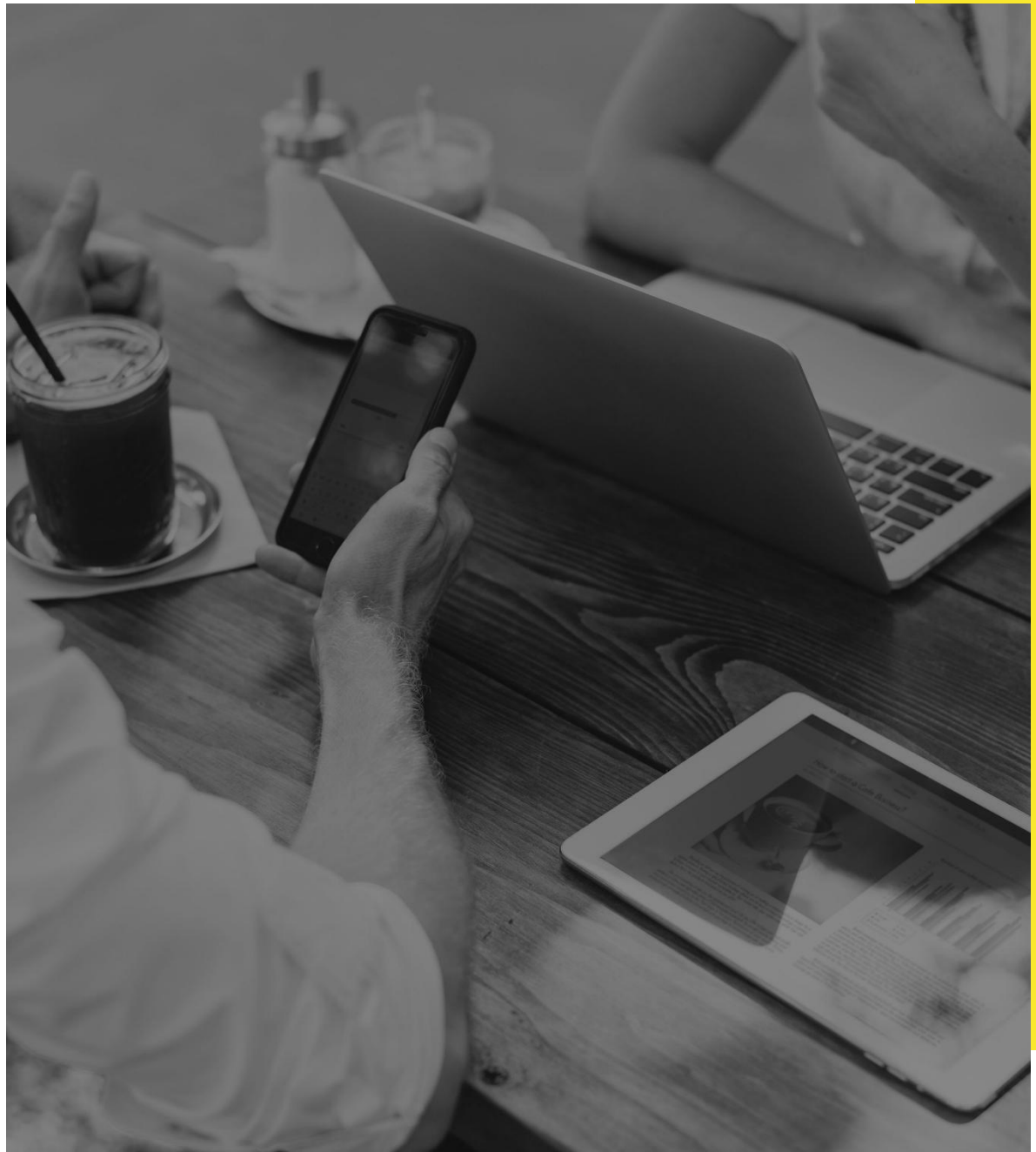
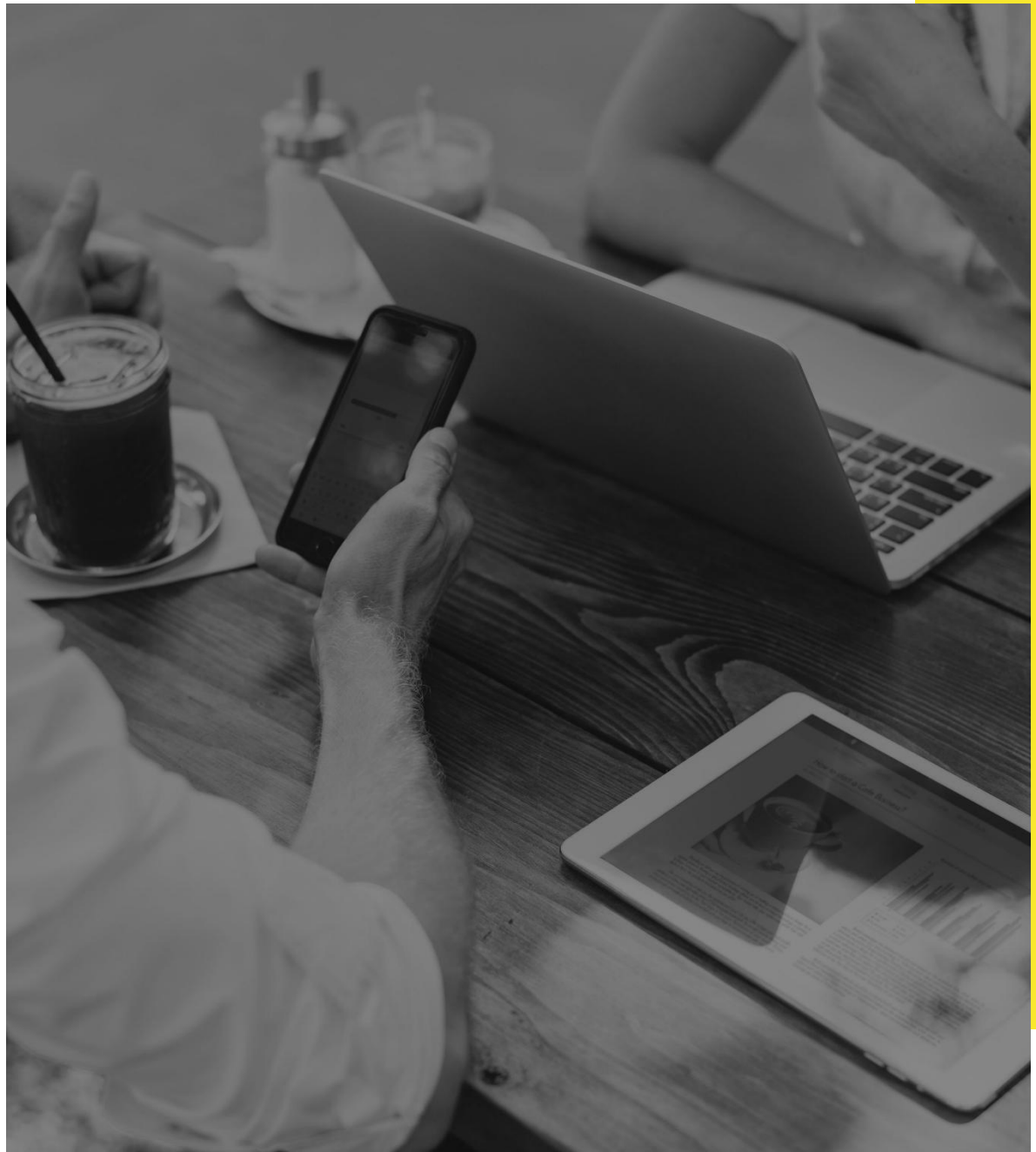
基于Python的RFM模型和K-Means算法实现用户价值分层



