**

**实验报告**

**课程名称 数据分析课程设计**

**实验项目名称 实验2 聚类分析：客户价值分析**

**班级与班级代码**

**实验室名称**

**专 业 大数据管理与应用**

**任课教师 周瑾**

**学 号： 22251109109**

**姓 名：**  黄文溥

**实验日期： 2025 年 3 月 13 日**

广东财经大学教务处 制

**姓名 黄文溥**   **实验报告成绩**

**评语：**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 实验评价 | 优 | 良 | 中 | 及格 | 不及格 |
| 实验准备（10%） |  |  |  |  |  |
| 实验报告格式规范（10%） |  |  |  |  |  |
| 实验内容完成（60%） |  |  |  |  |  |
| 实验心得、问题及解决（20%） |  |  |  |  |  |

**指导教师（签名）**

**年 月 日**

说明：指导教师评分后，实验报告交院（系）办公室保存。

**实验2 基于RFM分析的客户分群**

**一、实验目的：**

1掌握RFM模型，能够进行客户价值识别

2 掌握k-means聚类

3.掌握EM聚类（基于高斯混合模型的EM聚类）（选做）

**二、知识准备：**

RFM模型是衡量客户价值和客户创利能力的重要工具和手段。在客户分类中，RFM模型是一个经典的分类模型，利用通用交易环节中最核心的三个维度——最近消费(Recency)、消费频率(Frequency)、消费金额(Monetary)细分客户群体，从而分析不同群体的客户价值。

**三、实验准备**

1.使用算法：RFM模型、聚类算法

**2. 数据来源**

RFM数据集为英国在线零售商在2010年12月1日至2011年12月9日间发生的所有网络交易订单信息。该公司主要销售礼品为主，并且多数客户为批发商。"This is a transnational data set which contains all the transactions occurring between 01/12/2010 and 09/12/2011 for a UK-based and registered non-store online retail.The company mainly sells unique all-occasion gifts. Many customers of the company are wholesalers."

数据集特征说明：

InvoiceNo:订单编号，由六位数字组成，退货订单编号开头有字母C

StockCode:产品编号，由五位数字组成

Description：产品描述

Quantity：产品数量，负数表示退货

InvoiceDate：订单日期与时间

UnitPrice ：单价（英镑）

CustomerID：客户编号，由5位数字组成

Country：国家

**数据集介绍及来源：**

<https://archive.ics.uci.edu/ml/datasets/online+retail#>

<https://www.kaggle.com/datasets/carrie1/ecommerce-data>

**3.后续学习：客户分类**

**<https://www.kaggle.com/fabiendaniel/customer-segmentation>**

**四、实验步骤：**

**1、提出问题，确定目标**

对在线零售商客户数据，探讨如何利用KMeans算法（EM聚类）对客户群体进行细分，以及细分后如何利用RFM模型对客户价值进行分析，并识别出高价值客户。主要希望实现以下三个目标：

1）对客户进行群体分类

2）对不同的客户群体进行特征分析，比较各细分群体的客户价值

3）对不同价值的客户制定相应的运营策略

1. **数据获取**

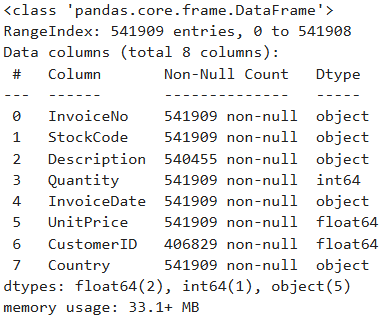
<https://archive.ics.uci.edu/ml/datasets/online+retail#>

<https://www.kaggle.com/datasets/carrie1/ecommerce-data>

1. **数据预处理**

1）数据初步信息获取：

df = pd.read\_csv("customer\_data.csv")

df.info()  


2）数据清洗：

重复值：

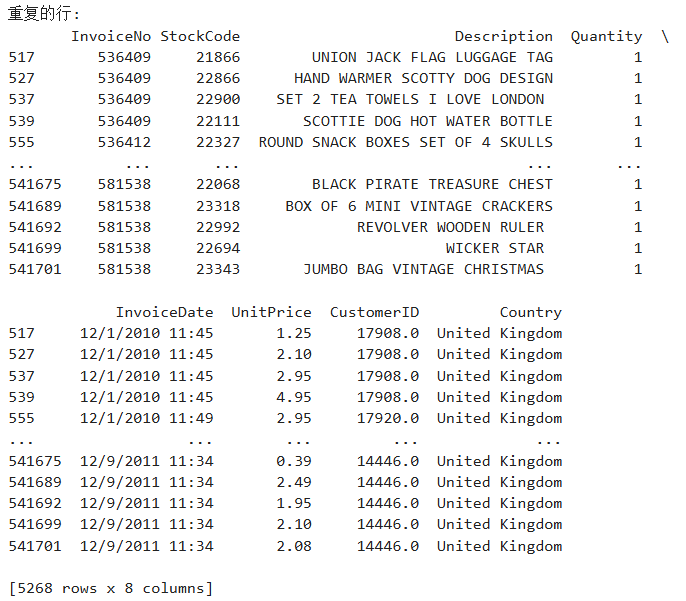
duplicated\_df = df.duplicated(keep='first')

removed\_rows = df[duplicated\_df]

print('重复的行:')

print(removed\_rows)

df = df.drop\_duplicates()



缺失值：鉴于仅讨论客户价值，数据中StockCode，Description，Country属性直接删除

df = df.drop(columns=['StockCode','Country','Description'],axis=1)

# 检测空值

null\_values = df.isnull().sum()

# 描述性统计数据

desc\_stats = df.describe().round(2)

print('\n空值：')

print(null\_values)

print('\n描述性统计数据：')

print(desc\_stats)



经过检查大致数据分布后，发现存在Quantity小于等于0，UnitPrice为负数，CustomerID为空值，InvoiceNo不统一为下单而是包含退单的情况，现对其进行相应的处理

首先判断退单的每条记录存在的规律，如Quantity为负数且InvoiceNo为C开头

negative\_quantity = df[df['Quantity'] <= 0]

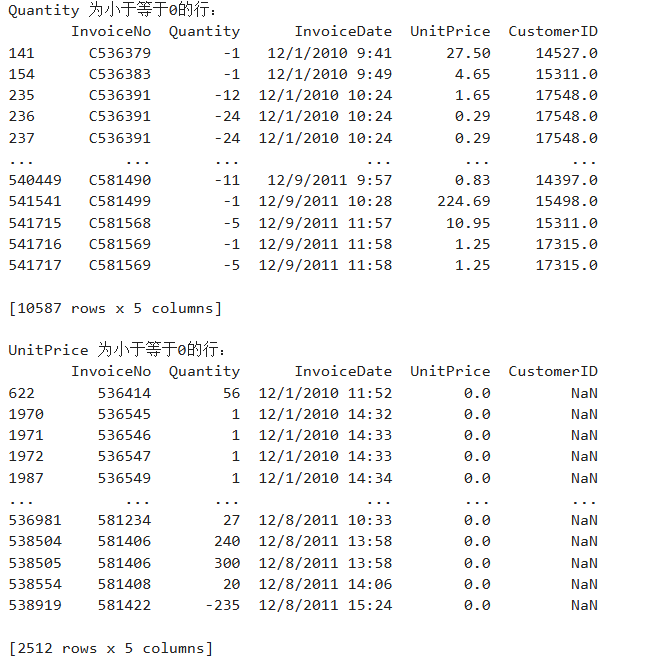
print('Quantity 为小于等于0的行：')

print(negative\_quantity)

negative\_unit\_price = df[df['UnitPrice'] <= 0]

print('\nUnitPrice 为小于等于0的行：')

print(negative\_unit\_price)



对这些退单数据进行标记，并对其对应的下单数据标记，即数据中存在除去以C开头的差异外其余编码相同的情况，为便于下面的探讨此处暂不进行处理

# InvoiceNo中以 'C' 开头的行以及对应的下单行

df['InvoiceNo'] = df['InvoiceNo'].astype(str)

c\_start\_rows = df['InvoiceNo'].str.startswith('C')

related\_invoice\_nos = df[c\_start\_rows]['InvoiceNo'].str[1:]

related\_rows = df['InvoiceNo'].isin(related\_invoice\_nos)

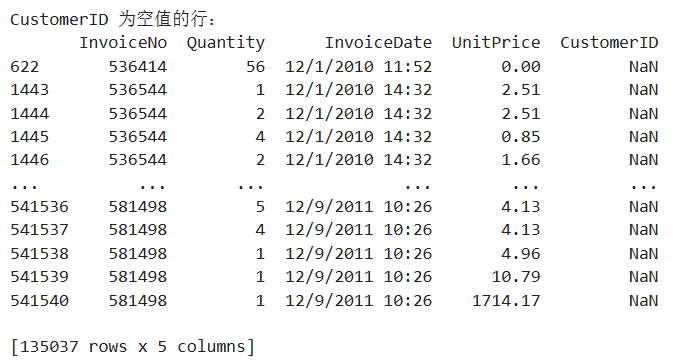
接着对空值检查并进行处理

null\_customer\_id = df[df['CustomerID'].isnull()]

print('\nCustomerID 为空值的行：')

print(null\_customer\_id)

df = df.drop(df[df['CustomerID'].isnull()].index,axis=0)



