

数据分析项目：用户消费数据分析

项目背景： 针对某网站两个月用户消费记录进行分析，了解消费的整体情况，发现存在的问题，根据复购率，回购率，高额消费用户等指标以及消费模型等，对客户分类，进行针对性的客户管理与维护

数据来源： 某网站用户消费记录，通过以下字段利用python进行数据分析

分析思路：

1. 数据准备阶段（数据预处理）
2. 数据概览
3. 从时间维度分析
4. 用户个体消费分析
5. 用户消费行为分析
6. 复购率，回购率分析

1. 数据准备

1.1 导入常用的python库

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
plt.rcParams['font.sans-serif']=['SimHei']
plt.rcParams['axes.unicode_minus'] = False
plt.style.use('ggplot')
import pymysql
from sqlalchemy import create_engine
import warnings
warnings.filterwarnings('ignore')
```

1.2 从MySQL读取数据，观察数据

```
sql = "SELECT userId as user_id, paidTime as order_dt, count(price) as order_products, sum(price) as order_amount FROM data.orderinfo where isPaid='已支付' group by userId, paidTime"
engine = create_engine('mysql+pymysql://root:8888@localhost:3306/data?charset=utf8')
data = pd.read_sql(sql,engine)
data.head()
```

	user_id	order_dt	order_products	order_amount
0	11211	2016-03-01 00:04:00	9	6272.000000
1	11211	2016-03-01 00:03:00	1	487.000000
2	74553	2016-03-01 00:02:00	1	737.000000
3	37799	2016-03-01 00:01:00	1	647.000000
4	13341	2016-03-01 00:06:00	1	890.700012

字段释义：

- user_id: 用户id
- order_dt: 购买日期
- order_products: 购买产品数
- order_amount: 购买金额

data.info()

```
RangeIndex: 211677 entries, 0 to 211676
Data columns (total 4 columns):
user_id      211677 non-null int64
order_dt     211677 non-null datetime64[ns]
order_products 211677 non-null int64
order_amount 211677 non-null float64
dtypes: datetime64[ns](1), float64(1), int64(2)
```

1.3 调整字段数据类型并提取月份，星期，时刻，年中的第几周等字段

```
data.user_id = data.user_id.astype('object')
data['order_dt'] = pd.to_datetime(data.order_dt, format='%Y-%m-%d')
data['month'] = data.order_dt.values.astype('datetime64[M]')
data['weekday'] = data.order_dt.dt.weekday+1
data['time'] = data.order_dt.dt.hour
data['week_year'] = data.order_dt.dt.weekofyear
data.weekday = data.weekday.astype('object')
data.time = data.time.astype('object')
data.week_year = data.week_year.astype('object')
data.info()
```

```
RangeIndex: 211677 entries, 0 to 211676
Data columns (total 8 columns):
user_id      211677 non-null object
order_dt     211677 non-null datetime64[ns]
order_products 211677 non-null int64
order_amount 211677 non-null float64
month        211677 non-null datetime64[ns]
weekday      211677 non-null object
time         211677 non-null object
week_year    211677 non-null object
dtypes: datetime64[ns](2), float64(1), int64(1), object(4)
```

1.4 去除日期时分秒，购买金额保留2位小数

```
data['order_dt'] = pd.to_datetime(data['order_dt']).dt.normalize()
data['order_amount'] = round(data.order_amount, 2)
data.head()
```

	user_id	order_dt	order_products	order_amount	month	weekday	time	week_year
0	11211	2016-03-01	9	6272.0	2016-03-01	2	0	9
1	11211	2016-03-01	1	487.0	2016-03-01	2	0	9
2	74553	2016-03-01	1	737.0	2016-03-01	2	0	9
3	37799	2016-03-01	1	647.0	2016-03-01	2	0	9
4	13341	2016-03-01	1	890.7	2016-03-01	2	0	9

1.5 缺失值检测

```
data.isnull().any()
```

```
user_id      False
order_dt     False
order_products False
order_amount False
month        False
weekday      False
time         False
week_year    False
```

1.6 分析用户个体消费时，发现异常值id，应予以去掉

```
print('id为11211用户消费记录数: ', data[data.user_id==11211].shape[0])
print('id为11211用户消费产品数: 318674')
print('id为11211用户消费总金额: 约2.3亿')
data.groupby('user_id').sum().sort_values('order_amount', ascending=False).head()
```

```
id为11211用户消费记录数 : 68902
id为11211用户消费产品数 : 318674
id为11211用户消费总金额 : 约2.3亿
```

	order_products	order_amount
user_id		
11211	318674	231167990.0
57282	713	457232.0
62590	350	223636.0
14427	240	221821.0
53616	337	203981.0