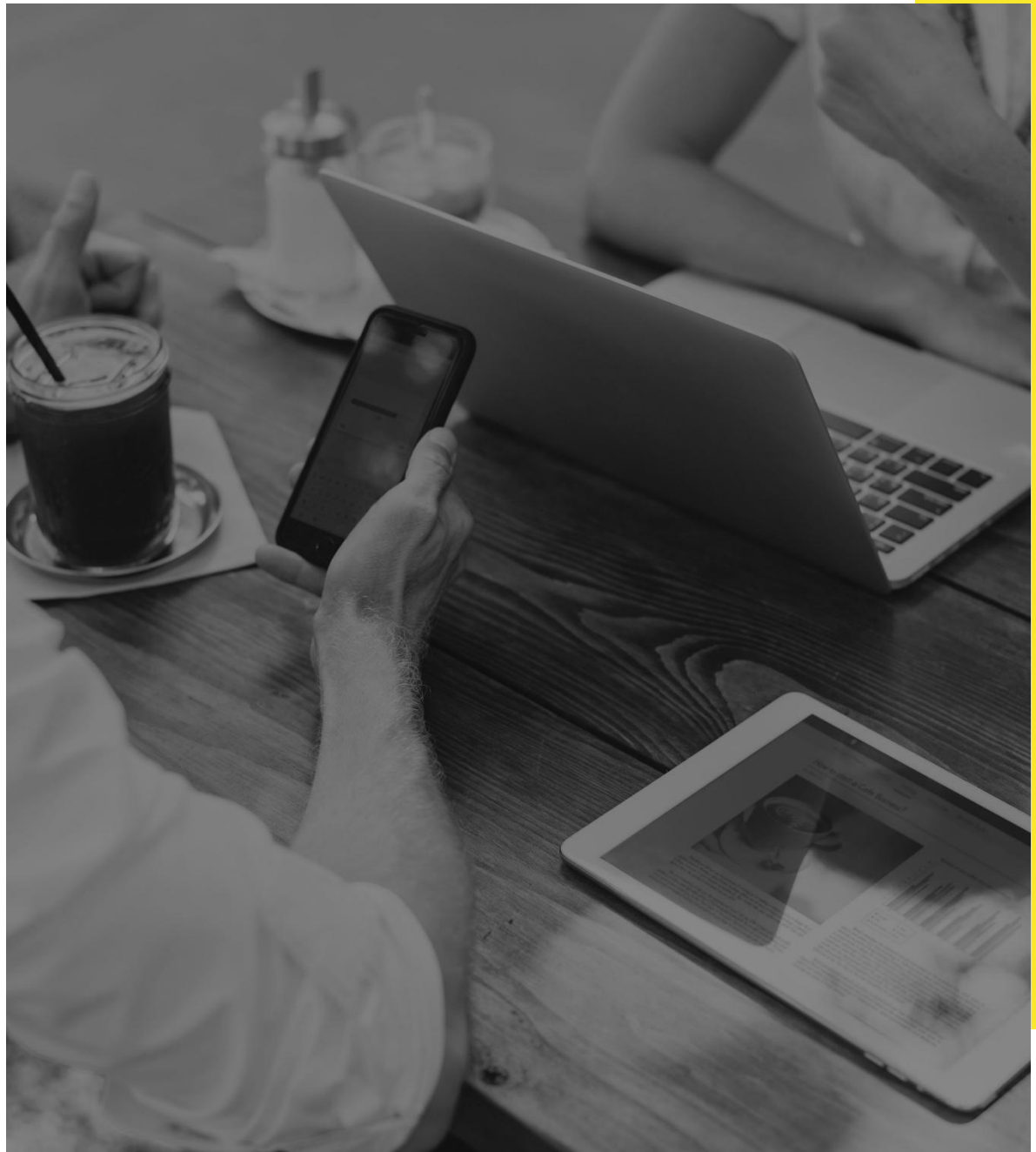


01

项目背景

项目背景

通过真实电商订单数据，采用 RFM模型与K-Means聚类算法对电商用户按照其价值进行分层。分析用户交易数据的用户行为特征，锁定最有价值的用户，从而实现个性化服务和运营。