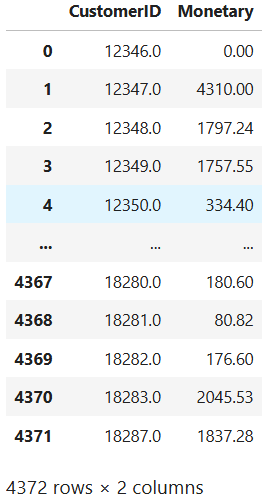
1. **初步构建客户价值模型并对相应指标进行可视化展示**

1）计算M:计算Quantity和UnitPrice的乘积，直接根据CustomerID分组计算总的金额

group = df.groupby('CustomerID')

df['TotalPrice'] = df['Quantity'] \* df['UnitPrice']

monetary = group['TotalPrice'].sum().rename('Monetary').to\_frame().reset\_index()



展示Monetary可视化视图

# 计算描述性统计信息，并保留两位小数

result = monetary[['CustomerID', 'Monetary']].describe().round(2)

print(result)

# 创建一个包含两个子图的画布

fig, axes = plt.subplots(1, 2, figsize=(12, 5))

# 绘制 Monetary 的直方图

axes[0].hist(monetary['Monetary'], bins=20, color='skyblue')

axes[0].set\_title('Monetary 直方图')

axes[0].set\_xlabel('Monetary')

axes[0].set\_ylabel('频数')

# 绘制 CustomerID 与 Monetary 的散点图

axes[1].scatter(monetary['CustomerID'], monetary['Monetary'], color='orange', alpha=0.5)

axes[1].set\_title('CustomerID 与 Monetary 散点图')

axes[1].set\_xlabel('CustomerID')

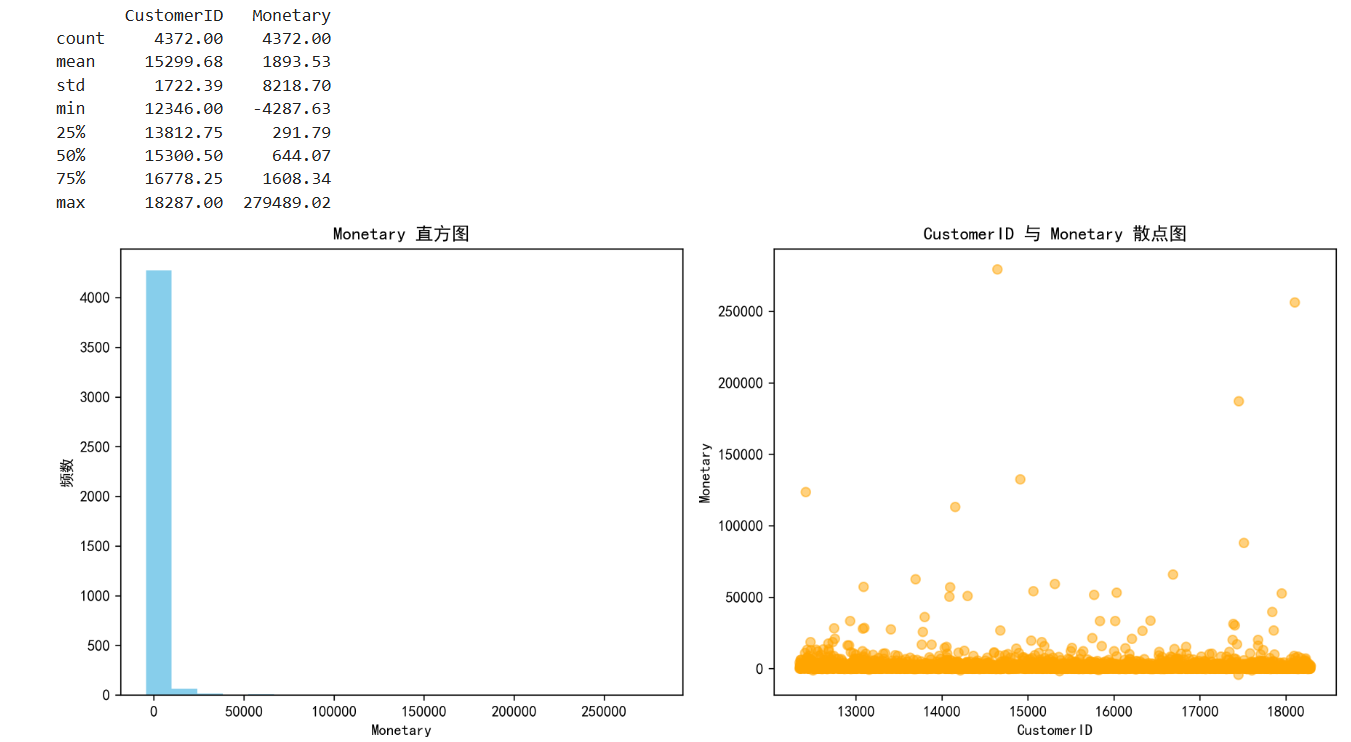
axes[1].set\_ylabel('Monetary')

# 自动调整子图布局

plt.tight\_layout()

# 显示图形

plt.show()



可见Monetary大体分布在0-2000

1. 计算R：使用2012/01/01 00:00的时间戳，减去每一顾客ID中最大的InvoiceDate

鉴于退单的记录操作时间可能晚于下单的操作时间，这里需要删除先前标记的C开头的InvoiceNo对应的记录以保证计算Recency时以最后一单最晚操作记录为准，Recency使用时间戳差值记录

df = df[~(c\_start\_rows)]

group = df.groupby('CustomerID')

df['InvoiceDate'] = pd.to\_datetime(df['InvoiceDate'])

def calculate\_recency(group):

td = pd.Timestamp('2012-01-01 00:00:00')

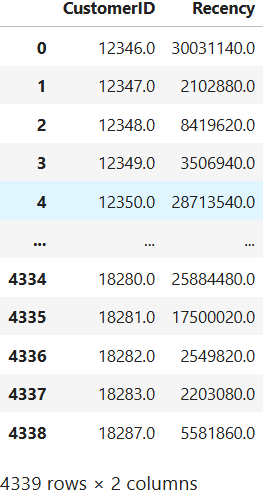
tts = td.timestamp()

md = group['InvoiceDate'].max()

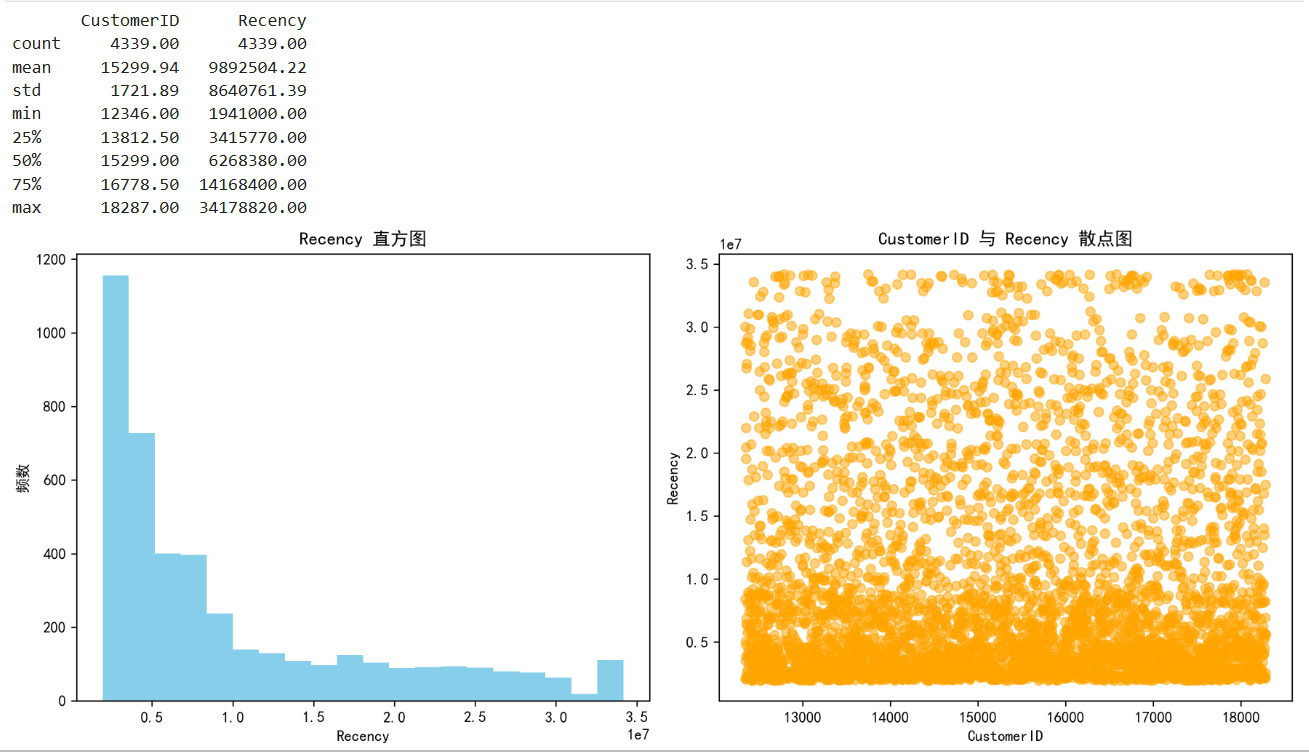
mts = md.timestamp()

return tts - mts

recency = group.apply(calculate\_recency).rename('Recency').reset\_index()