```
# 去掉异常值id
data = data[data.user_id!=11211]
```

2. 数据概览

```
print('消费总金额数:',data.order_amount.sum())
print('统计时间范围: ', data.order_dt.min(), '---', data.order_dt.max())
print('购买产品总数:', data.order_products.sum())
print('总的用户数量: ', data.user_id.nunique())
data.describe()
```

消费总金额数：87335092.10
统计时间范围：2016-03-01 --- 2016-05-02
购买产品总数：143131
总的用户数量：85648

	order_products	order_amount
count	142775.000000	142775.000000
mean	1.002493	611.697371
std	0.050012	287.967807
min	1.000000	-19.000000
25%	1.000000	422.000000
50%	1.000000	567.000000
75%	1.000000	746.000000
max	3.000000	5254.000000

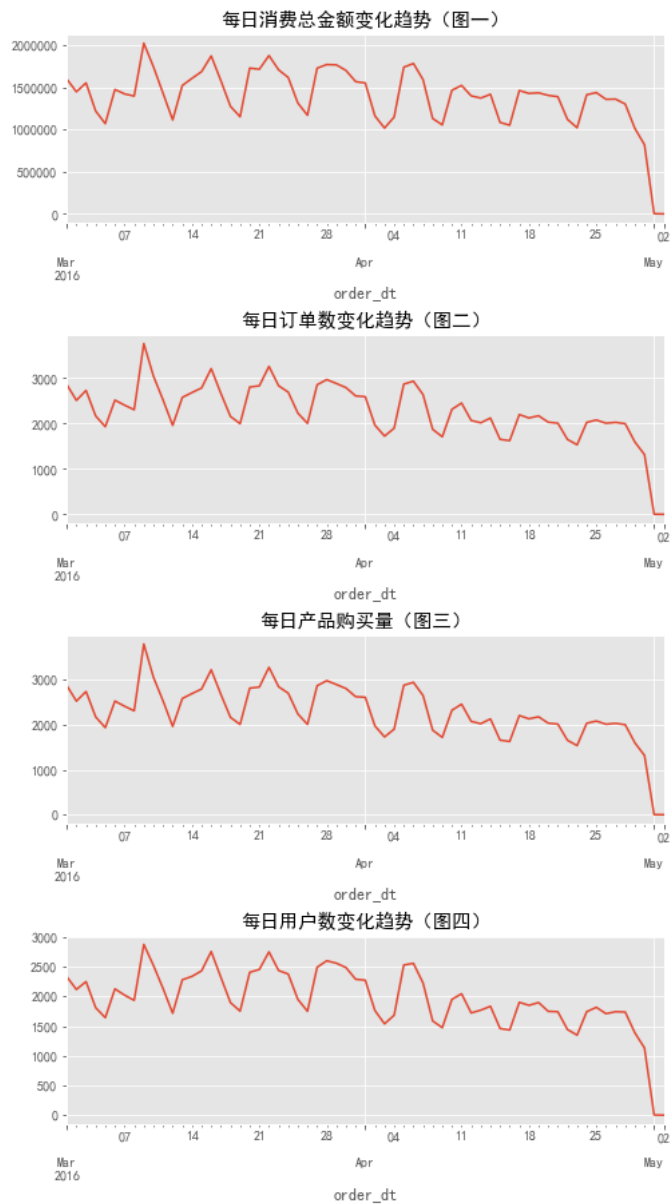
- 数据统计的时间范围从2016-03-01至2016-05-02，共2个月数据
- 用户总数为85648人，用户消费总金额约8733万，购买产品总数约14.3万个
- 用户平均订单购买1个商品，标准差为0.05，数据分布平稳，中位数为1个商品，说明绝大部分订单只购买1件商品，最大值为3
- 用户平均消费611元左右，中位数567元，最大金额达到5254元，有一定极值干扰，最小值为-19元，可能与活动返现有关

3.从时间消费维度分析

- 每日消费趋势
- 周一至周日订单量趋势
- 每天各时间段订单量趋势

3.1 每日消费趋势

```
plt.figure(figsize=(8,15))
plt.subplots_adjust(wspace =0, hspace =0.6)
#每日消费总金额分析
plt.subplot(411)
data.groupby('order_dt').sum().order_amount.plot()
plt.title('每日消费总金额变化趋势（图一）')
#每日订单数
plt.subplot(412)
data.groupby('order_dt').count().order_amount.plot()
plt.title('每日订单数变化趋势（图二）')
#每日产品购买量
plt.subplot(413)
data.groupby('order_dt').sum().order_products.plot()
plt.title('每日产品购买量（图三）')
#每日用户数
plt.subplot(414)
data.groupby('order_dt').nunique().user_id.plot()
plt.title('每日用户数变化趋势（图四）')
```



总体来看：每日消费趋势整体呈周期性分布，每7天为一周期，在周期内上下波动，四月中下旬开始出现用户流失，消费额下降的情况

图一：每日消费总额初期在100万到200万范围内波动，后3周开始下降，在100万到150万范围波动

图二：每日订单数初期波动范围在2000到4000笔，后3周开始下降，在2000笔上下波动

图三：每日产品购买量初期波动范围在约2000到4000个左右，后3周开始下降，在2000个上下波动

图四：每日消费人数初期波动范围在约1500到3000人左右，后3周下降到1500人上下波动

数据透视查看，每月的购买总额、购买总产品数、总用户数

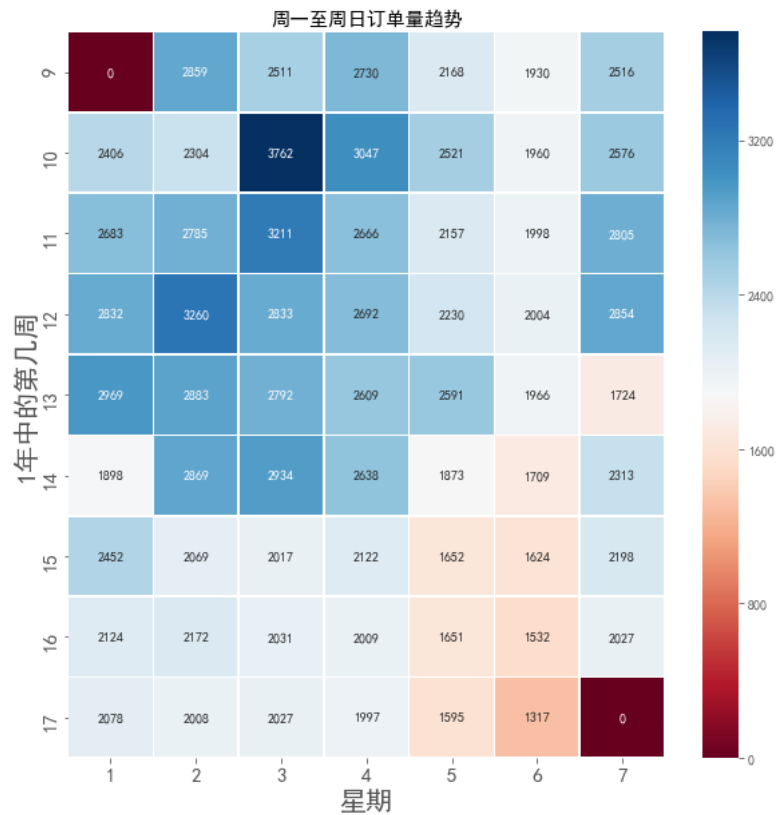
```
data[data.month!='2016-05-01'].pivot_table(index='month',
values=['order_amount', 'order_products', 'user_id'],
aggfunc={'order_amount':'sum', 'order_products':'sum', 'user_id':'nunique'})
```

	order_amount	order_products	user_id
month			
2016-03-01	47859907.69	81775	54798
2016-04-01	39471991.65	61351	43966

- 三、四月份销售总额分别为4786万和3947万，呈下降趋势
- 三、四月份销售总产品数分别为81.7万和61.3万，呈下降趋势
- 三、四月份总用户数分别为5.48万，4.40万，呈下降趋势

3.2 周一至周日订单量趋势