02

理解数据

1. 数据来源
2. 数据简介
3. 字段含义

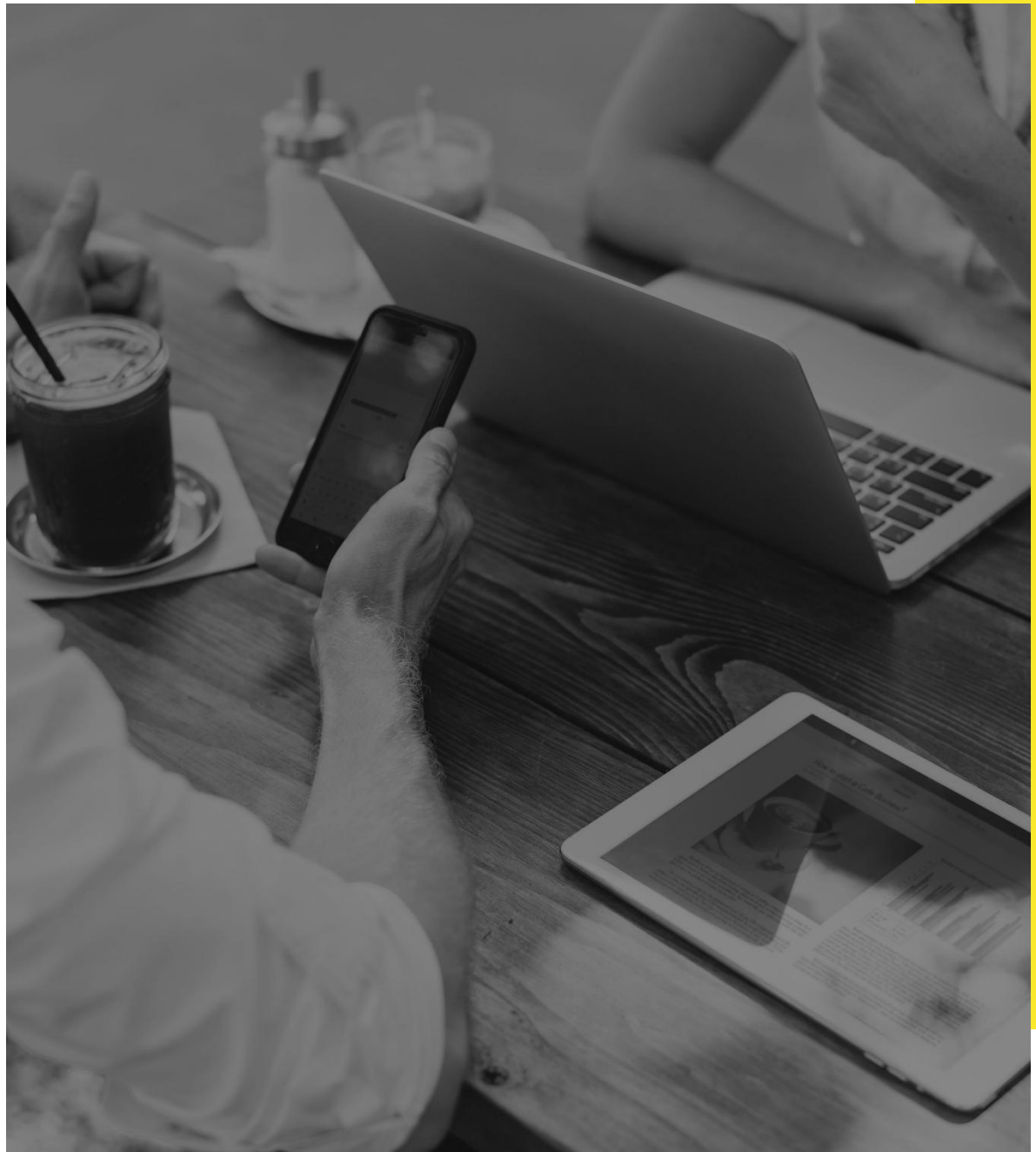
理解数据

数据来源
https://archive.ics.uci.edu/ml/datasets/online+retail#

数据简介
这是一个交易数据集，里面包含了在2010年12月1日至2011年12月9日之间所有网络交易订单信息。
数据集一共包含541909 条数据，8个字段。

字段含义

Columns	含义	数据类型
InvoiceNo	订单编号，由六位数字组成，对于每一笔交易唯一性，退货订单编号开头有字母'C	string
StockCode	产品编号，由五位数字组成，对于不同类别的产品是唯一的	string
Description	产品描述	string
Quantity	产品数量，负数表示退货	integer