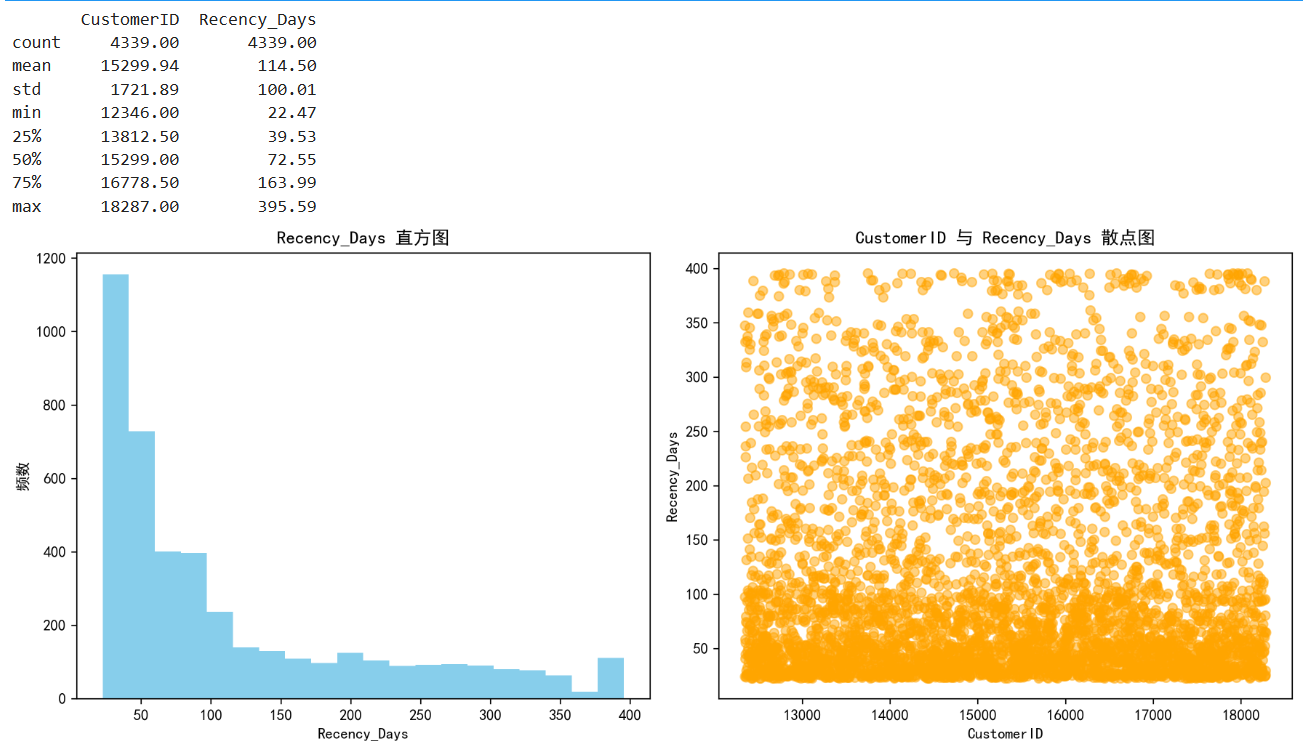


展示Recency可视化视图



对Recency进行换算为天数的可视化视图

recency['Recency\_Days'] = [x/86400 for x in recency['Recency']]



1. 计算F:根据分组结果，统计InvoiceNo的类别数以及总数

通常认为每个客户的所有有效消费记录才能用于计算Frequency，所以需要使用先前标记的数据删除退单的记录以及其对应下单记录，此处统计了两种F值，一种InvoiceNo\_Category\_Count即客户累计下单数，另一种为InvoiceNo\_Total\_Count即购入累计下单商品数

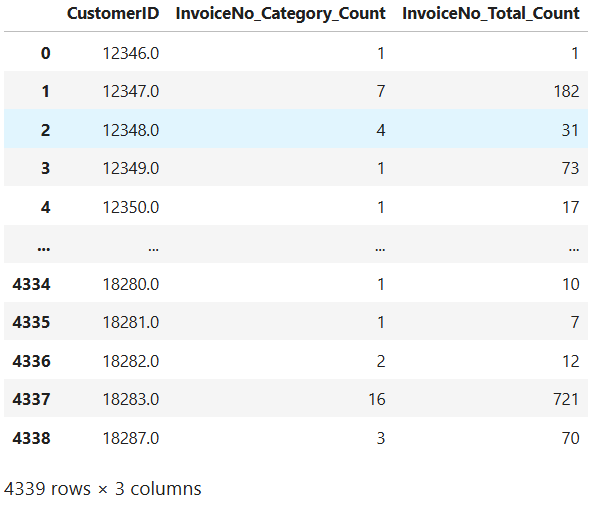
df = df[~(c\_start\_rows|related\_rows)]

group = df.groupby('CustomerID')

frequency = group['InvoiceNo'].agg(['nunique', 'count']).rename(

columns={'nunique': 'InvoiceNo\_Category\_Count', 'count': 'InvoiceNo\_Total\_Count'})

frequency = frequency.reset\_index()



# 计算描述性统计信息，并保留两位小数

result = frequency[['CustomerID', 'InvoiceNo\_Category\_Count', 'InvoiceNo\_Total\_Count']].describe().round(2)

print(result)

# 创建一个 1 行 3 列的子图画布

fig, axes = plt.subplots(1, 3, figsize=(15, 5))

# 绘制 InvoiceNo\_Category\_Count 的直方图

axes[0].hist(frequency['InvoiceNo\_Category\_Count'], bins=20, color='skyblue')

axes[0].set\_title('发票数量直方图')

axes[0].set\_xlabel('发票数量')

axes[0].set\_ylabel('频数')

# 绘制 InvoiceNo\_Total\_Count 的直方图

axes[1].hist(frequency['InvoiceNo\_Total\_Count'], bins=20, color='orange')

axes[1].set\_title('购入货物总数量直方图')

axes[1].set\_xlabel('货物总数量')

axes[1].set\_ylabel('频数')

# 绘制 InvoiceNo\_Category\_Count 和 InvoiceNo\_Total\_Count 的散点图

axes[2].scatter(frequency['InvoiceNo\_Category\_Count'], frequency['InvoiceNo\_Total\_Count'], color='green', alpha=0.5)

axes[2].set\_title('发票数量与货物总数量散点图')

axes[2].set\_xlabel('发票数量')

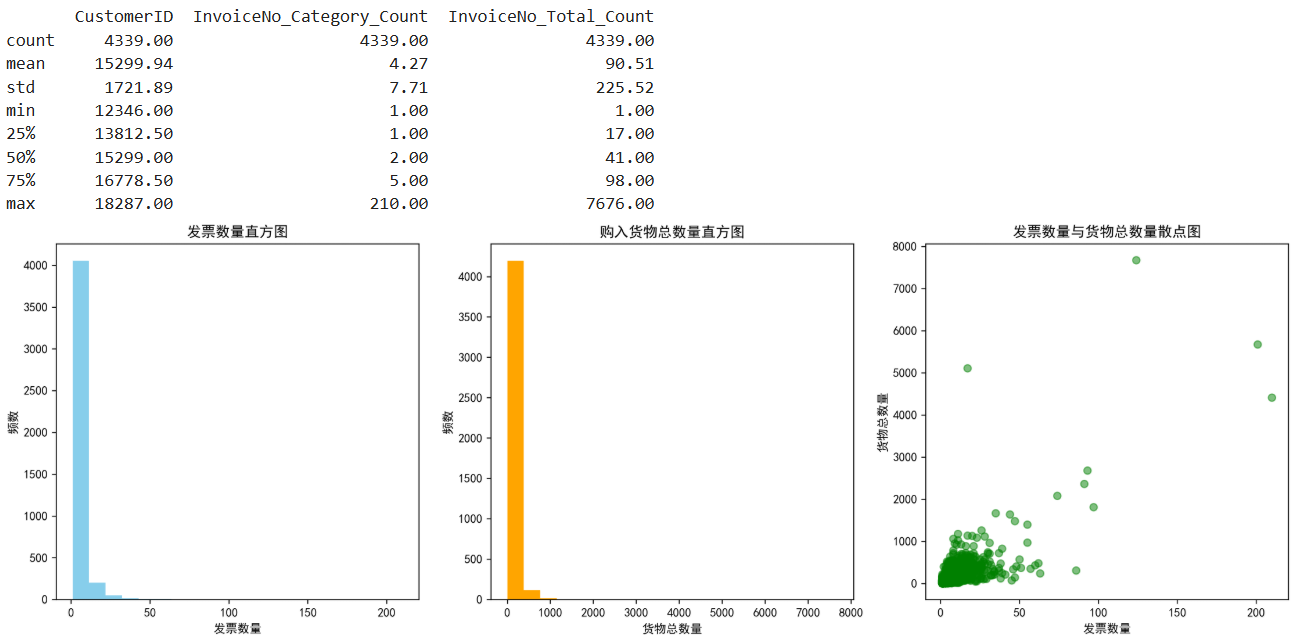
axes[2].set\_ylabel('货物总数量')