```
# 用热力图显示周一至周日订单量趋势
# 根据日期提取一年当中的第几周
df = data[data.month!='2016-05-01'].pivot_table(index='week_year', columns='weekday', values='order_dt', aggfunc='count').fillna(0)
plt.figure(figsize=(10,10))
plt.title('周一至周日订单量趋势')
ax = sns.heatmap(df, linewidths=.5, cmap='RdBu', annot=True, fmt='lg')
ax.set_ylabel('1年中的第几周', fontsize=20)
ax.set_xlabel('星期', fontsize=20)
ax.tick_params(axis='y', labels=15)
ax.tick_params(axis='x', labels=15)
```

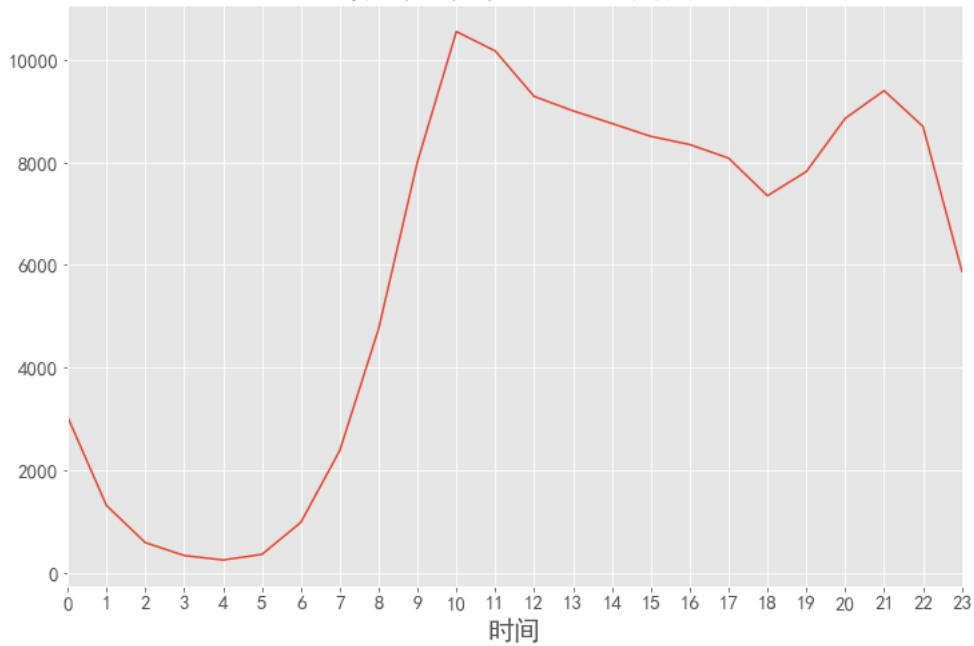


- 订单大部分集中在周日到周四，而周五和周六较少
- 后3周的订单量都普遍减少

3.3 每天各时间段订单量趋势

```
plt.figure(figsize=(12,8))
plt.title('每天各时间段订单量趋势', fontsize=25)
data.groupby('time').count().order_dt.plot()
plt.tick_params(axis='y', labels=15)
plt.tick_params(axis='x', labels=15)
plt.xlabel('时间', fontsize=20)
plt.xticks(np.arange(0,24,1))
plt.show()
```

每天各时间段订单量趋势



- 用户下单时间主要集中在9点到22点
- 从凌晨到6点订单量较少，4点左右为最低点，6点到10点订单数开始猛增，在10点出现一天的第一高峰；18点左右用餐时间订单量有所下降；之后从18点到21点出现第二个订单波峰

4 个体消费维度分析

- 用户消费金额，消费产品数描述统计
- 用户消费金额和消费产品数散点图
- 用户消费金额的分布图
- 用户消费产品数的分布图
- 用户累计消费金额占比

4.1 用户消费金额和消费产品数的描述统计

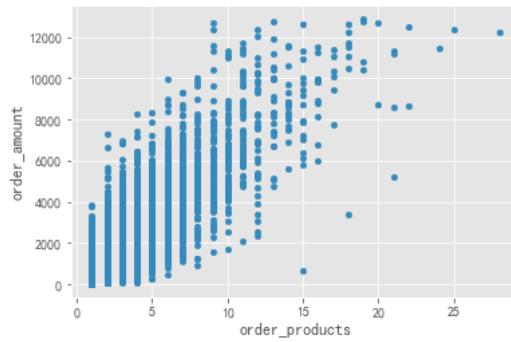
```
data.groupby('user_id').sum().describe()
```

	order_products	order_amount
count	85648.000000	85648.000000
mean	1.671154	1019.697974
std	3.759634	2516.263988
min	1.000000	18.000000
25%	1.000000	502.000000
50%	1.000000	737.000000
75%	2.000000	1175.250000
max	713.000000	457232.000000

- 用户平均购买1.67个产品，中位数为1，最大值是713，存在极值干扰，有小部分用户购买了大量产品
- 用户平均消费1019元，中位数737，最大值45万，同样存在极值干扰

4.2 用户消费金额和消费产品数散点图

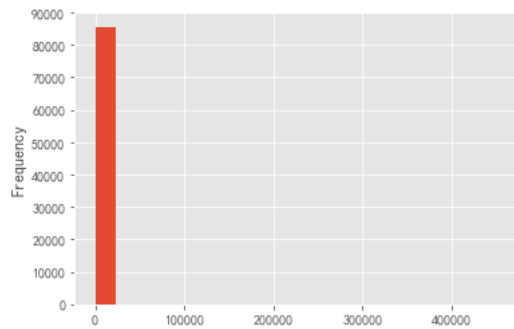
```
#用query结合切比雪夫定理，排除一些极值的干扰，
data.groupby('user_id').sum().query('order_amount<13000').\
    plot.scatter(x='order_products', y='order_amount')
plt.show()
```



- 从散点图看出，同样的消费产品数有不同的消费金额，说明产品种类比较丰富，或者是由于针对不同人群采取不同的价格策略

4.3 用户消费金额的分布图

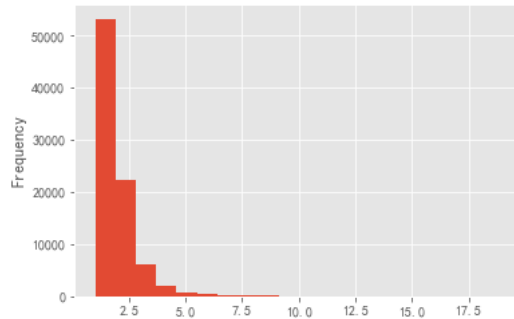
```
data.groupby('user_id').sum().order_amount.plot.hist(bins=20)
plt.show()
```



- 有直方图看出，大部分用户消费金额并不高，高消费用户在图中基本看不到，这也符合消费行为的行业规律

4.4 用户消费产品数的分布图

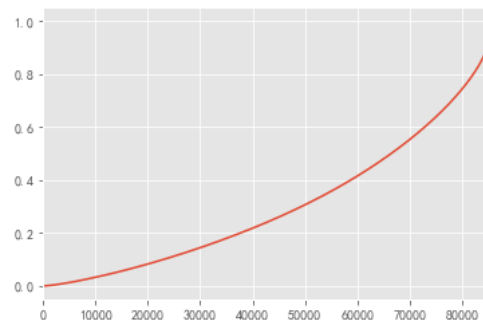
```
data.groupby('user_id').sum().query('order_products<20').order_products.plot.hist(bins=20)
plt.show()
```



- 使用切比雪夫定理过滤异常值，计算95%的数据分布情况（均值+5*标准差）

4.5 用户累计消费金额占比