InvoiceDate	订单日期与时间	datetime
UnitPrice	单价（英镑）	float
CustomerID	客户编号，由5位数字组成，对于每一个客户是唯一的	string
Country	国家，每个客户居住的国家/地区的名称	string



03

数据清洗

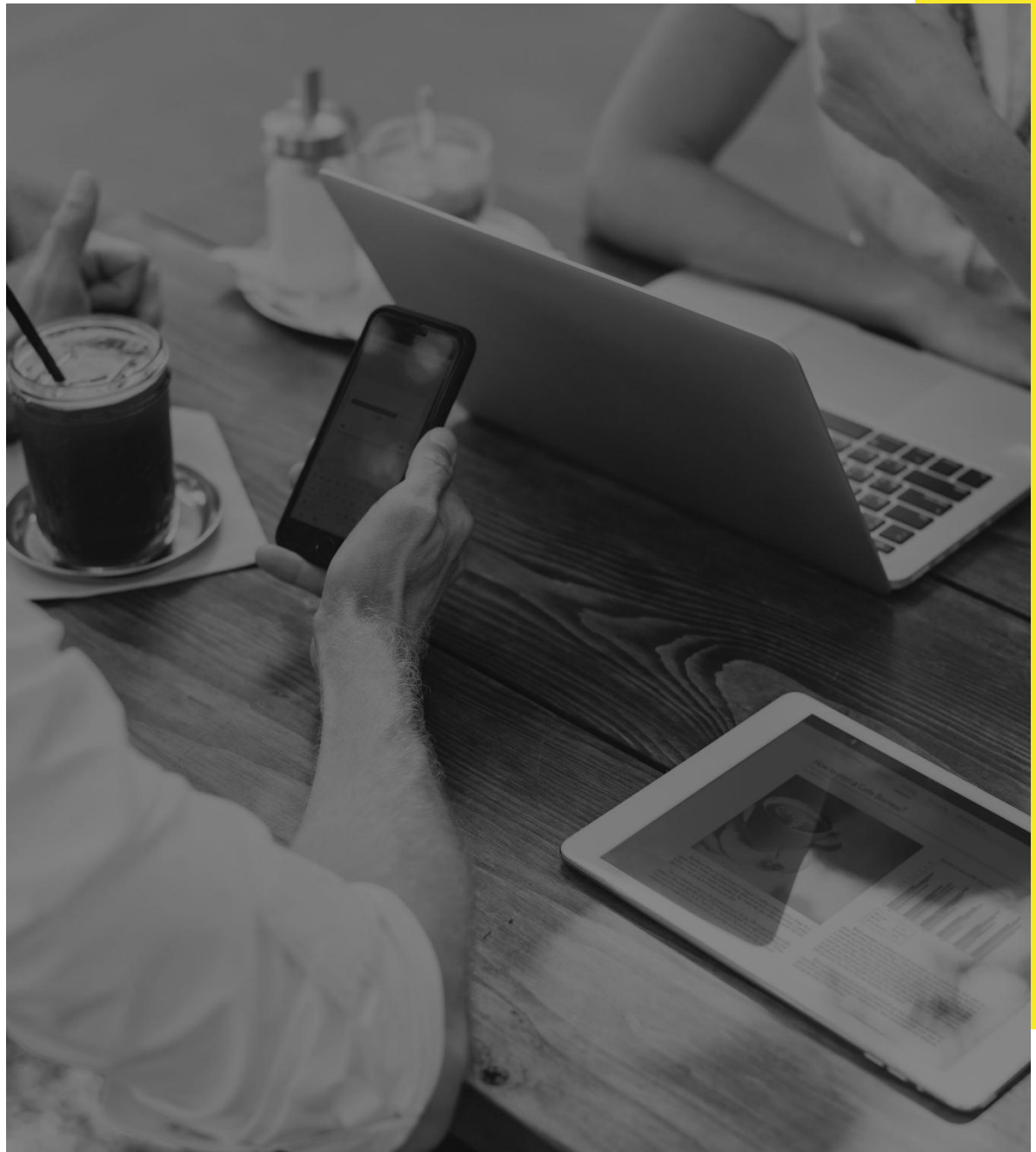
数据清洗

- 重复值处理　共删除5225条重复值
- 异常值处理

```
# Quantity是购买的产品数量，存在负数表示退货订单，这里分析不考虑退货订单信息，直接删除，UnitPrice是单价，不可能存在负值，直接删除异常值
sale = dataUni.loc[(dataUni['Quantity']>0) & (dataUni['UnitPrice']>0)]
```

