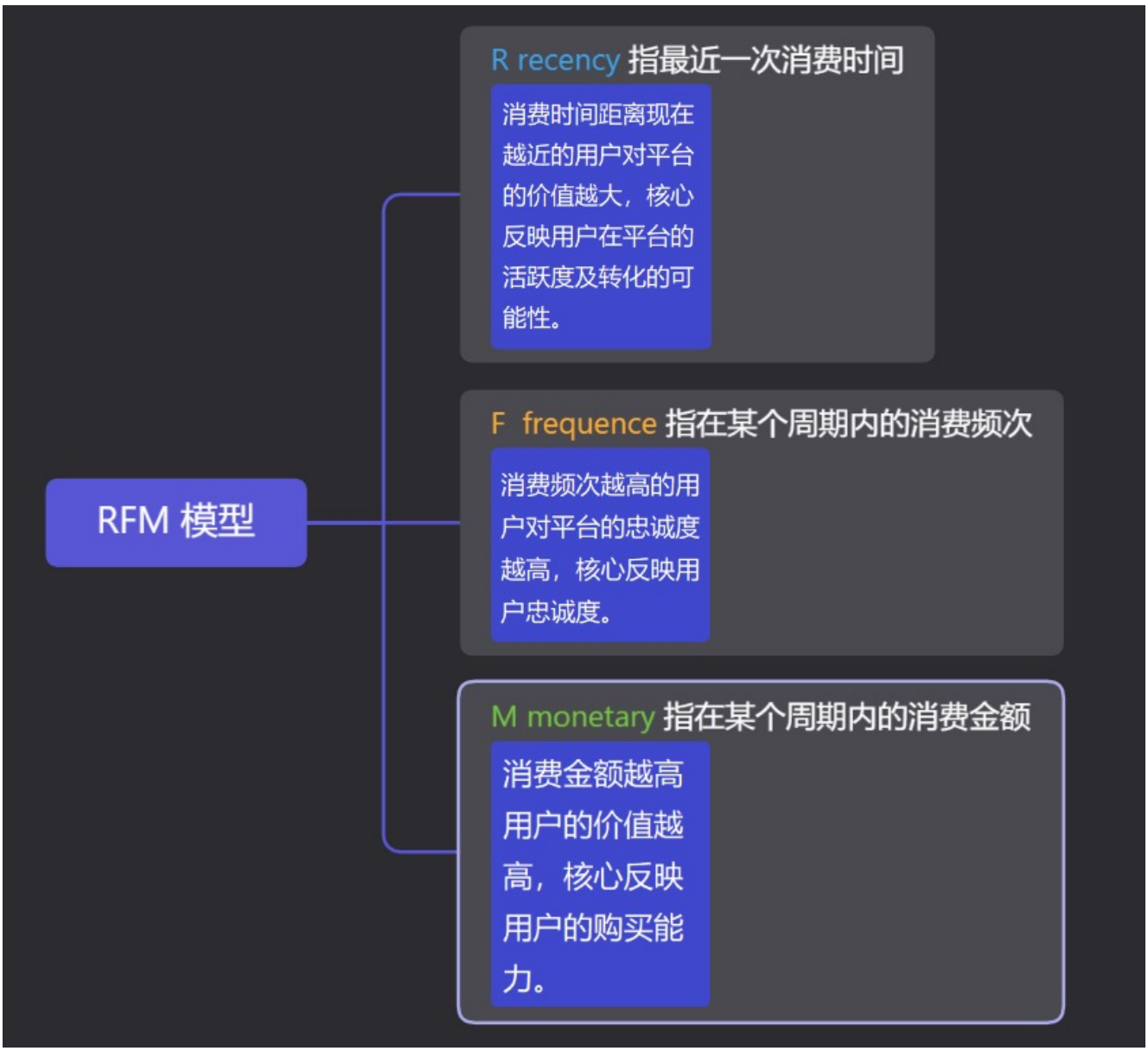
项目背景

某生鲜外卖APP于2018年1月1日成立，主营新鲜蔬菜瓜果，海鲜肉禽。APP上线后，市场推广期为一年。通过分析发现原来几个重要的客户被竞争对手挖走了，而这几个用户对平台贡献了80%的销售额。之前对所有用户采用一样的运营策略，为了解决这个问题，需要对用户进行**分类**，了解当前用户分层情况，进行**精细化运营**。