```
user_cumsum = data.groupby('user_id').sum().sort_values('order_amount').\
    apply(lambda x : x.cumsum()/x.sum())
user_cumsum.reset_index(drop=True).order_amount.plot()
plt.show()
```



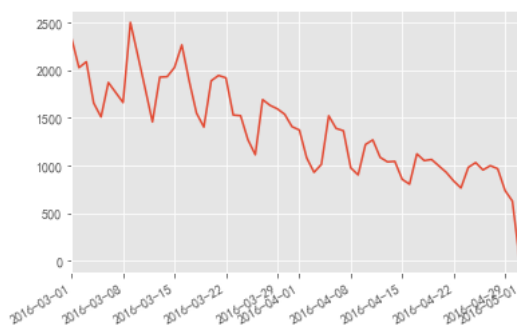
- 通过观察，50%的用户仅仅贡献了25%左右的消费额度，而排名前10000用户就贡献了40%左右的消费额

5. 用户生命周期维度分析

- 用户第一次消费（首购）
- 用户最后一次消费
- 新老客户消费比
 - 多少用户仅仅消费一次
 - 每月新客占比
- 用户分层
 - RFM模型
 - 新、老、活跃、回流/不活跃
- 用户购买周期（按订单）
 - 用户消费周期描述
 - 用户消费周期分布
- 用户生命周期（按第一次 & 最后一次消费）
 - 用户生命周期描述
 - 用户生命周期分布

5.1 第一次购买时间分布

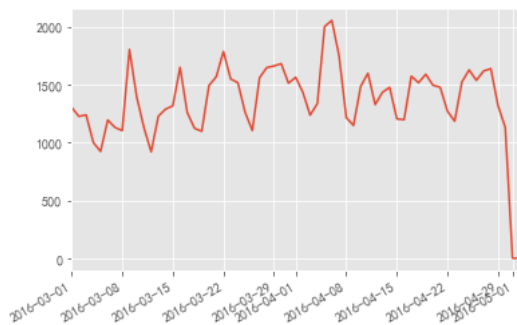
```
df1 = data.groupby('user_id').min().order_dt
df1.value_counts().plot()
plt.show()
```



- 用户第一次购买时间分布，以7天为单位，呈周期性上下波动分布，最高峰为2500，最低为700左右，整体表现出逐渐下降的趋势，说明新增用户逐渐减少

5.2 用户最后一次购买时间分布

```
data.groupby('user_id').max().order_dt.value_counts().plot()
plt.show()
```



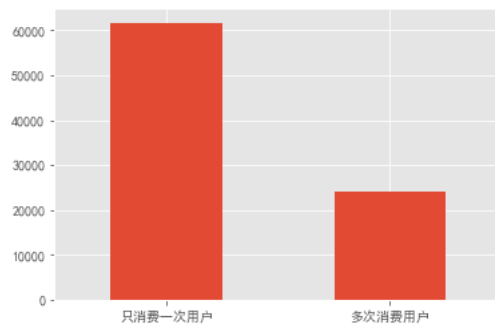
- 用户最后一次购买的分布也是呈7天周期性波动分布，最高峰为2100左右，最低为900左右，分布较平稳，说明用户一直在稳定流失
- 初期流失用户小于新增用户，四月中下旬开始，流失用户多于新增用户数，因此总用户数的减少主要与新增用户数减少有关

5.3 新老客户消费比

```
# 求出用户第一次购买与最后一次购买
user_life = data.groupby('user_id').order_dt.agg(['min', 'max'])
rate=(user_life['min'] == user_life['max']).value_counts()
rate
```

```
True      61656
False     23992
```

```
rate.index = ['只消费一次用户', '多次消费用户']
rate.plot.bar()
plt.xticks(rotation=0)
plt.show()
```

- 新老用户占比约为3:1，即有四分之三的用户只消费1次

5.4 用户分层（RFM分析以及KMeans聚类分析）

```
#RFM， 最近一次消费，消费次数（这里用购买产品总数代替），消费金额（数据透视）
rfm = data.pivot_table(index='user_id',
                        values=['order_dt', 'order_products', 'order_amount'],
                        aggfunc={
                            'order_dt': 'max',
                            'order_products': 'sum',
                            'order_amount': 'sum'
                        })
rfm.rename(columns={'order_amount': 'M', 'order_products': 'F'}, inplace=True)
#求最近一次消费距今（这里用日期最大值代替）的天数
rfm['R']=(rfm.order_dt.max()-rfm.order_dt)/np.timedelta64(1, 'D')
rfm = rfm.drop(columns='order_dt')
```

```
r_mean = rfm['R'].mean()
# 去掉最高的前100，排除极值干扰
temp_f = rfm.sort_values('F', ascending=False)['F'].reset_index(drop=True)
f_mean = temp_f[100:].mean()
# 去掉最高的前100，排除极值干扰
temp_m = rfm.sort_values('M', ascending=False)['M'].reset_index(drop=True)
m_mean = temp_m[100:].mean()
```