- 辅助列

```
# 辅助列
sale['CustomerID'] = sale['CustomerID'].astype(int)
# 添加总价列,表示购买某一种商品的总额
sale['TotalSum'] = sale['UnitPrice']*sale['Quantity']
# 对时间属性做转换，保留年月日
sale['InvoiceDate'] = pd.to_datetime(sale.InvoiceDate)
sale['Date'] = sale['InvoiceDate'].dt.date
# 添加年,月,日,日期列
sale['Year'] = sale['InvoiceDate'].dt.year
sale['Month'] = sale['InvoiceDate'].dt.month
sale['Day'] = sale['InvoiceDate'].dt.day
sale.head()
```



04

RFM模型

1.指标计算

2.RFM模型搭建（方法一）

3.RFM模型搭建（方法二）

4.分析

指标计算

关于R,F,M值：
对于最近1次消费时间间隔R，上一次消费离的越近，也就是R值越小，用户价值越高。
对于消费频率F，购买的频率越高，也就是F的值越大，用户价值越高。
对于消费金额M，消费金额越高，也就是M的值越大，用户价值越高

计算R,F,M值：
根据定义进行运算出值后，进行分箱并评分。

	CustomerID	Recency	Frequency	MonetaryValue	Recency_Q	Frequency_Q	Moneytary_Q
0	12346	326	1	77183.60	1	1	5
1	12347	3	7	4310.00	5	4	5
2	12348	76	4	1797.24	2	3	4
3	12349	19	1	1757.55	4	1	4
4	12350	311	1	334.40	1	1	2

RFM模型搭建（方法一）

分别计算R值,F值,M值的中位数，每个指标与中位数进行比较，为每一个用户的R，F, M值进行高低维度的划分。
高用'H'表示，低用 ‘L’ 表示，高与低值针对用户的价值而言的。

R值若小于中位数，则为高，否则为低。

F值若大于中位数，则为高，否则为低。

M值若大于中位数，则为高，否则为低。

一共会得到8组分类

```
# 为R值打分划分高低
R_median = RFM_1['Recency_Q'].median()
RFM_1['R_label'] = pd.cut(RFM_1['Recency_Q'],bins=[0,R_median,RFM_1['Recency_Q'].max()+1],
                          right=False,labels=['L','H'])
RFM_1.groupby(['R_label'])['CustomerID'].count()

# 为F值打分划分高低
F_median = RFM_1['Frequency_Q'].median()
RFM_1['F_label'] = pd.cut(RFM_1['Frequency_Q'],bins=[0,F_median,RFM_1['Frequency_Q'].max()+1],
                          right=False,labels=['L','H'])
RFM_1.groupby(['F_label'])['CustomerID'].count()

# 为M值打分划分高低
M_median = RFM_1['Moneytary_Q'].median()
RFM_1['M_label'] = pd.cut(RFM_1['Moneytary_Q'],bins=[0,M_median,RFM_1['Moneytary_Q'].max()+1],
                          right=False,labels=['L','H'])
RFM_1.groupby(['M_label'])['CustomerID'].count()
```

	CustomerID	Recency	Frequency	MonetaryValue	Recency_Q	Frequency_Q	Moneytary_Q	RFM_Score	R_label	F_label	M_label	RFM_label	CustmerLevel
0	12346	326	1	77183.60	1.0	1.0	5.0	7	L	L	H	LLH	重要挽留客户
1	12347	3	7	4310.00	5.0	4.0	5.0	14	H	H	H	HHH	重要价值客户
2	12348	76	4	1797.24	2.0	3.0	4.0	9	L	H	H	LHH	重要保持客户
3	12349	19	1	1757.55	4.0	1.0	4.0	9	H	L	H	HLH	重要发展客户
4	12350	311	1	334.40	1.0	1.0	2.0	4	L	L	L	LLL	一般挽留客户