一、整体分析流程

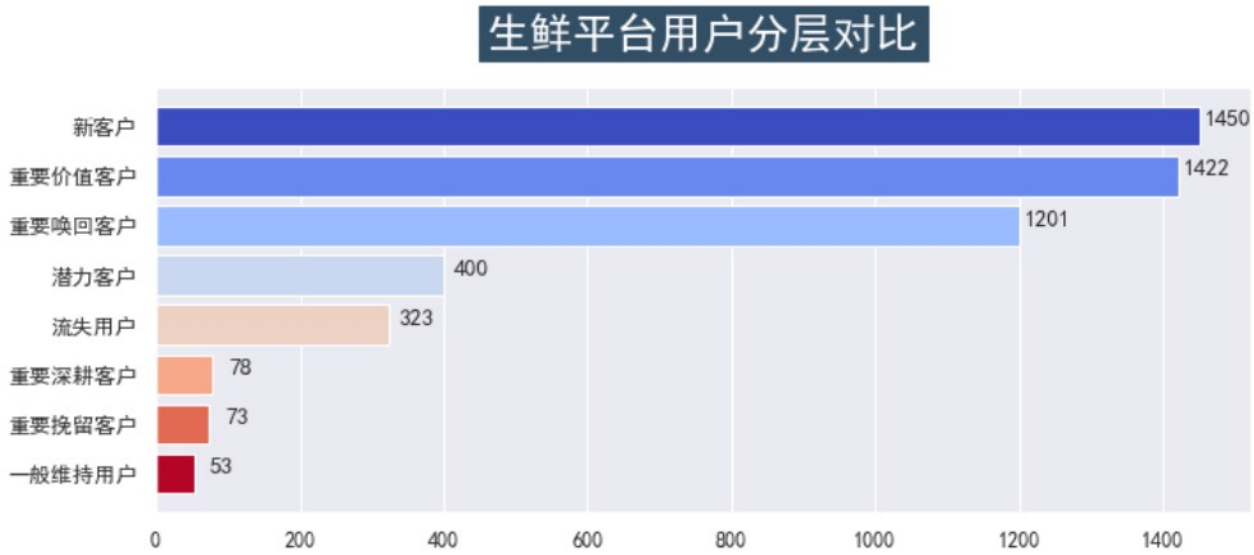
1. 分析目的：用户分类
2. 数据获取：Excel 数据
3. 清洗加工：Excel、Python
4. 建立模型：RFM
5. 数据可视化
6. 结论与建议

二、RFM 模型的理解



客户类型	最近交易距离当前天数(黏性)	累计单数(忠诚度)	累计交易金额(收入)	对应货运圈的成员场景
重要价值客户	↑	↑	↑	RFM都很大，优质客户，需要保持
重要唤回客户	↓	↑	↑	交易金额和交易次数大，但最近无交易。需要唤回
重要深耕客户	↑	↓	↑	交易金额大贡献度高，且最近有交易。需要重点识别
重要挽留客户	↓	↓	↑	交易金额大，潜在的有价值客户。需要挽留
潜力客户	↑	↑	↓	交易次数大，且最近有交易。需要挖掘
新客户	↑	↓	↓	最近有交易，接触的新客户，有推广价值
一般维持客户	↓	↑	↓	交易次数多，但是贡献不大，一般维持
流失客户	↓	↓	↓	FM值均低过平均值，最近也没再发货相当于流失

最终将 RFM 模型处理后的结果，作为用户标签，帮助运营更精准地制定活动规则以提升用户使用黏性，强化用户感知。最终实现的效果图如下：



三、利用 Python 实现 RFM 用户分层

3.1 获取数据

```python
import pandas as pd
data = pd.read_excel('./用户信息.xlsx')
data.head()
```