```
rfm_new = pd.DataFrame({
    'M': rfm['M'].apply(lambda x: x-m_mean),
    'F': rfm['F'].apply(lambda x: x-f_mean),
    'R': rfm['R'].apply(lambda x: x-r_mean)
})
```

```
def rfm_func(x):
    level = x.apply(lambda x: '1' if x>0 else '0')

    label = level.R + level.F + level.M
    d = {
        '111': '重要价值客户',
        '011': '重要保持客户',
        '101': '重要挽留客户',
        '001': '重要发展客户',
        '110': '一般价值客户',
        '010': '一般保持客户',
        '100': '一般挽留客户',
        '000': '一般发展客户'
    }
    result = d[label]
    return result
rfm_new['label'] = rfm_new[['R', 'F', 'M']].apply(rfm_func, axis=1)
rfm_new.head()
```

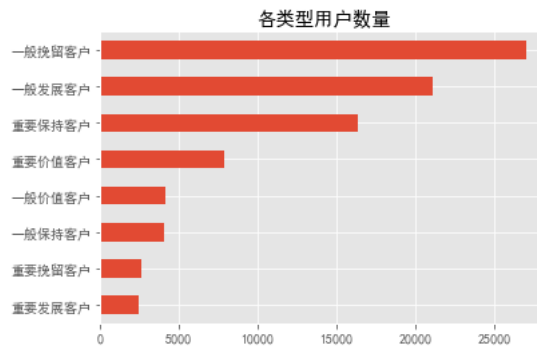
		M	F	R	label
user_id					
1	90.347859	2.395602	5.990449		重要价值客户
2	-599.062141	-0.604398	21.990449		一般挽留客户
3	-82.062141	0.395602	-3.009551		一般保持客户
4	914.937859	-0.604398	0.990449		重要挽留客户
5	1268.937859	1.395602	-11.009551		重要保持客户

- 重要价值客户：最近无消费，购买产品多，消费金额高
- 重要保持客户：最近有消费，购买产品多，消费金额高
- 重要挽留客户：最近无消费，购买产品少，消费金额高
- 重要发展客户：最近有消费，购买产品少，消费金额高
- 一般价值客户：最近无消费，购买产品多，消费金额低

- 一般保持客户：最近有消费，购买产品多，消费金额低
- 一般挽留客户：最近无消费，购买产品少，消费金额低
- 一般发展客户：最近有消费，购买产品少，消费金额低

```
print(rfm_new['label'].value_counts())
plt.title('各类型用户数量')
rfm_new['label'].value_counts().sort_values().plot.barh()
plt.show()
```

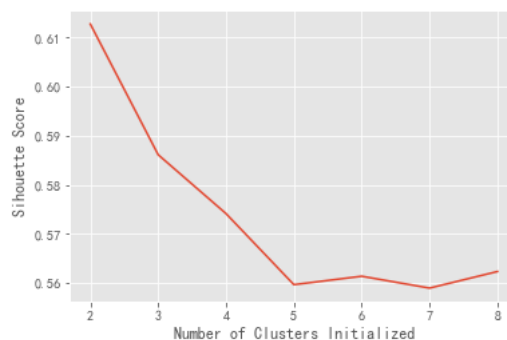
一般挽留客户	27062
一般发展客户	21111
重要保持客户	16388
重要价值客户	7895
一般价值客户	4145
一般保持客户	4036
重要挽留客户	2578
重要发展客户	2433



- 各类型用户数量前三的分别是一般挽留客户，一般发展客户，重要保持客户
- 层次较多，可能不便于管理，尝试使用KMeans进行聚类分类

使用轮廓系数评估不同K值 KMeans效果

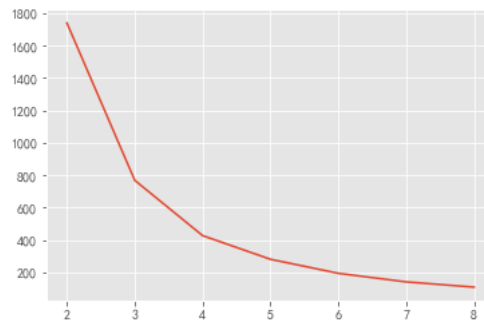
```
# 使用轮廓系数
from sklearn.cluster import KMeans
from sklearn import metrics
scores = []
for k in range(2,9):
    labels = KMeans(n_clusters=k, random_state=1).fit(rfm_new).labels_
    score = metrics.silhouette_score(rfm_new, labels)
    scores.append(score)
#可视化展示
plt.plot(list(range(2,9)), scores)
plt.xlabel('Number of Clusters Initialized')
plt.ylabel('Sihouette Score')
```



```
for k in range(0,7):
    print('k=',k+2,',Sihouette Score:', scores[k])
```

```
k= 2 ,Sihouette Score: 0.6127256266932026
k= 3 ,Sihouette Score: 0.5861070763696219
k= 4 ,Sihouette Score: 0.574116721028366
k= 5 ,Sihouette Score: 0.5596437722314277
k= 6 ,Sihouette Score: 0.5613572421530512
k= 7 ,Sihouette Score: 0.55894153167654
k= 8 ,Sihouette Score: 0.5623162932733812
```

```
# 使用均方误差
em = []
ks=range(2,9)
for k in ks:
    kc = KMeans(n_clusters=k, random_state=1)
    kc.fit(rfm_new)
    em.append(kc.inertia_)
plt.plot(ks, em)
```



```
for k in range(0,7):
    print('k=',k+2,', 均方根误差:', em[k])
```

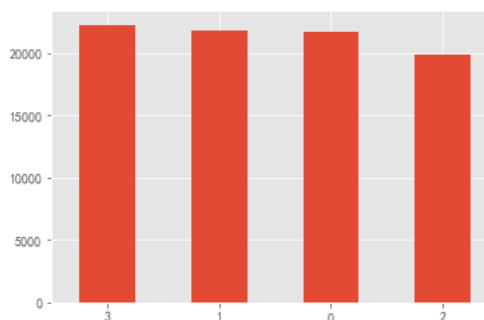
```
k= 2 , 均方根误差: 1739.8091407068264
k= 3 , 均方根误差: 768.5217704592015
k= 4 , 均方根误差: 427.28979600907695
k= 5 , 均方根误差: 281.18235440276686
k= 6 , 均方根误差: 194.4819635871585
k= 7 , 均方根误差: 141.6171363169188
k= 8 , 均方根误差: 109.24767243603569
```

- 结合轮廓系数和均方根误差函数，选K=4