# 自动调整子图布局

plt.tight\_layout()

# 显示图形

plt.show()



1. 合并RFM，展示数据分布

鉴于F取值，这里取用InvoiceNo\_Category\_Count展示

rfm = recency.merge(frequency, on='CustomerID').merge(monetary, on='CustomerID')

columns = ['Recency', 'InvoiceNo\_Category\_Count', 'Monetary']

# 计算描述性统计信息，保留两位小数

result = df[columns].describe().round(2)

print('描述性统计信息：')

print(result)

# 创建一个包含 3 个子图的画布

fig, axes = plt.subplots(1, 3, figsize=(15, 5))

# 绘制直方图

for i, col in enumerate(columns):

axes[i].hist(df[col], bins=20, edgecolor='black')

axes[i].set\_title(col)

axes[i].set\_xlabel('值')

axes[i].set\_ylabel('频数')

plt.tight\_layout()

plt.show()

# 创建一个新的画布用于绘制三维散点图

fig = plt.figure(figsize=(11, 8))

ax = fig.add\_subplot(111, projection='3d')

# 绘制三维散点图

ax.scatter(df['Recency'], df['InvoiceNo\_Category\_Count'], df['Monetary'])

# 设置坐标轴标签和标题

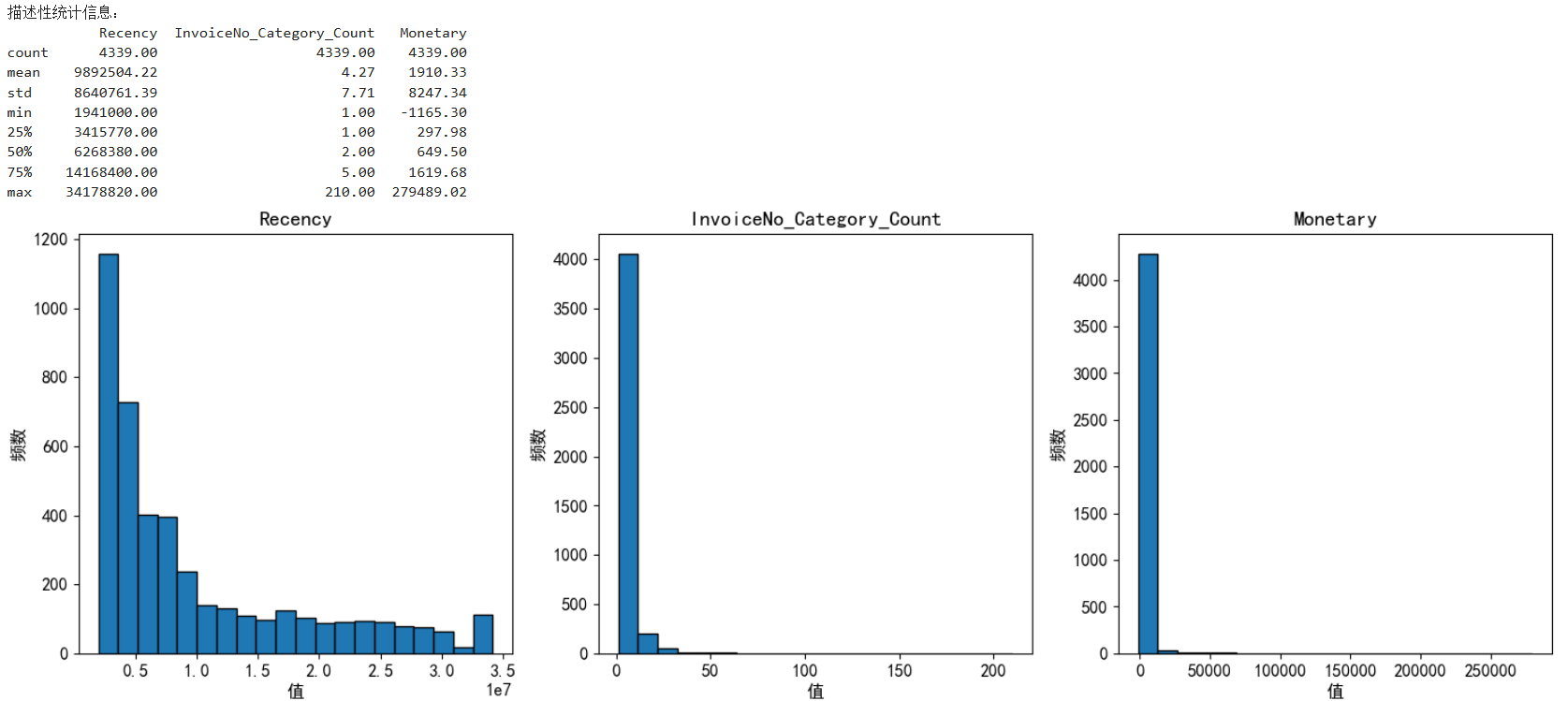
ax.set\_xlabel('Recency')

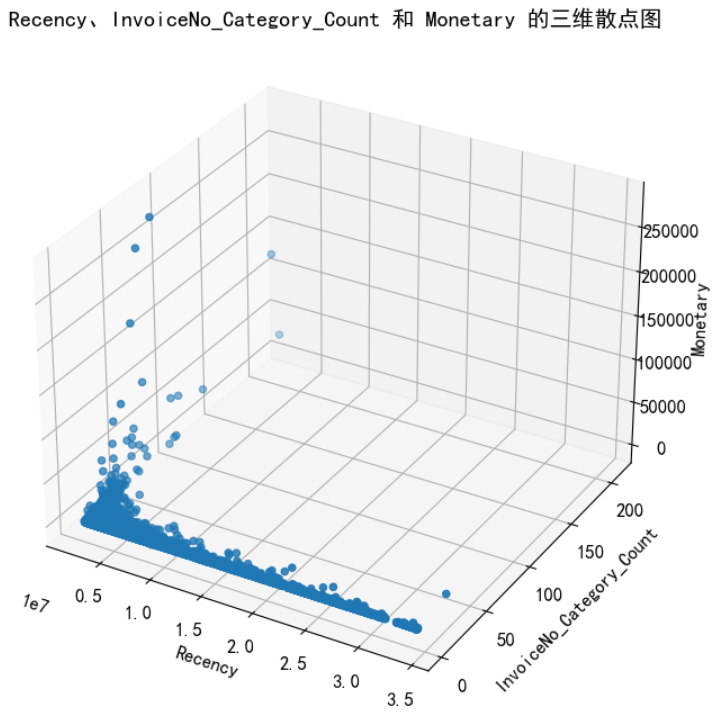
ax.set\_ylabel('InvoiceNo\_Category\_Count')

ax.set\_zlabel('Monetary')

ax.set\_title('Recency、InvoiceNo\_Category\_Count 和 Monetary 的三维散点图')

plt.show()





1. RFM的异常值探究

对数据进行z-score标准化处理，使用箱线图展示，使用四分位距法判断离群值

# 提取需要标准化的列

columns\_to\_standardize = ['Recency', 'Recency\_Days', 'InvoiceNo\_Category\_Count', 'InvoiceNo\_Total\_Count', 'Monetary']

data\_to\_standardize = df[columns\_to\_standardize]

# 标准化处理

scaler = StandardScaler()

scaled\_data = scaler.fit\_transform(data\_to\_standardize)

# 将标准化后的数据转换回 DataFrame

scaled\_df = pd.DataFrame(scaled\_data, columns=columns\_to\_standardize)

# 设置字体大小

plt.rcParams.update({'font.size': 12})

# 绘制箱线图检测离群值

scaled\_df.boxplot(figsize=(12,10))

plt.title('标准化数据的箱线图', fontsize=14)

plt.ylabel('标准化值', fontsize=13)

plt.xlabel('列名', fontsize=13)

plt.xticks(rotation=30)

plt.show()

# 查看是否存在离群值（以四分位距法为例）

Q1 = scaled\_df.quantile(0.25)

Q3 = scaled\_df.quantile(0.75)