	用户ID	注册时间	性别	年龄	会员开通时间	会员类型	城市	区域	最后一次登陆	最后一次成交	非会员累计购买次数	非会员累计消费	会员累计购买次数	会员累计消费
0	10000001	2018-01-01	女	43	2018-07-05	VIP	北京市	北京市	2019-06-19 02:24:00	2018-06-16 02:24:00	12	1268.28	23	1835.17
1	10000002	2018-01-01	男	56	NaT	NaN	北京市	北京市	2019-06-30 00:00:00	2018-06-29 00:00:00	36	3142.08	0	0.00
2	10000003	2018-01-01	男	50	2018-07-06	VIP	北京市	北京市	2019-06-30 00:00:00	2018-07-27 00:00:00	12	1369.56	19	1526.65
3	10000004	2018-01-01	女	23	2018-07-06	VIP	北京市	丰台区	2019-06-08 04:48:00	2018-06-08 04:48:00	12	1314.96	45	2484.00
4	10000005	2018-01-01	女	33	2018-07-01	VIP	北京市		2019-05-11 22:48:00	2019-05-07 22:48:00	12	1006.80	23	2046.77

PYTHON

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 14 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   用户ID        5000 non-null   int64
 1   注册时间        5000 non-null   datetime64[ns]
 2   性别          5000 non-null   object
 3   年龄          5000 non-null   int64
 4   会员开通时间      2795 non-null   datetime64[ns]
 5   会员类型        2795 non-null   object
 6   城市          5000 non-null   object
 7   区域          4928 non-null   object
 8   最后一次登陆      5000 non-null   datetime64[ns]
 9   最后一次成交      5000 non-null   datetime64[ns]
 10  非会员累计购买次数  5000 non-null   int64
 11  非会员累计消费     5000 non-null   float64
 12  会员累计购买次数    5000 non-null   int64
 13  会员累计消费      5000 non-null   float64
dtypes: datetime64[ns](4), float64(2), int64(4), object(4)
memory usage: 547.0+ KB
```

说明：当前数据集是5000条用户数据，存在缺失值对本次分析不会造成影响。数据清洗，通常包括处理缺失值、重复值、转换数据类型三种。所以仅考虑数据类型即可。这里有个前提条件，R、F、M 应该有一个参照时间，如果活动持续到现在，可以截止到现在。但是我们的数据是历史数据，所以需要查找活动结束时间。

PYTHON

```
data.sort_values(by='最后一次成交', ascending=False)
```

	用户ID	注册时间	性别	年龄	会员开通时间	会员类型	城市	区域	最后一次登陆	最后一次成交	非会员累计购买次数	非会员累计消费	会员累计购买次数	会员累计消费
4999	10005000	2019-06-30	男	26	2019-06-30	VIP	北京市	朝阳区	2019-06-30 00:00:00	2019-06-30 00:00:00	0	0.00	1	52.00
4933	10004934	2019-06-18	女	35	NaT	NaN	北京市	北京市	2019-06-25 16:19:12	2019-06-30 00:00:00	0	0.00	0	0.00
4892	10004893	2019-06-08	男	21	2019-06-08	VIP	北京市	西城区	2019-06-30 00:00:00	2019-06-30 00:00:00	0	0.00	1	70.83
4896	10004897	2019-06-09	男	35	NaT	NaN	北京市	朝阳区	2019-06-27 21:36:00	2019-06-30 00:00:00	0	0.00	0	0.00

3.2 数据处理

PYTHON

```
# 活动结束时间 2019-06-30

data['最后一次成交']=data['最后一次成交'].astype('str')

stop_date = pd.to_datetime('2019-06-30')

datas = data.drop(columns=['注册时间','会员开通时间','会员类型','城市','区域','最后一次登陆'])

datas['最后一次成交时间'] = datas['最后一次成交'].apply(lambda x:x.split()[0])

datas['最后一次成交时间'] = pd.to_datetime(datas['最后一次成交时间'])

datas['R1'] = datas['最后一次成交时间'].apply(lambda x:stop_date-x)

datas['F1'] = datas['非会员累计购买次数'] + datas['会员累计购买次数']

datas['M1'] = datas['非会员累计消费'] + datas['会员累计消费']

datas['R1'] = datas['R1'].astype(str)