```
from sklearn.cluster import KMeans
# 提取出R、F、M三个字段，对数据进行离差标准化
rfm_kmeans = rfm[['R', 'F', 'M']].apply(lambda x: (x-x.min()/(x.max()-x.min())))
# 转化为数组
rfm_kmeans = np.array(rfm_kmeans)
# 根据误差函数和轮廓系数，选K=4，迭代次数2000000000
model = KMeans(n_clusters=4, max_iter=2000000000, random_state=8)
model = model.fit(rfm_kmeans)
```

```
# 对原始数据表进行类别标记
rfm['label_kmeans'] = model.labels_
rfm['label_rfm'] = rfm_new['label']
print(rfm['label_kmeans'].value_counts())
rfm['label_kmeans'].value_counts().plot.bar()
plt.xticks(rotation=0)
plt.show()
```

```
3    22266
1    21822
0    21702
2    19858
```

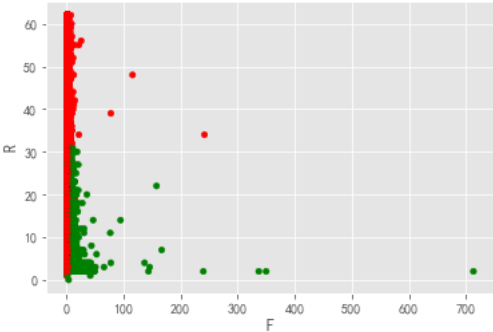


```
# 查看各类用户的消费总金额
rfm.groupby('label_rfm').sum()
```

	M	F	R	label_kmeans
label_rfm				
一般价值客户	2970749.20	8589	185309.0	5872
一般保持客户	3091411.57	8410	74548.0	7290
一般发展客户	12145968.57	21111	362576.0	33672
一般挽留客户	14330256.12	27062	1272079.0	41061
重要价值客户	13407321.90	19925	342873.0	10692
重要保持客户	35156287.03	53023	258530.0	22428
重要发展客户	3011006.13	2433	38850.0	3408
重要挽留客户	3222091.58	2578	121141.0	3913

- 重要保持客户贡献了3515万左右的消费额

```
#制作R与F关系散点图
rfm.loc[rfm.label_rfm=='重要保持客户', 'color'] = 'g'
rfm.loc[~(rfm.label_rfm=='重要保持客户'), 'color'] = 'r'
rfm.plot.scatter(x='F', y='R',c=rfm.color)
plt.show()
```



- 从RFM分层可知，重要保持客户占比较高，可根据情况选择均值、中位数、众数，或以业务出发选择相应的节点值对用户划分RFM模型

观察KMeans各个分类数据分布，进行用户分类

```
rfm[rfm['label_kmeans']==0].describe()
```

	M	F	R	label_kmeans
count	21702.000000	21702.000000	21702.000000	21702.0
mean	1414.952897	2.180951	9.025666	0.0
std	4650.203379	6.942370	4.245934	0.0
min	30.000000	1.000000	0.000000	0.0
25%	602.000000	1.000000	5.000000	0.0
50%	923.000000	1.000000	9.000000	0.0
75%	1510.000000	2.000000	13.000000	0.0
max	457232.000000	713.000000	16.000000	0.0

```
rfm[rfm['label_kmeans']==1].describe()
```

	M	F	R	label_kmeans
count	21822.000000	21822.000000	21822.000000	21822.0
mean	868.339966	1.476171	38.684676	1.0
std	1306.769671	1.898882	4.262003	0.0
min	35.000000	1.000000	32.000000	1.0
25%	478.000000	1.000000	35.000000	1.0
50%	687.000000	1.000000	39.000000	1.0
75%	1050.000000	2.000000	42.000000	1.0
max	162415.290000	242.000000	46.000000	1.0

```
rfm[rfm['label_kmeans']==2].describe()
```

	M	F	R	label_kmeans
count	19858.000000	19858.000000	19858.000000	19858.0
mean	754.431668	1.306325	54.246399	2.0
std	789.201663	1.083734	4.693879	0.0
min	18.000000	1.000000	47.000000	2.0
25%	429.000000	1.000000	50.000000	2.0
50%	603.000000	1.000000	54.000000	2.0
75%	888.000000	1.000000	58.000000	2.0
max	77072.820000	116.000000	62.000000	2.0

```
rfm[rfm['label_kmeans']==3].describe()
```

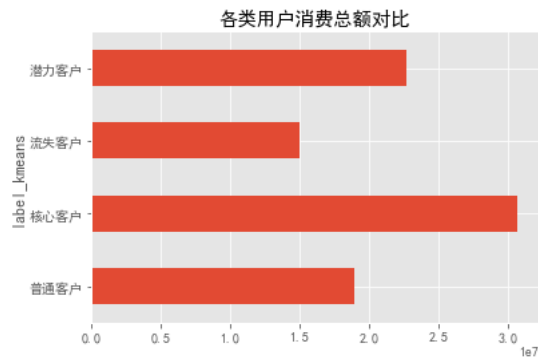
	M	F	R	label_kmeans
count	22266.000000	22266.000000	22266.000000	22266.0
mean	1019.373283	1.690739	24.190649	3.0
std	901.437929	1.551141	4.230193	0.0
min	32.000000	1.000000	17.000000	3.0
25%	539.000000	1.000000	21.000000	3.0
50%	798.000000	1.000000	25.000000	3.0
75%	1256.000000	2.000000	28.000000	3.0
max	56393.000000	158.000000	31.000000	3.0

- 0：核心客户（平均消费最高，频次多，最近一次消费时间小于16天）
- 1：普通客户（平均消费一般，频次一般，最近一次消费时间小于46天）
- 2：流失客户（平均消费最低，频次最低，最近一次消费时间小于62天）
- 3：潜力客户（平均消费较高，频次稍多，最近一次消费时间小于31天）