IQR = Q3 - Q1

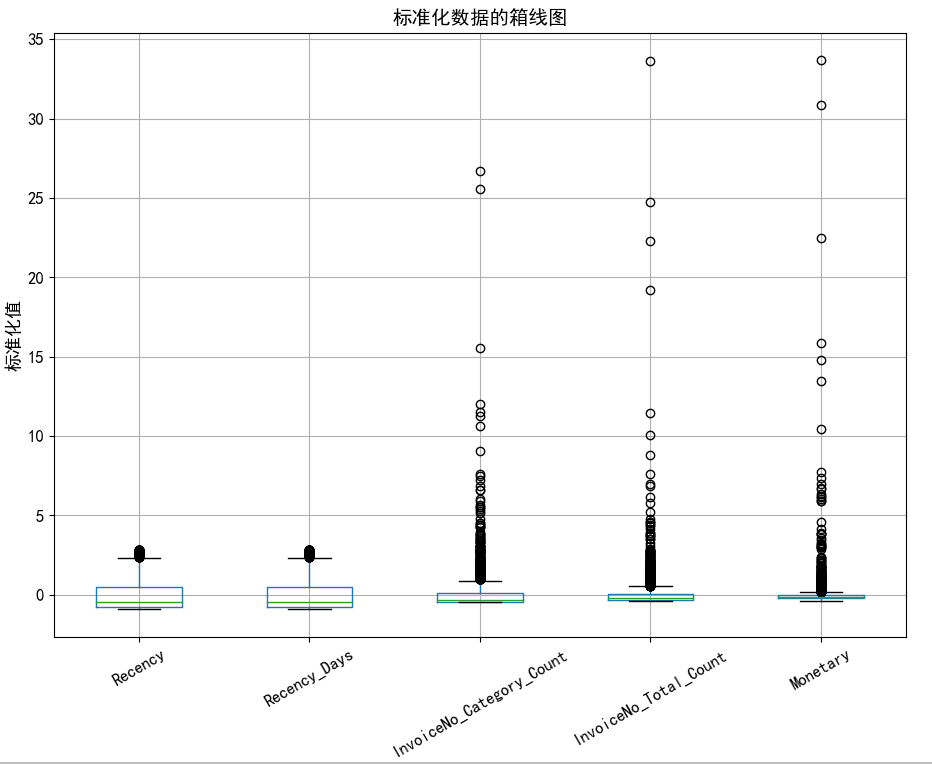
outliers = ((scaled\_df < (Q1 - 1.5 \* IQR)) | (scaled\_df > (Q3 + 1.5 \* IQR)))

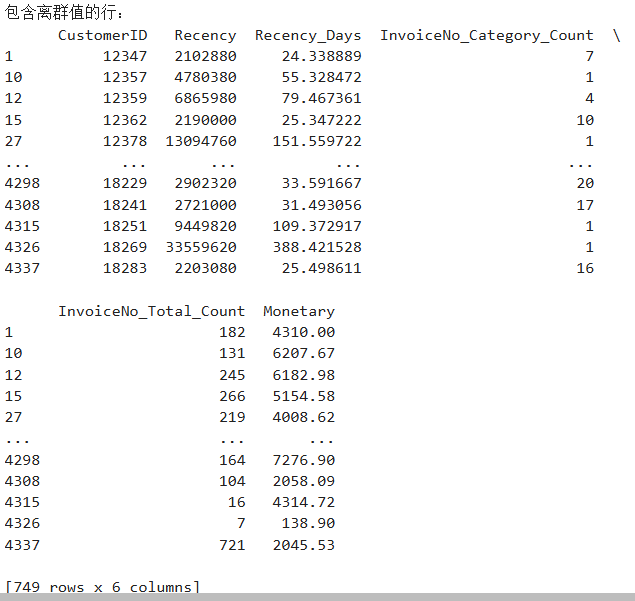
# 找出包含离群值的行

outlier\_rows = df[outliers.any(axis=1)]

print('包含离群值的行：')

print(outlier\_rows)





对数据进行z-score标准化处理，使用PCA对RFM进行姜维后使用基于DBSCAN的密度分析，探讨离群值

from sklearn.preprocessing import StandardScaler

from sklearn.cluster import DBSCAN

from sklearn.decomposition import PCA

# 提取需要分析的列

data = df[['Recency\_Days', 'InvoiceNo\_Category\_Count', 'Monetary']]

# 数据标准化

scaler = StandardScaler()

scaled\_data = scaler.fit\_transform(data)

# 使用 DBSCAN 检测异常值

# eps 和 min\_samples 需要根据数据特点调整

dbscan = DBSCAN(eps=0.5, min\_samples=5)

labels = dbscan.fit\_predict(scaled\_data)

# 标记异常值（标签为 -1 的是异常值）

is\_outlier = labels == -1

# 使用 PCA 进行降维以便可视化（将三维数据降到二维）

pca = PCA(n\_components=2)

reduced\_data = pca.fit\_transform(scaled\_data)

# 可视化结果

plt.figure(figsize=(10, 6))

plt.scatter(reduced\_data[~is\_outlier, 0], reduced\_data[~is\_outlier, 1], c='b', label='正常点')

plt.scatter(reduced\_data[is\_outlier, 0], reduced\_data[is\_outlier, 1], c='r', label='异常点')

plt.title('DBSCAN 异常值检测结果')

plt.xlabel('主成分1')

plt.ylabel('主成分2')

plt.legend()

plt.show()

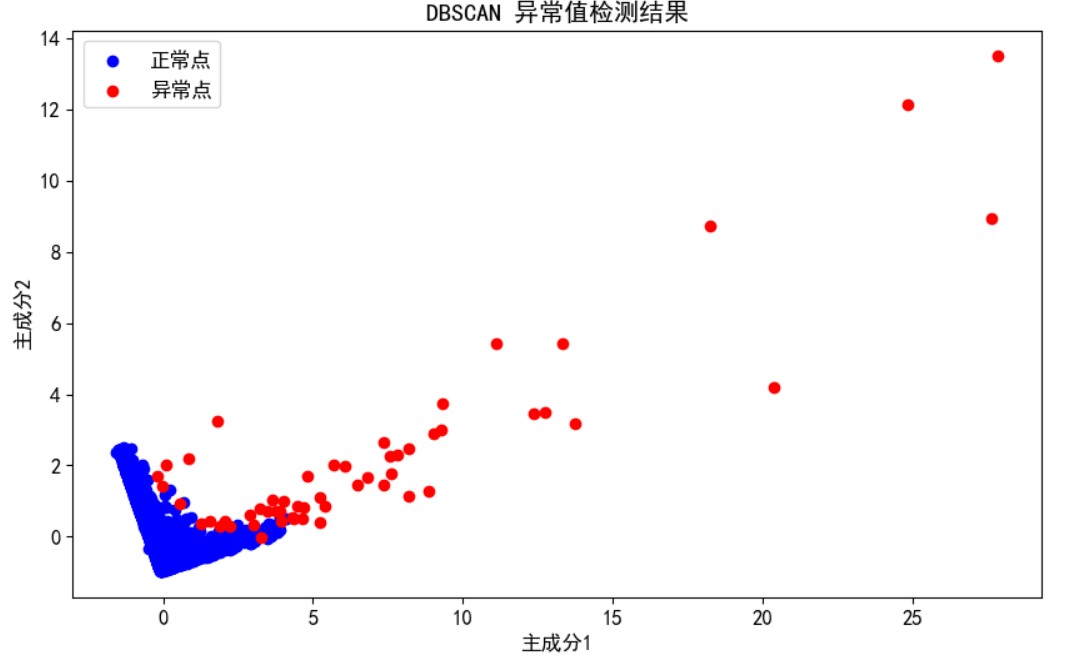
# 输出包含异常值的行

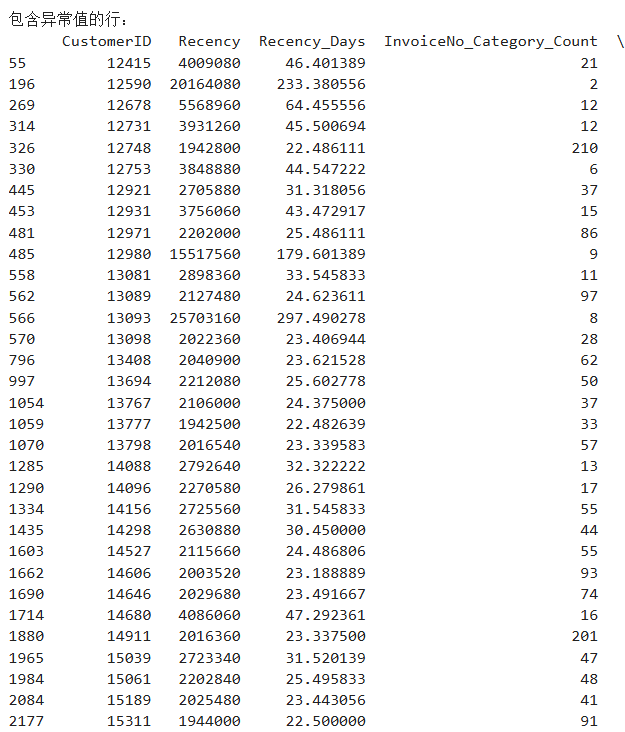
outlier\_rows = df[is\_outlier]

print('包含异常值的行：')

print(outlier\_rows)

print(len(outlier\_rows))





基于此探讨的离群值用于接下来的客户聚类，由于使用z\_score标准化，对于数据的离群值异常敏感，经过去除离群值后能使聚类效果更加明显，这里采用DBSCAN的离群值探究，在后续聚类中去除求得的异常值

1. **针对RFM建模**
2. 关于是否需要去除离群值使得聚类效果明显的探讨

以Kmeans聚类为例，首先使用未去除离群值的聚类效果，这里同时展示了使用肘部法及轮廓系数法确定最佳聚类簇数，这里确认为n=4较优

scaler = StandardScaler()

features = rfm[['Recency', 'InvoiceNo\_Category\_Count', 'Monetary']]

scaled\_features = scaler.fit\_transform(features)

scaled\_features\_df = pd.DataFrame(scaled\_features, columns=['Recency', 'InvoiceNo\_Category\_Count', 'Monetary'])

silhouette\_scores = []

for k in range(2, 11):

kmeans = KMeans(n\_clusters=k, random\_state=28)

kmeans.fit(scaled\_features)

labels = kmeans.labels\_

score = silhouette\_score(scaled\_features, labels)

silhouette\_scores.append(score)

plt.figure(figsize=(10, 5))

plt.plot(range(2, 11), silhouette\_scores, marker='x')

plt.xlabel('簇数')

plt.xticks(rotation=45)

plt.ylabel('轮廓系数')

plt.title('轮廓系数法确定最佳簇数')

plt.show()

sse = []

for k in range(2, 11):

kmeans = KMeans(n\_clusters=k, random\_state=28)

kmeans.fit(scaled\_features)

sse.append(kmeans.inertia\_)