RFM模型搭建（方法二）

分别计算出R值打分，F值打分，M值打分的平均值，将每个指标与平均值进行比较，为每一个用户的R,F,M值进行高低维度的打分

高用'H'表示，低用 ‘L’ 表示。

一共会得到8组分类。

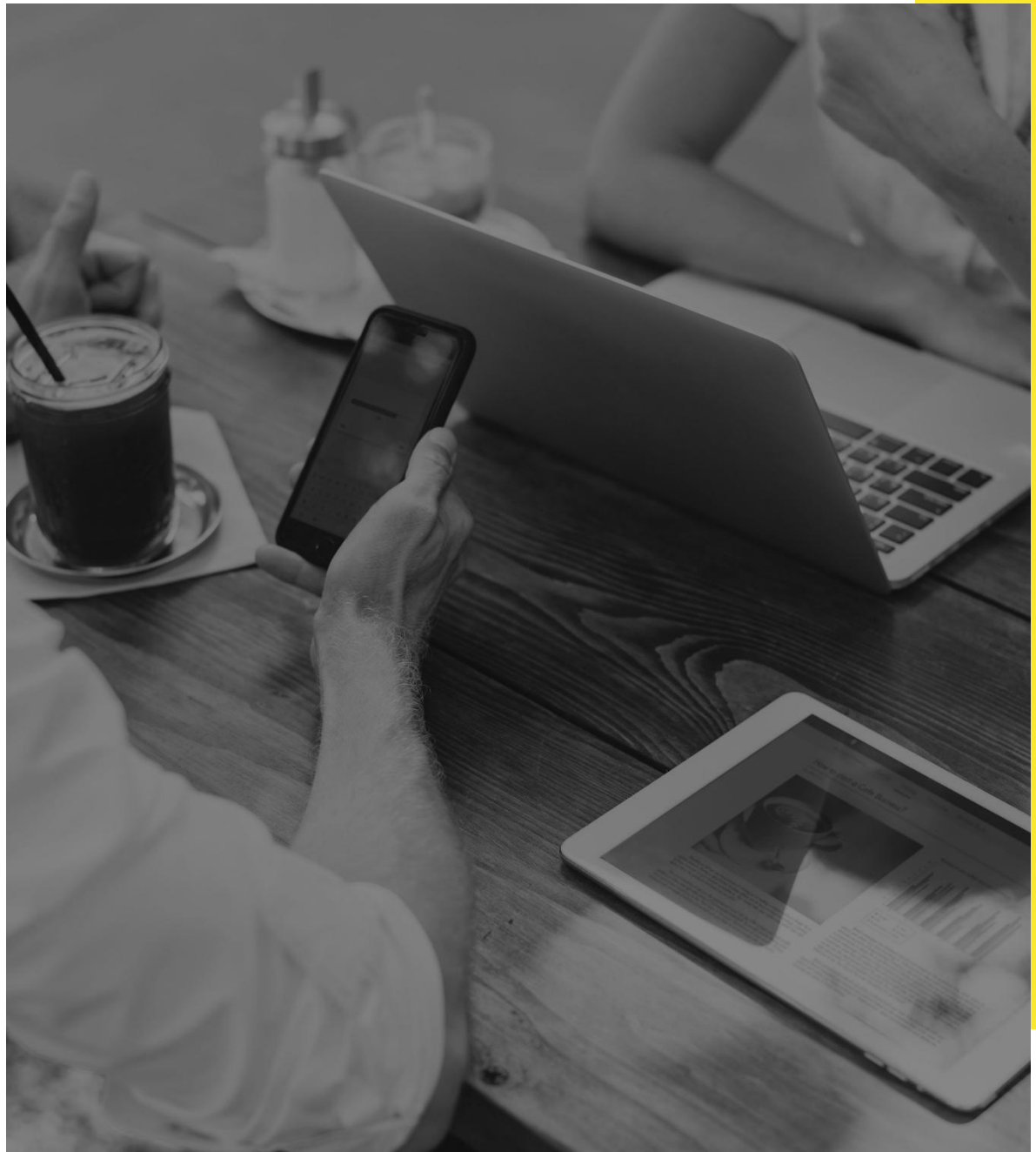
```
# 分别求均值
mean_R_score = RFM_2['Recency_Q'].mean()
mean_F_score = RFM_2['Frequency_Q'].mean()
mean_M_score = RFM_2['Moneytary_Q'].mean()
print(mean_R_score,mean_F_score,mean_M_score)

# R, F, M高低划分
RFM_2['R_label'] = pd.cut(RFM_2['Recency_Q'],bins=[0,mean_R_score,6],
                          right=True,labels=['L','H'])
RFM_2['F_label'] = pd.cut(RFM_2['Frequency_Q'],bins=[0,mean_R_score,6],
                          right=True,labels=['L','H'])
RFM_2['M_label'] = pd.cut(RFM_2['Moneytary_Q'],bins=[0,mean_M_score,6],
                          right=True,labels=['L','H'])
RFM_2.head()
```

	CustomerID	Recency	Frequency	MonetaryValue	Recency_Q	Frequency_Q	Moneytary_Q	RFM_Score	R_label	F_label	M_label	RFM_label	CustmerLevel
0	12346	326	1	77183.60	1.0	1.0	5.0	7	L	L	H	LLH	重要挽留客户
1	12347	3	7	4310.00	5.0	4.0	5.0	14	H	H	H	HHH	重要价值客户
2	12348	76	4	1797.24	2.0	3.0	4.0	9	L	L	H	LLH	重要挽留客户
3	12349	19	1	1757.55	4.0	1.0	4.0	9	H	L	H	HLH	重要发展客户
4	12350	311	1	334.40	1.0	1.0	2.0	4	L	L	L	LLL	一般挽留客户