```
# 定义KMeans簇类
dict = {
    0 : '核心客户',
    1 : '普通客户',
    2 : '流失客户',
    3 : '潜力客户'
}
rfm['label_kmeans'] = rfm['label_kmeans'].apply(lambda x : dict[x])
rfm.head()
```

	M	F	R	label_kmeans	label_rfm	color
user_id						
1	1065.41	4	37.0	普通客户	重要价值客户	r
2	376.00	1	53.0	流失客户	一般挽留客户	r
3	893.00	2	28.0	潜力客户	一般保持客户	r
4	1890.00	1	32.0	普通客户	重要挽留客户	r
5	2244.00	3	20.0	潜力客户	重要保持客户	g

```
plt.title('各类用户消费总额对比')
rfm.groupby('label_kmeans').sum()['M'].plot.barh()
plt.show()
```



5.5 用户购买周期 (按订单)

```
pivoted_counts = data[data.month!='2016-05-01'].pivot_table(index='user_id', columns='order_dt', values='order_amount',  
aggfunc='count').fillna(0)  
pivoted_counts.head()
```

order_dt	2016-03-01 00:00:00	2016-03-02 00:00:00	2016-03-03 00:00:00	2016-03-04 00:00:00	2016-03-05 00:00:00	2016-03-06 00:00:00	2016-03-07 00:00:00	2016-03-08 00:00:00	2016-03-09 00:00:00	2016-03-10 00:00:00	...	2016-04-21 00:00:00	2016-04-22 00:00:00
user_id													
1	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	...	0.0	0.0
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0
5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0

5 rows × 61 columns

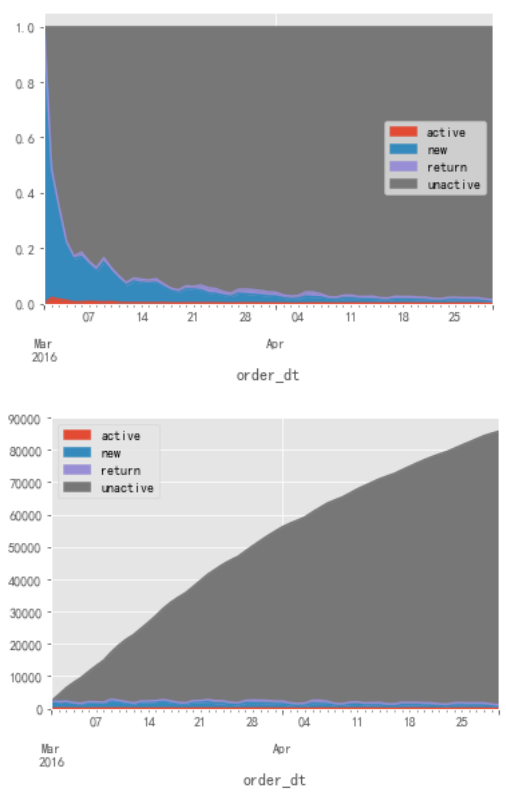
```
#消费过的为1，没消费过的为0  
data_purchase = pivoted_counts.applymap(lambda x: 1 if x>=1 else 0)  
#用函数，将用户分为 新 (new)、活跃 (active)、回流 (return) /不活跃 (unactive)、unreg  
def active_status(data):  
    status = []  
    for i in range(61):  
        #若为第一天  
        if i == 0 :  
            #若有购买  
            if data[i]>=1:  
                status.append('new')  
            #若无购买  
            else:  
                status.append('unreg')  
  
        #若不是第一天  
        else:  
            #若本日有购买  
            if data[i]>=1:  
                if status[i-1] == 'unreg':  
                    status.append('new')  
                elif status[i-1]=='unactive':  
                    status.append('return')  
                else:  
                    status.append('active')  
            #若本日无购买  
            else:  
                if status[i-1]=='unreg':  
                    status.append('unreg')  
                else:  
                    status.append('unactive')  
    return pd.Series(status, index=data.index)  
purchase_status = data_purchase.apply(active_status, axis=1, result_type='expand')  
purchase_status.head()
```

order_dt	2016-03-01 00:00:00	2016-03-02 00:00:00	2016-03-03 00:00:00	2016-03-04 00:00:00	2016-03-05 00:00:00	2016-03-06 00:00:00	2016-03-07 00:00:00	2016-03-08 00:00:00	2016-03-09 00:00:00	2016-03-10 00:00:00	...	2016-04-21 00:00:00	2016-04-22 00:00:00
user_id													
1	unreg	new	unactive	unactive	unactive	unactive	unactive	unactive	unactive	unactive	...	unactive	unactive
2	unreg	unreg	unreg	unreg	unreg	unreg	unreg	unreg	unreg	new	...	unactive	unactive
3	unreg	unreg	unreg	unreg	unreg	unreg	unreg	unreg	unreg	unreg	...	unactive	unactive
4	unreg	unreg	unreg	unreg	unreg	unreg	unreg	unreg	unreg	unreg	...	unactive	unactive
5	unreg	unreg	unreg	unreg	unreg	unreg	unreg	unreg	unreg	unreg	...	unactive	unactive

```
#将未注册为NaN，并求出每日各类用户数量
purchase_status_ct = purchase_status.replace('unreg', np.NaN).apply(lambda x:pd.value_counts(x))
purchase_status_ct
```

order_dt	2016-03-01 00:00:00	2016-03-02 00:00:00	2016-03-03 00:00:00	2016-03-04 00:00:00	2016-03-05 00:00:00	2016-03-06 00:00:00	2016-03-07 00:00:00	2016-03-08 00:00:00	2016-03-09 00:00:00	2016-03-10 00:00:00	...	2016-04-21 00:00:00	2016-04-22 00:00:00
active	NaN	93.0	99	90	51	66	92	90	90	122	...	58	60
new	2330.0	2023.0	2086	1653	1507	1868	1763	1658	2498	2155	...	926	836
return	NaN	NaN	63	66	83	192	166	185	286	244	...	757	546
unactive	NaN	2237.0	4191	6283	7958	9341	11209	12955	14512	17020	...	76029	77164