# 绘制手肘图

plt.figure(figsize=(10, 5))

plt.plot(range(2, 11), sse, marker='x')

plt.xlabel('簇数')

plt.xticks(rotation=45)

plt.ylabel('SSE')

plt.title('肘部法确定最佳簇数')

plt.show()

clusters = 4

kmeans = KMeans(n\_clusters = clusters , random\_state=123)

kmeans.fit(scaled\_features\_df)

labels = kmeans.labels\_

scaled\_features\_df['Cluster\_Label'] = labels

rfm['kmeans\_cluster\_01'] = labels

# 绘制 3D 可视化图

plt.rcParams['figure.dpi'] = 100

fig = plt.figure(figsize=(12, 10))

ax = fig.add\_subplot(111, projection='3d')

colors = ['r', 'g', 'b', 'c', 'm', 'y', 'k', 'orange', 'purple']

ax.view\_init(elev=20, azim=30)

# 先绘制质心，增大标记大小和轮廓宽度

centroids = kmeans.cluster\_centers\_

for i, centroid in enumerate(centroids):

ax.scatter(centroid[0], centroid[1], centroid[2], c=colors[i], marker='\*', s=1200, edgecolor='black', linewidth=3)

# 绘制数据点，降低透明度

for i in range(clusters):

cluster = scaled\_features\_df[scaled\_features\_df['Cluster\_Label'] == i]

ax.scatter(cluster['Recency'], cluster['InvoiceNo\_Category\_Count'], cluster['Monetary'], c=colors[i],

label=f'Cluster {i}', alpha=0.2)

ax.set\_xlabel('Recency')

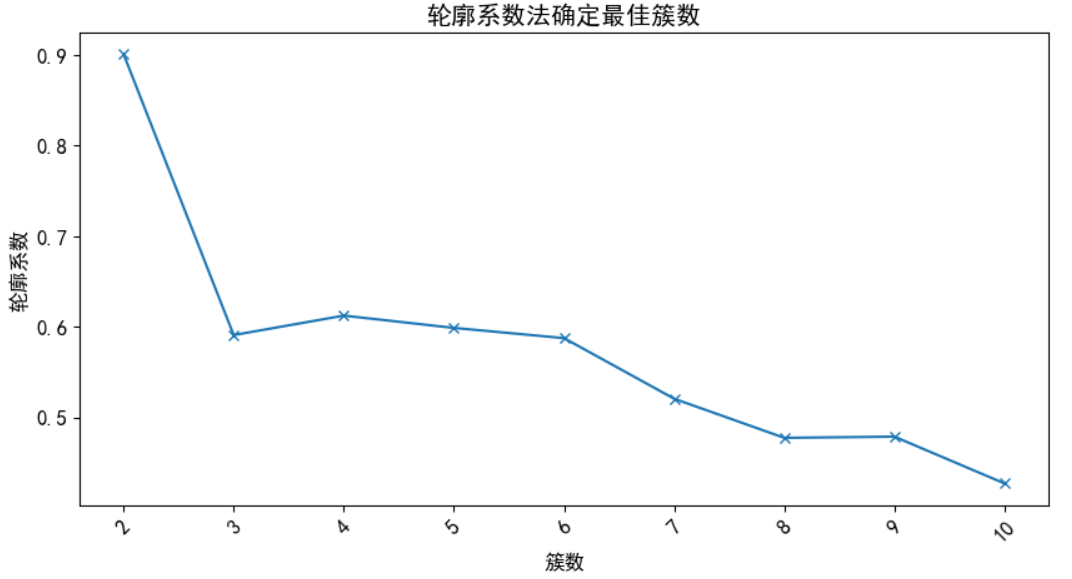
ax.set\_ylabel('InvoiceNo\_Category\_Count')

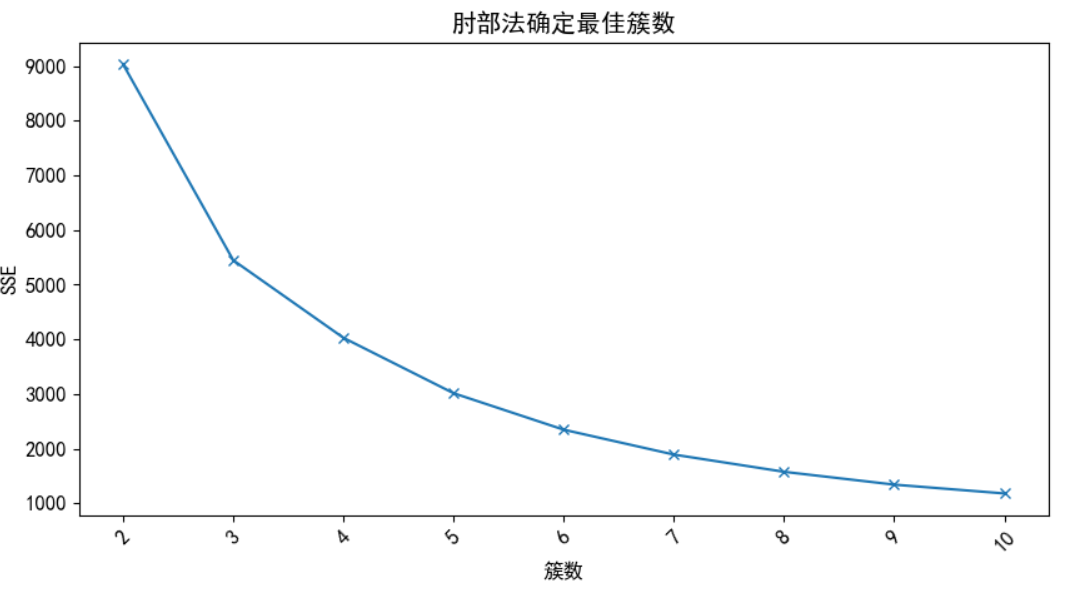
ax.set\_zlabel('Monetary')

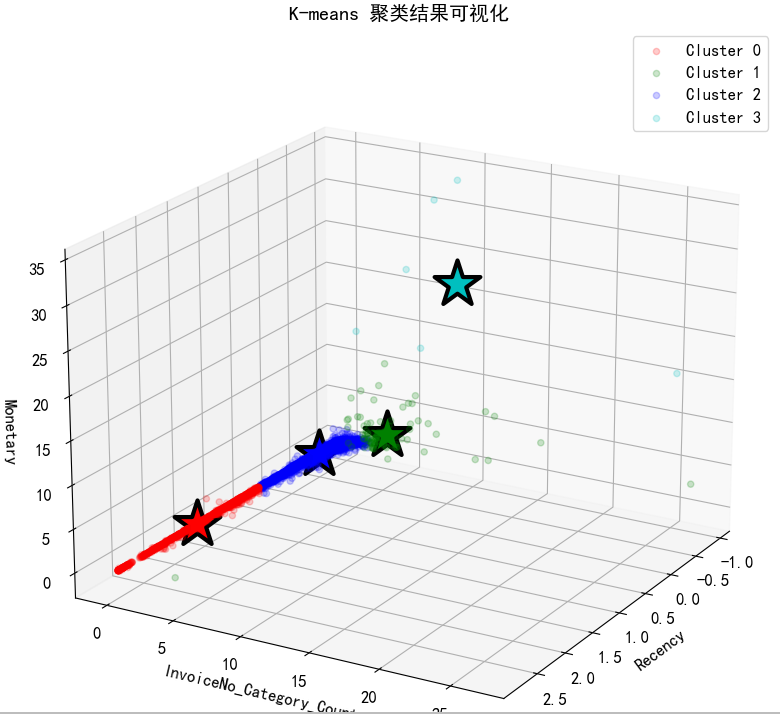
ax.set\_title('K-means 聚类结果可视化')

ax.legend()

plt.show()

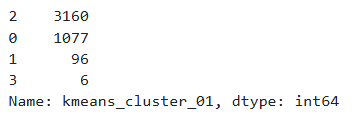




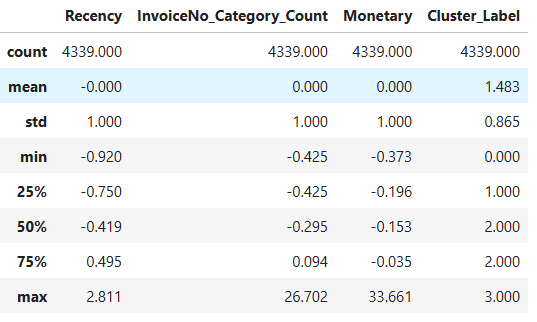


可见未去除离群值是数据呈现条状分布，只有少数的点属于散装分布，这里认为是标准化处理时离群值使得整体的均值以及标准差偏大，使得绝大部分处于同一取值范围的数据标准化后偏小，参考InvoicNo\_Category\_Count，整体在-3-3之间，为验证聚类结果这里输出聚类后的情况