datas['R1']= datas['R1'].apply(lambda x:x.split()[0])
datas
```

	用户ID	性别	年龄	最后一次成交	非会员累计购买次数	非会员累计消费	会员累计购买次数	会员累计消费	最后一次成交时间	R1	F1	M1
0	10000001	女	43	2018-06-16 02:24:00	12	1268.28	23	1835.17	2018-06-16	379	35	3103.45
1	10000002	男	56	2018-06-29 00:00:00	36	3142.08	0	0.00	2018-06-29	366	36	3142.08
2	10000003	男	50	2018-07-27 00:00:00	12	1369.56	19	1526.65	2018-07-27	338	31	2896.21
3	10000004	女	23	2018-06-08 04:48:00	12	1314.96	45	2484.00	2018-06-08	387	57	3798.96
4	10000005	女	33	2019-05-07 22:48:00	12	1006.80	23	2046.77	2019-05-07	54	35	3053.57
...	...	...	...	...	...	...	...	...	...	...	...	...
4995	10004996	女	48	2019-06-30 00:00:00	0	0.00	0	0.00	2019-06-30	0	0	0.00
4996	10004997	女	26	2019-06-30 00:00:00	0	0.00	0	0.00	2019-06-30	0	0	0.00
4997	10004998	男	18	2019-06-30 00:00:00	0	0.00	1	98.00	2019-06-30	0	1	98.00
4998	10004999	男	30	2019-06-30	0	0.00	0	0.00	2019-06-30	0	0	0.00

说明：以上操作目的是将R指标由时间类型转换成可计算格式，为接下来建立模型，计算时间间隔做准备。

3.3 建立模型

建立模型，需要分别对F、R、M 分别计算各自的平均值。但是要注意三个指标数据存在极大值、极小值的情况，这对结果会产生一定的误差，所以解决方案是将其标准化，设置分段区间，5分制，5分为最高。（数值区间可根据具体业务灵活调整或者用四分位数）

R数据	R分	F数据	F分	M数据	M分
0	5	0	1	0	1
80	4	14	2	1500	2
160	3	28	3	3000	3
240	2	42	4	4500	4
320	1	56	5	6000	5

PYTHON

```
def R_score(n):

    n = int(n)

    if 0<n<=80:
        r = 5
    elif 80<n<=160:
        r = 4
    elif 160<n<=240:
        r = 3
    elif 240<n<=320:
        r = 2
    else:
        r = 1

    return r
```

```
def F_score(n):

    n = int(n)

    if 0<n<=14:
        r = 1
    elif 14<n<=28:
        r = 2
    elif 28<n<=42:
        r = 3
    elif 42<n<=56:
        r = 4
    else: r = 5

    return r

def M_score(n):

    n = int(n)

    if 0<n<=1500:
        r = 1
    elif 1500<n<=3000:
        r = 2
    elif 3000<n<=4500:
        r = 3
    elif 4500<n<=6000:
        r = 4
    else:
        r = 5

    return r

datas['M1_score'] =datas['M1'].apply(M_score)

datas['F1_score'] =datas['F1'].apply(F_score)

datas['R1_score'] =datas['R1'].apply(R_score)

datas.head()
```

	用户ID	性别	年龄	最后一次成交	非会员累计购买次数	非会员累计消费	会员累计购买次数	会员累计消费	最后一次成交时间	R1	F1	M1	R1_score	F1_score	M1_score
0	10000001	女	43	2018-06-16 02:24:00	12	1268.28	23	1835.17	2018-06-16	379	35	3103.45	1	3	3
1	10000002	男	56	2018-06-29 00:00:00	36	3142.08	0	0.00	2018-06-29	366	36	3142.08	1	3	3
2	10000003	男	50	2018-07-27 00:00:00	12	1369.56	19	1526.65	2018-07-27	338	31	2896.21	1	3	2
3	10000004	女	23	2018-06-08 04:48:00	12	1314.96	45	2484.00	2018-06-08	387	57	3798.96	1	5	3
4	10000005	女	33	2019-05-07 22:48:00	12	1006.80	23	2046.77	2019-05-07	54	35	3053.57	5	3	3

说明：这里对R、F、M再求平均值，以平均值为标准，如果单个指标大于平均值，显示1，否则显示0。最终RFM的结果由0和1拼接组成，即可得出用户最终类型。

PYTHON

```
R_mean = datas['R1_score'].mean()

F_mean = datas['F1_score'].mean()