```
# 各个状态的占比
purchase_status_ct.T.fillna(0).apply(lambda x:x/x.sum(), axis=1).plot.area()
purchase_status_ct.T.fillna(0).plot.area()
```



- 由上表可知每日的用户消费状态变化情况
- 大部分用户属于不活跃用户，即当日消费，第二日不再购买的用户，这要结合商品的购买周期来看，因为只有两个月的数据，若商品购买周期为半年，就不能很好反应不活跃用户
- 每日新增用户初期都保持稳定，后3周开始下降
- 每日都有固定回流用户
- 每日都有固定活跃客户，即每天购买产品的忠诚客户

5.6 用户生命周期

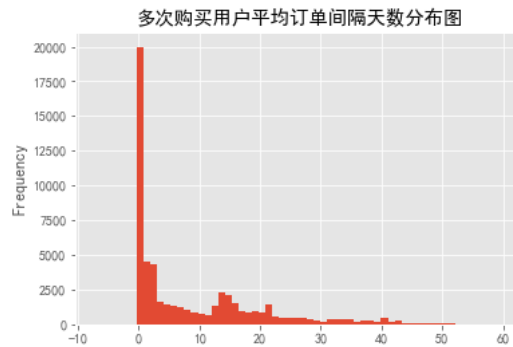
```
# 一个用户可能多个订单 每个订单的间隔
order_diff = data.groupby('user_id').apply(lambda x: x.order_dt - x.order_dt.shift())
```

```
order_diff.describe()
```

count	57127
mean	8
std	11
min	-7
25%	0
50%	3
75%	15
max	60

- 多次购买的用户平均订单间隔是8天 中位数3天 最大值60天

```
# 间隔天数分布图
(order_diff/np.timedelta64(1, 'D')).plot.hist(bins=60)
plt.title('多次购买用户平均订单间隔天数分布图')
plt.show()
```



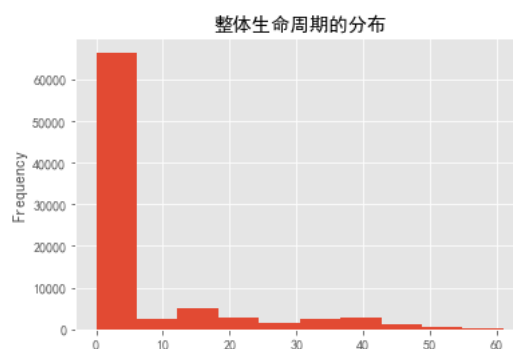
- 大部分用户平均订单间隔为3天以内，且以1天以内居多

```
#整体生命周期
user_life = data.groupby('user_id').order_dt.agg(['min', 'max'])
(user_life['max']-user_life['min']).describe()
```

count	85648
mean	5
std	12
min	0
25%	0
50%	0
75%	3
max	61

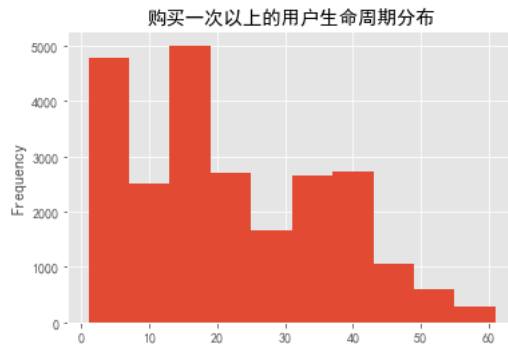
- 所有用户的平均生命周期为5天，中位数0天，最大值61天
- 说明大部分为只消费一次的用户，有四分之一的用户生命周期为3天到61天

```
#整体生命周期的分布
((user_life['max']-user_life['min'])/np.timedelta64(1, 'D')).plot.hist()
plt.title('整体生命周期的分布')
plt.show()
```



- 用户的生命周期受只购买一次的用户影响比较厉害（可以进行排除，只提取出购买一次以上的用户）

```
#提取出购买一次以上的用户
u_l = ((user_life['max']-user_life['min'])).reset_index()[0] / np.timedelta64(1, 'D')
u_l[u_l>0].plot.hist()
plt.title('购买一次以上的用户生命周期分布')
plt.show()
```

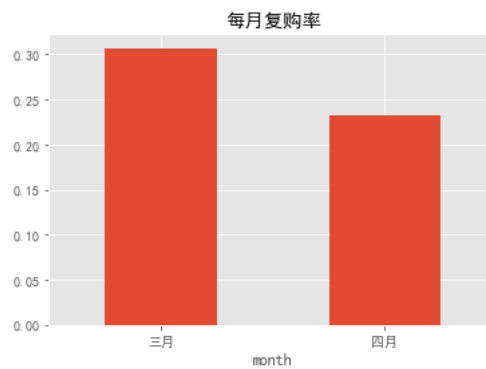



- 左边比较高，还是有些用户的生命周期是比较短的，例如5天以内
- 有不少用户生命周期也是比较稳定，例如15到20天用户占比最多

6.复购率和回购率分析

- 复购率
 - 自然月内，购买多次的用户占比
- 回购率
 - 曾经购买过的用户在某一时期内再次购买占比

```
#复购率
df1 = data[data.month!='2016-05-01'].groupby(['month', 'user_id']).count().order_dt.reset_index()
df1['ct'] = df1.order_dt.apply(lambda x: 1 if x>1 else 0)
df1.groupby('month').apply(lambda x: x.sum()/x.count()).ct.plot.bar()
plt.title('每月复购率')
plt.xticks([0,1], ['三月', '四月'], rotation=0)
plt.show()
```



- 三月份复购率约0.3，四月份约0.23，有所下降

```
#回购率
pivoted_counts = data[data.month!='2016-05-01'].pivot_table(index='user_id', columns='month', values='order_dt',
aggfunc='count').fillna(0)
df1 = pivoted_counts.applymap(lambda x: 1 if x>0 else 0)
def purchase_back(data):
    status = []
    for i in range(1):
        #本月购买
        #下月购买
        #下月无购买
        #本月无购买
        if data[i]==1:
            if data[i+1]==1:
                status.append(1)
            else:
                status.append(0)
        else:
            status.append(np.NaN)
    status.append(np.NaN)
    return pd.Series(status, index = data.index)
df1 = df1.apply(purchase_back, axis=1)
df1.head()
```

month	2016-03-01 00:00:00	2016-04-01 00:00:00
user_id		
1	0.0	NaN
2	0.0	NaN
3	NaN	NaN
4	0.0	NaN
5	1.0	NaN

- 0 表示本月购买了，次月没有进行消费，即没有回购
- 1代表本月消费过次月依旧消费，即回购了
- NaN表示当月没有消费（不进行计算）

```
(df1.sum()/df1.count())
```

month	
2016-03-01	0.285714
2016-04-01	NaN

- 只有两个月的数据，只能算出三月份回购率为0.28，与复购率0.3差不多，说明新客与老客整体质量差不多