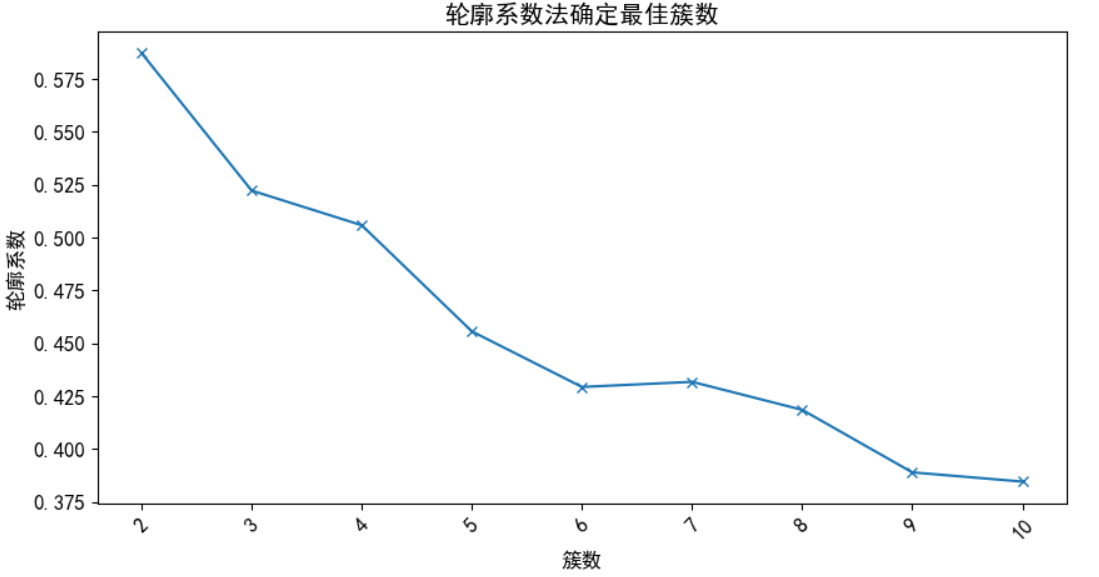
rfm['kmeans\_cluster\_01'].value\_counts()

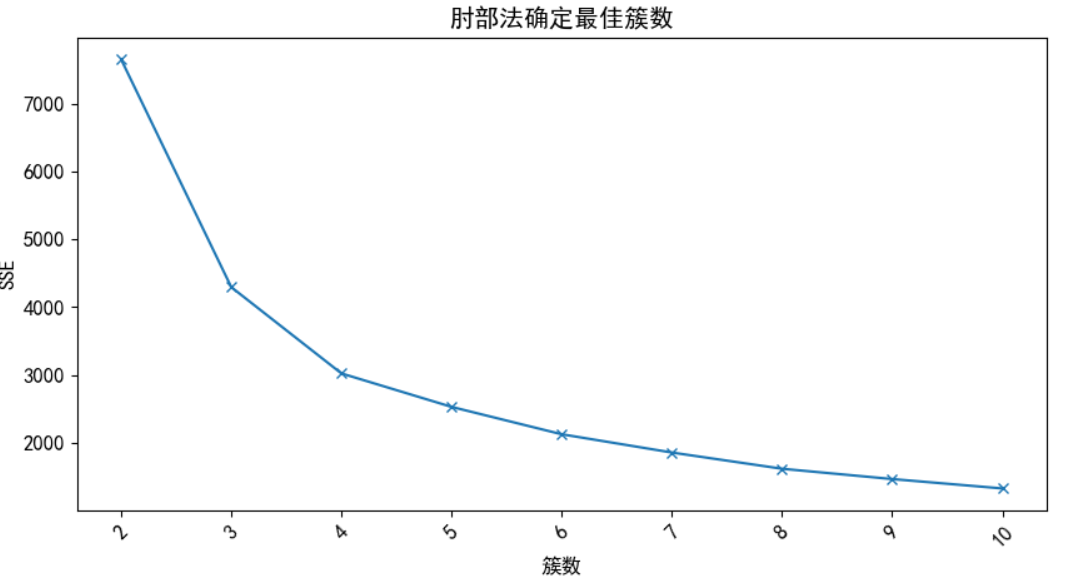


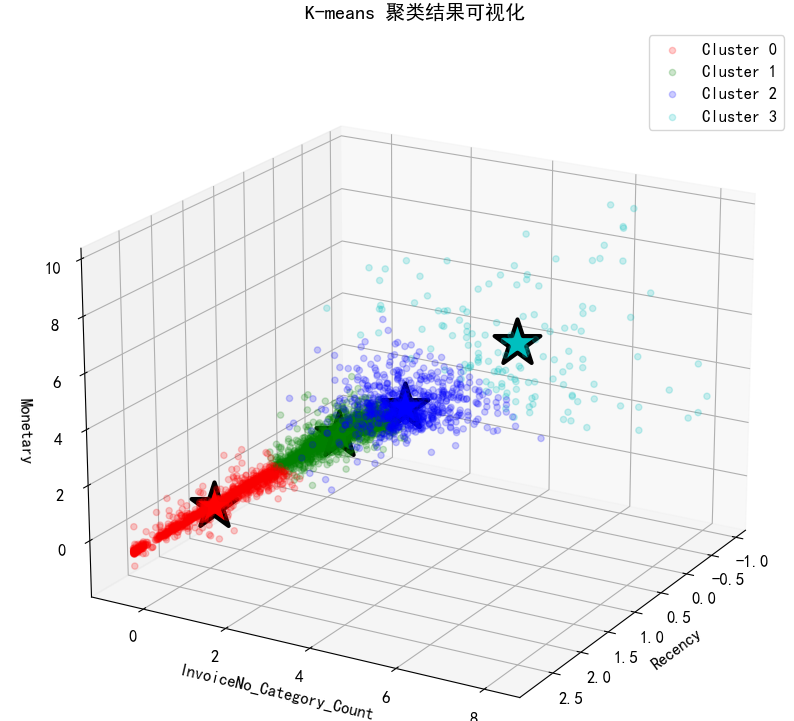
scaled\_features\_df.describe().round(3)

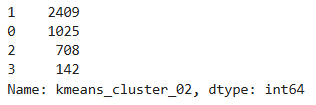


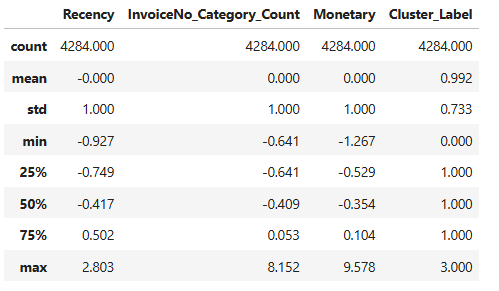
接下来使用去除由DBSCAN检测出的离群值的数据，代码同上











1. 确定cluster聚类簇数

原则上肘部法使用SSE最大的值，轮廓系数使用最大的值，结合两者来参考应选择参数偏大且符合实际需求的簇数，下文主要将客户分为4类，这里使用cluster=4

1. EM分布（混合高斯模型）聚类

使用AIC和BIC确定聚类簇数，绘制高斯分布可视化图，数据使用去除离群值后的

# 选取特征

X = df[['Recency', 'InvoiceNo\_Category\_Count', 'Monetary']]

# 定义要尝试的簇数范围

n\_components\_range = range(1, 11)

aic\_values = []

bic\_values = []

# 迭代不同的簇数

for n\_components in n\_components\_range:

gmm = GaussianMixture(n\_components=n\_components, random\_state=123)

gmm.fit(X)

aic = gmm.aic(X)

bic = gmm.bic(X)

aic\_values.append(aic)

bic\_values.append(bic)

# 绘制 AIC 和 BIC 曲线

plt.figure(figsize=(12, 6))

plt.plot(n\_components\_range, aic\_values, 'o-', label='AIC')

plt.plot(n\_components\_range, bic\_values, 's-', label='BIC')

plt.xlabel('Number of components')

plt.ylabel('Information criterion value')

plt.title('AIC and BIC for different number of components')

plt.legend()

plt.grid(True)

plt.show()

optimal\_n\_components = 4

# 使用最优簇数进行聚类

gmm\_optimal = GaussianMixture(n\_components=optimal\_n\_components, random\_state=123)

gmm\_optimal.fit(X)

df['GMM\_Cluster\_Label'] = gmm\_optimal.predict(X)

# 创建 3D 图形

fig = plt.figure(figsize=(10, 8))

ax = fig.add\_subplot(111, projection='3d')

# 调整视角，可根据实际情况修改

ax.view\_init(elev=20, azim=30)

# 先绘制聚类中心（均值），增大标记大小和轮廓宽度

means = gmm\_optimal.means\_

colors = plt.cm.rainbow(np.linspace(0, 1, optimal\_n\_components))

for i, mean in enumerate(means):

ax.scatter(mean[0], mean[1], mean[2], c=[colors[i]], marker='\*', s=1200, edgecolor='black', linewidth=3)