分析

根据R,F,M高低区分的8类客户，可以根据其特点，给出针对性的营销策略
结合图表，总结如下：

用户分类	行为特征	精细化运营
重要价值客户	近期购买过，购买频率高，客单价较高，消费金额高，为主要消费客户	升级为VIP客户，提供个性化服务，倾斜较多的资源
重要发展客户	近期购买过，购买频率低，客单价高，消费金额较高，可能是新的批发商或企业采购者，想办法提高消费频率。	提供会员积分服务，给与一定程度的优惠来提高留存率
重要保持客户	近期没有购买过，购买频率较高，客单价较高，消费金额较高，可能是一段时间没来的忠实客户。	通过短信邮件等方式主动和客户保持联系，介绍最新产品/功能，提高复购率
重要挽留客户	近期没有购买，购买频率低，客单价高，消费金额较高，这种用户即将流失	通过电话短信等方式主动联系用户，调查清楚哪里出现问题，避免流失
一般价值客户	近期购买过，购买频率较高，客单价低，消费较低	潜力股， 提供社群服务，介绍新产品/功能促进消费
一般发展客户	近期购买过，购买频率低，客单价较低，消费较低，可能是新用户	提供社群服务，介绍新产品/功能，提供折扣等提高留存率
一般保持客户	近期没有购买过，购买频率较高，，客单价低，消费低	介绍新产品/功能等方式唤起此部分用户
一般挽留客户	近期没有购买过，购买频率低，客单价较低，消费低，已流失	通过促销折扣等方式唤起此部分用户，当资源分配不足时可以暂时放弃此部分用户



05

K-Means聚类

1. 数据预处理
2. K-Means建模

数据预处理

- 对数变换
- 标准化处理

```
# 数据预处理
# 将等于0的值替换成1，否则log变换后会出现无穷大的情况
RFM.Recency[RFM['Recency']==0]=0.01
RFM.Frequency[RFM['Frequency']==0]=0.01
RFM.MonetaryValue[RFM['MonetaryValue']==0]=0.01
RFM_log = RFM[['Recency','Frequency','MonetaryValue']].apply(np.log,axis=1).round(3)
RFM.head()
```

```
from sklearn.preprocessing import StandardScaler    # 标准化
scaler = StandardScaler()
scaler.fit(RFM_log)
RFM_normalization = scaler.transform(RFM_log)
```

	R	F	M
0	1.435500	-1.048593	3.700239
1	-1.953082	1.111983	1.413494
2	0.383073	0.490234	0.719941
3	-0.619479	-1.048593	0.702503
4	1.401527	-1.048593	-0.613267

K-Means建模

- K值的选取

绘制每个k值对应的inertia_，使用轮廓系数评估聚类效果—轮廓系数的区间为：[-1, 1]
结合calinski-harabaz Index，结果说明了分类2类效果好，其次是3类，但是不符合业务诉求，分3类效果次之，验证k=3。

```
# 模型计算—分为3类
kc = KMeans(n_clusters=3, random_state=1)
kc.fit(rfm_data)
#  每个样本对应的类簇标签，顺序与样本原始顺序一致
cluster_label = kc.labels_
RFM['K-means_label'] = cluster_label
RFM.head()
```

	CustomerID	Recency	Frequency	MonetaryValue	K-means_label
0	12346	326	1	77183.60	1
1	12347	3	7	4310.00	2
2	12348	76	4	1797.24	1
3	12349	19	1	1757.55	1
4	12350	311	1	334.40	0