M_mean = datas['M1_score'].mean()

datas['R'] = datas['R1_score'].apply(lambda x: 1 if x> R_mean else 0)

datas['F'] = datas['F1_score'].apply(lambda x: 1 if x> F_mean else 0)

datas['M'] = datas['M1_score'].apply(lambda x: 1 if x> M_mean else 0)

datas
```

	用户ID	性别	年龄	最后一次成交	非会员累计购买次数	非会员累计消费	会员累计购买次数	会员累计消费	最后一次成交时间	R1	F1	M1	F1_score	M1_score	R1_score	R	F	M
0	10000001	女	43	2018-06-16 02:24:00	12	1268.28	23	1835.17	2018-06-16	379	35	3103.45	3	3	1	0	1	1
1	10000002	男	56	2018-06-29 00:00:00	36	3142.08	0	0.00	2018-06-29	366	36	3142.08	3	3	1	0	1	1
2	10000003	男	50	2018-07-27 00:00:00	12	1369.56	19	1526.65	2018-07-27	338	31	2896.21	3	2	1	0	1	1
3	10000004	女	23	2018-06-08 04:48:00	12	1314.96	45	2484.00	2018-06-08	387	57	3798.96	5	3	1	0	1	1
4	10000005	女	33	2019-05-07 22:48:00	12	1006.80	23	2046.77	2019-05-07	54	35	3053.57	3	3	5	1	1	1

PYTHON

```
datas['RFM'] =
datas['R'].apply(str)+datas['F'].apply(str)+datas['M'].apply(str)

datas

def user_tag(rfm):
```

```
    if rfm=='000':
        res = '流失用户'
    elif rfm=='010':
        res = '一般维持用户'
    elif rfm=='100':
        res = '新客户'
    elif rfm=='110':
        res = '潜力客户'
    elif rfm=='001':
        res = '重要挽留客户'
    elif rfm=='101':
        res = '重要深耕客户'
    elif rfm=='011':
        res = '重要唤回客户'
    else:
        res = '重要价值客户'

    return res

datas['user_tag']=datas['RFM'].apply(user_tag)

datas
```

	用户ID	性别	年龄	最后一次成交	非会员累计购买次数	非会员累计消费	会员累计购买次数	会员累计消费	最后一次成交时间	R1	F1	M1	R1_score	F1_score	M1_score	R	F	M	RFM	user_tag
0	10000001	女	43	2018-06-16 02:24:00	12	1268.28	23	1835.17	2018-06-16	379	35	3103.45	1	3	3	0	1	1	011	重要唤回客户
1	10000002	男	56	2018-06-29 00:00:00	36	3142.08	0	0.00	2018-06-29	366	36	3142.08	1	3	3	0	1	1	011	重要唤回客户
2	10000003	男	50	2018-07-27 00:00:00	12	1389.56	19	1526.65	2018-07-27	338	31	2896.21	1	3	2	0	1	1	011	重要唤回客户
3	10000004	女	23	2018-06-08 04:48:00	12	1314.96	45	2484.00	2018-06-08	387	57	3798.96	1	5	3	0	1	1	011	重要唤回客户

3.4 数据可视化

PYTHON

```
import matplotlib.pyplot as plt
import seaborn as sns
import matplotlib as mpl

sns.set(font='SimHei',style='darkgrid')

user_tag = datas.groupby(datas['user_tag']).size()
```

```
plt.figure(figsize = (10,4),dpi=80)

user_tag.sort_values(ascending=True,inplace=True)

plt.title(label='生鲜平台用户分层对比',
          fontsize=22, color='white',
          backgroundcolor='#334f65', pad=20)

s = plt.barh(user_tag.index,user_tag.values , height=0.8,
color=plt.cm.coolwarm_r(np.linspace(0,1,len(user_tag))))
for rect in s:
    width = rect.get_width()
    plt.text(width+40,rect.get_y() + rect.get_height()/2,
str(width),ha= 'center')