# 绘制散点图，不同类别用不同颜色表示，降低数据点透明度

for i in range(optimal\_n\_components):

subset = df[df['GMM\_Cluster\_Label'] == i]

ax.scatter(subset['Recency'], subset['InvoiceNo\_Category\_Count'], subset['Monetary'], c=[colors[i]],

label=f'Cluster {i}', alpha=0.2)

# 设置轴标签

ax.set\_xlabel('Recency')

ax.set\_ylabel('InvoiceNo\_Category\_Count')

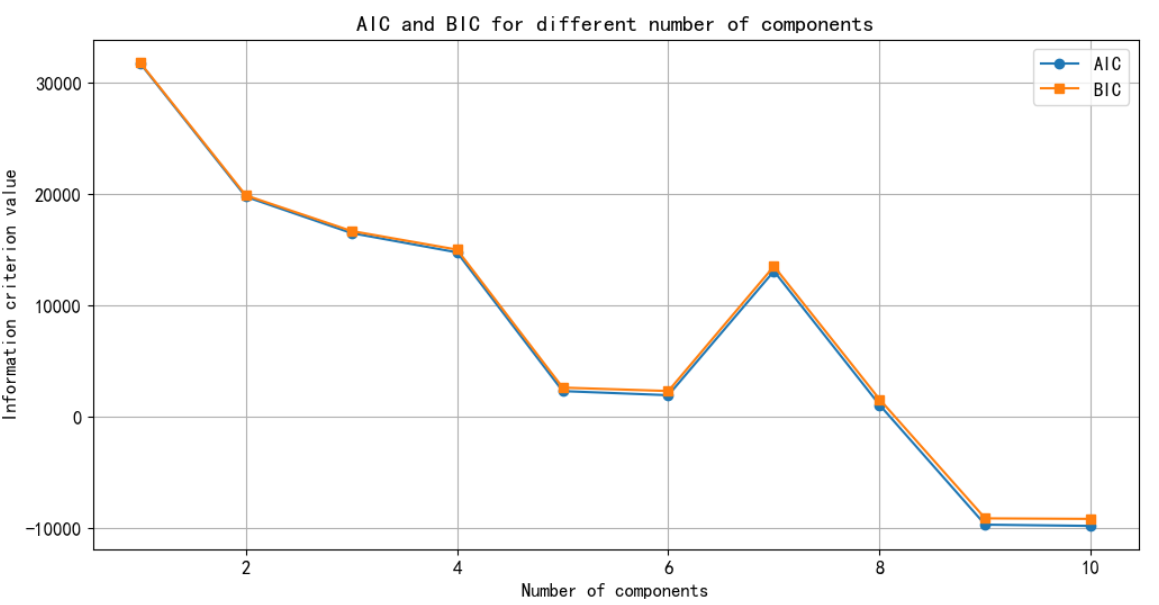
ax.set\_zlabel('Monetary')

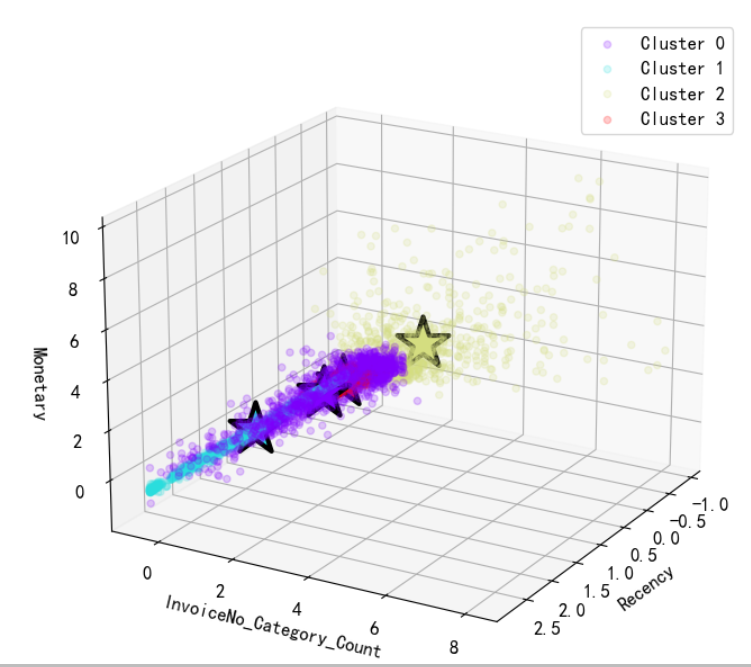
# 添加图例

ax.legend()

# 显示图形

plt.show()





1. 根据聚类标签对客户进行分类

首先使用Kmeans聚类结果，根据RFM客户分类模型定义，通常认为R小F大M大的结果为高价值用户，同理，根据聚类结果我们可以对相应类别打上标签，如下

rfm.loc[:, '客户类型'] = rfm['kmeans\_cluster\_02'].map({

0: '易流失客户',

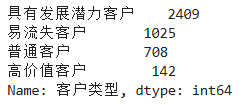
1: '具有发展潜力客户',

2: '普通客户',

3: '高价值客户'

})

rfm['客户类型'].value\_counts()



接着使用EM聚类结果

df.loc[:, '客户类型'] = df['GMM\_Cluster\_Label'].map({

0: '普通客户',

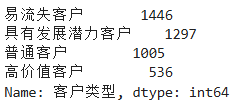
1: '易流失客户',

2: '高价值客户',

3: '具有发展潜力客户'

})

df['客户类型'].value\_counts()



**6、模型应用**

1）针对易流失客户

会员升级：为这类客户提供专属的会员升级通道，如降低升级门槛。例如，若企业常规会员升级需要在一定时间内消费达到一定金额，对于易流失客户，可将消费金额要求降低 20% - 30%，同时减少消费次数要求，以此鼓励他们增加消费，提升客户忠诚度，避免其流失。

积分兑换：设计高吸引力的积分兑换活动，提高积分的价值。比如，推出仅针对易流失客户的限时积分兑换商品，包括热门礼品或独家优惠券，这些商品的积分兑换所需积分可比普通兑换低 30% - 50%，刺激他们使用积分，增加与企业的互动。

交叉销售：根据客户之前的购买记录，精准推送相关产品。若客户购买过某类礼品，可向其推荐配套的礼品包装、贺卡等产品，并提供组合购买的优惠，如购买礼品和包装组合可享受 8 折优惠，增加客户的购买金额。

2）针对具有发展潜力客户

会员升级：设置具有挑战性但有吸引力的会员升级任务，如在规定时间内完成一定次数的不同品类产品购买，即可升级会员。升级后可享受优先配送、专属客服等特权，激励客户拓展消费品类，提升其在企业的消费层次。

积分兑换：建立积分成长体系，随着客户积分的增加，逐步解锁更高级别的积分兑换权益。例如，当积分达到一定阶段，可兑换更高价值的商品或参与专属抽奖活动，奖品包括高端礼品或大额购物券，引导客户持续消费以积累积分。

交叉销售：定期向客户推荐新品或高利润产品，并提供个性化的推荐理由。若客户经常购买某品牌的礼品，可推荐该品牌新推出的限量版礼品，同时给予首次购买的折扣优惠，如新品 9 折优惠，帮助客户发现更多心仪产品，提高客户的消费价值。

3）针对普通客户

会员升级：告知客户会员升级后的权益，如会员专享折扣、生日福利等，鼓励他们提升消费等级。可设定消费满一定金额即可升级会员的规则，如消费满 500 英镑升级为银卡会员，享受 9.5 折优惠，激发客户的升级欲望。

积分兑换：提供多样化的积分兑换选择，涵盖日常用品、小额优惠券等。例如，客户可用积分兑换满 100 减 20 的优惠券，或兑换一些实用的小礼品，增加客户对积分的认可和使用积极性。

交叉销售：在客户购买产品时，推荐一些性价比高的关联产品。如客户购买了一批节日礼品，可推荐节日装饰品，通过组合销售的方式，提高客单价。

4）针对高价值客户

会员升级：为高价值客户提供顶级会员服务，如专属的私人购物顾问、定制化产品推荐等。还可设立顶级会员俱乐部，举办高端社交活动或专属品鉴会，邀请高价值客户参加，增强他们的归属感和尊贵感。

积分兑换：设置高端、稀缺的积分兑换商品，如限量版奢侈品、豪华旅游套餐等，彰显高价值客户的特殊地位。这些兑换商品不仅能满足客户的高端需求，还能进一步提升他们对企业的忠诚度。

交叉销售：为高价值客户提供独家的高端产品或服务，如限量版礼品套装、定制化礼品服务等。针对他们的个性化需求，提供一对一的定制方案，收取合理的定制费用，进一步挖掘高价值客户的消费潜力。

**7、拓展思考（不用做）**

本实验只针对客户价值进行分析，但客户流失并没有提出具体的分析，对客户流失有兴趣的，自己查阅资料学习

1. **实验问题和体会**

数据处理复杂：原始数据集中存在多种问题

模型参数选择困难：在使用 K-means 聚类和 EM 聚类（基于高斯混合模型）时，确定合适的聚类簇数是一个关键问题。

结果解释：如何准确地将不同聚类与实际客户价值和营销策略联系起来具有一定难度