K-Means建模

- 分析

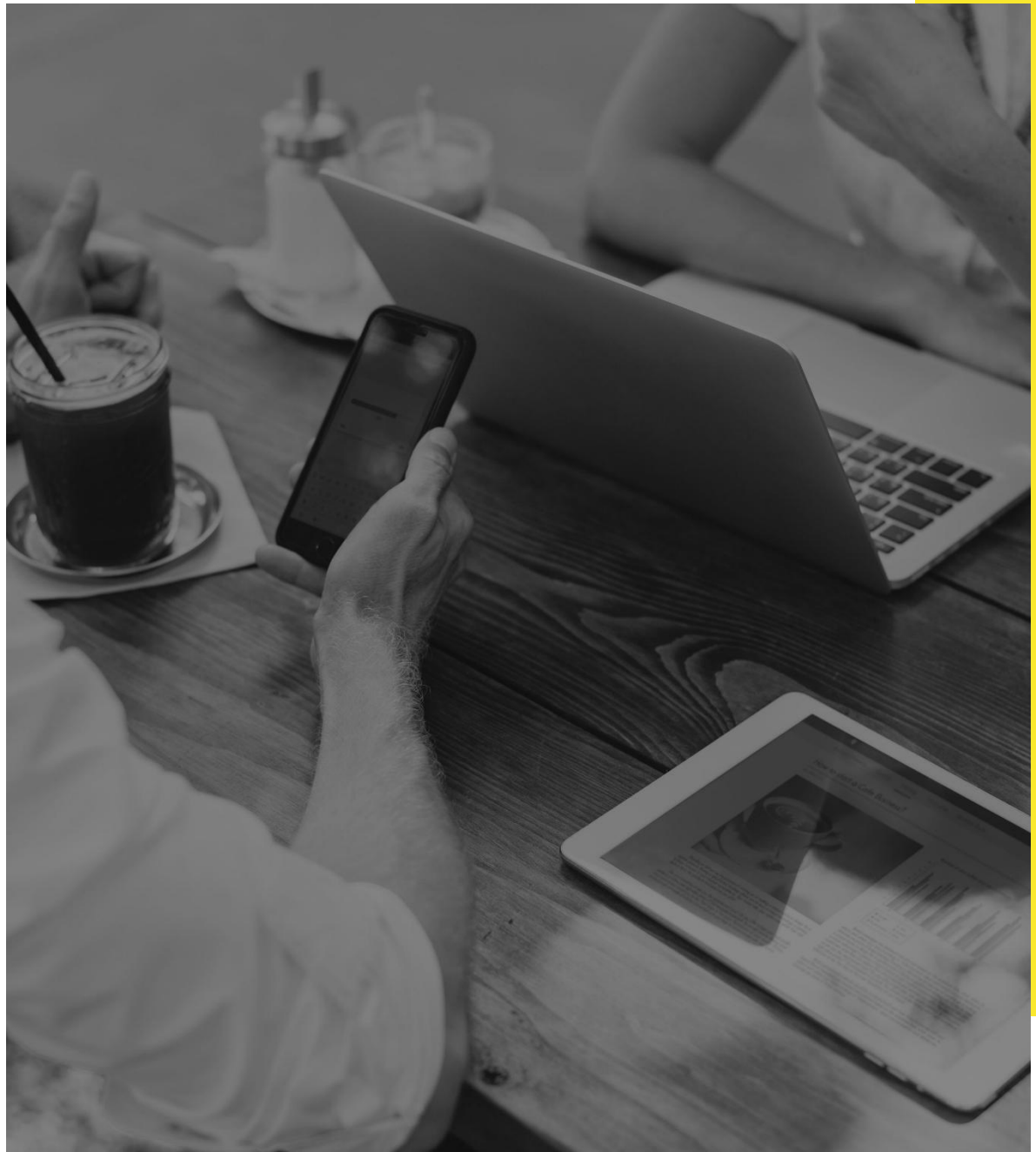
```
]: # 分析
# 选出每个客户的消费总金额，订单总量，购买的商品总数
data_temp3 = sale_F.groupby(['CustomerID']).agg({'InvoiceNo':'count'}).reset_index()
data_temp4 = sale.groupby(['CustomerID']).agg({'TotalSum':'sum','Quantity':'sum'}).reset_index()
kmeans_data = pd.concat([RFM['K-means_label'],data_temp4,data_temp3['InvoiceNo'],RFM['Recency'],RFM['Frequency']],axis=1)
kmeans_data.columns = ['客户等级','客户编号','消费金额', '购买商品总量', '订单总量', '最近消费天数','消费次数']
rfm_level  = customer_level_statistic(kmeans_data)
rfm_level
```

	消费金额			购买商品总量			订单总量			客户等级		最近消费天数	消费次数	客单价
	均值	总量	占比	均值	总量	占比	均值	总量	占比	总量	占比	均值	均值	均值
客户等级														
2	7304.5	6201531.9	69.8%	4103.5	3483843	67.6%	12.5	10589	57.1%	849	19.6%	15.4	12.5	585.7
1	1242.3	2074694.3	23.3%	775.9	1295835	25.2%	3.4	5644	30.5%	1670	38.5%	54.8	3.4	367.6
0	335.9	610982.7	6.9%	204.7	372324	7.2%	1.3	2299	12.4%	1819	41.9%	164.4	1.3	265.8

观察可知：
2类客户数量占比19.5%，占比最少，但是创造了近70%的消费金额，57.1%的订单总量，是主要的消费客户。

1类客户数量占比38.5%，消费金额占比23.3%，订单总量占比30.5%，对平台具有一定的价值。

0类客户数量占比41.9%，总人数最多，仅创造6.9%的消费金额，12.4% 的订单总量，最近消费天数均值已经超过5个月了，消费频率低，已基本流失。



06

总结

总结

本次分析主要使用Python语言对某份英国电子零售企业的交易数据进行数据挖掘，使用RFM模型和K-Means聚类算法对用户进行分层，寻找有价值的用户

无论是进行传统的RFM模型搭建还是使用聚类算法，都能将用户进行分层，在进行传统的RFM模型搭建的时候，使用两种方法来对用户进行分层，两个方法得出的结果有所差距，需要结合具体的业务来衡量所搭建模型的好坏。使用K-Means聚类算法也能在一定程度上将用户分层。但这两大类方法都有使用场景，也都有局限性。

- RFM模型得到的不同层级的客户，可以采取针对性措施进行营销，但销售场景受限
- 聚类算法可以较好的区分出各层用户，对于业务来说解释性还不够，数据更新前后的两次聚类结果会不同。


thanks