plt.grid(axis='y')
plt.show()
```

PYTHON

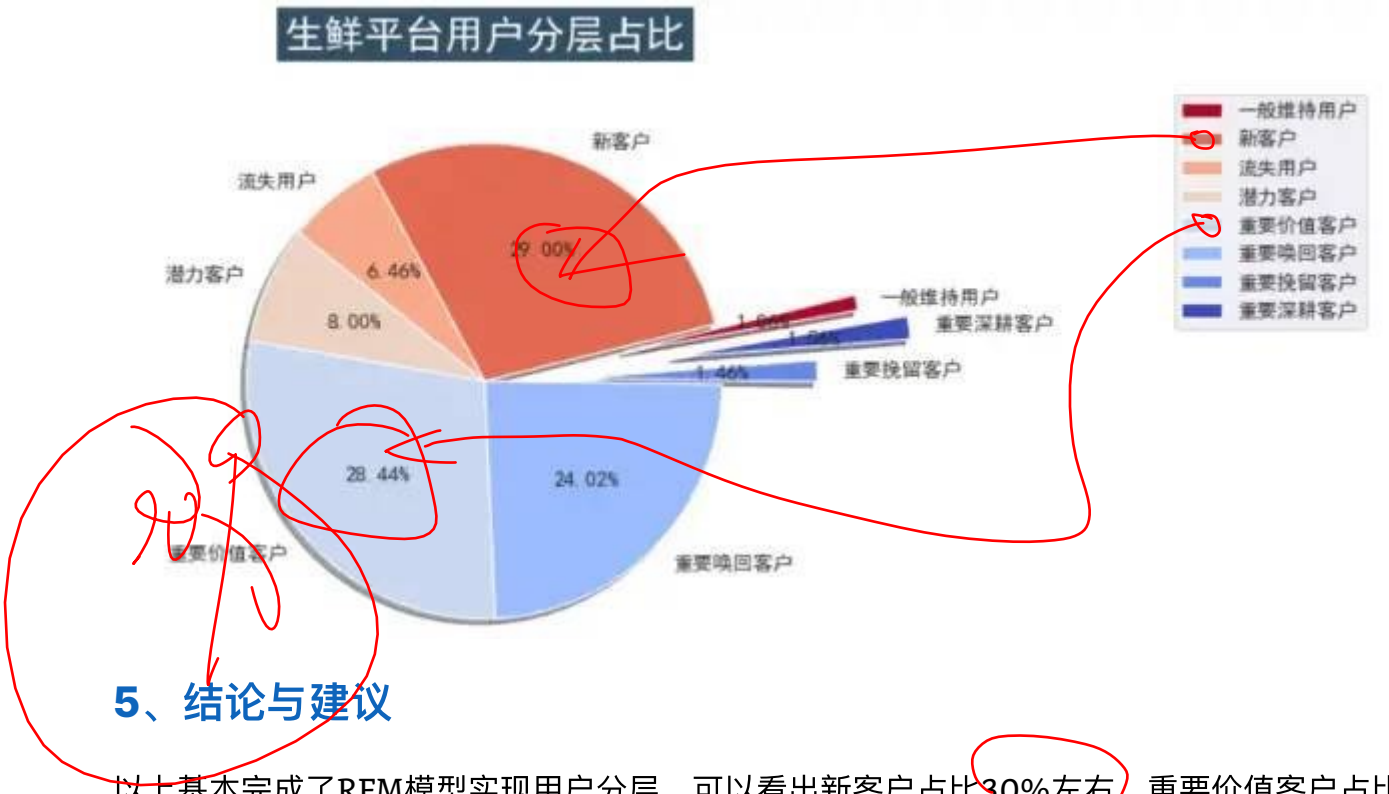
```
groups_b = datas.groupby(by='user_tag').size()

plt.figure(figsize = (10,6),dpi=80)
plt.title(label='生鲜平台用户分层占比',
          fontsize=22, color='white',
          backgroundcolor='#334f65', pad=20)

explodes = [0.6, 0, 0, 0, 0,0,0.4,0.8]

patches, l_text, p_text = plt.pie(groups_b.values,labels =
```

```
groups_b.index,
shadow=True,colors=plt.cm.coolwarm_r(np.linspace(0,1,len(groups_b))),
autopct='%.2f%%', explode=explodes,startangle=370)
plt.legend(ins,bbox_to_anchor=(2, 1.0))
plt.show()
```



5、结论与建议

以上基本完成了RFM模型实现用户分层，可以看出新客户占比30%左右，重要价值客户占比30%左右。两者是平台的最主要用户类型。

接下来就需要结合具体业务来制定运营策略。最后分享的是，现在我们看到最多的招聘需求是具备分析思维。那什么是分析思维。

我的理解是，首先要理解业务，其次要掌握分析方法，要明确分析方法存在的意义是帮助我们将零散业务问题归类，归类的过程形成分析思路，有了分析思路，那你就具备了分析思维。