1. 目的

通过对信用卡客户行为与信用特征数据进行清洗，降维和聚类分析，从原始数据中提取有用信息，并使用聚类算法（K-Means）进行客户细分。

客户细分即根据客户的共同特征将客户划分为不同的群体，一方面能够帮助银行或金融机构评估客户的信用状况、还款能力和忠诚度等，进而进行风险管理和个性化服务，另一方面能够帮助电商公司有效地针对每个群体定制流量推送和营销策略。

2. 数据集介绍

数据集来源：本实验使用的数据集为 Kaggle 上名为“Customer Segmentation”的数据集（Customer Segmentation），包含有关客户的各种财务行为和账户信息。

数据集内容：数据表格中包含的属性有信用卡持有者 ID、月平均余额、过去 12 个月中有余额的比例、过去 12 个月的总消费金额、一次性购买的总金额、分期购买的总金额、总现金预借金额、消费频率、一次性购买的频率、分期购买的频率、现金预借频率、每次现金预借交易的平均金额、每次消费交易的平均金额、信用额度、总付款金额、该期间到期的最低付款总额、全额支付到期对账单余额的月份百分比、作为客户的月份数。

数据维度：数据集包含 8951 行样本和 18 列特征，表格的源文件在压缩包中呈现。

3. 数据预处理

数据预处理是进行有效分析和建模的关键步骤。

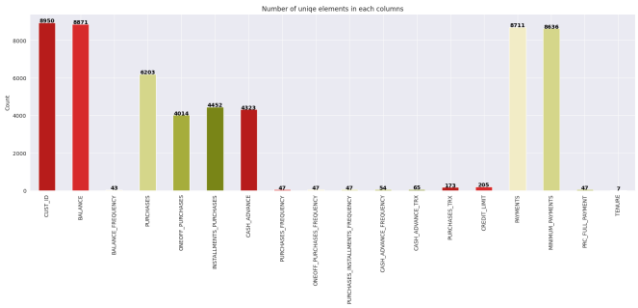
3.1 统计原数据集

首先，在数据预处理之前，查看数据的维度，输出各个特征的基本统计信息。

输出结果如下：

```
Dimensions of dataset: (8950, 18)    Rows: 8950    Columns: 18
```

然后，对数据进行统计，包括均值、标准差、最小值、最大值等，以便了解数据的分布情况，再使用条形图可视化，可视化结果如下：



图表 1：原数据可视化

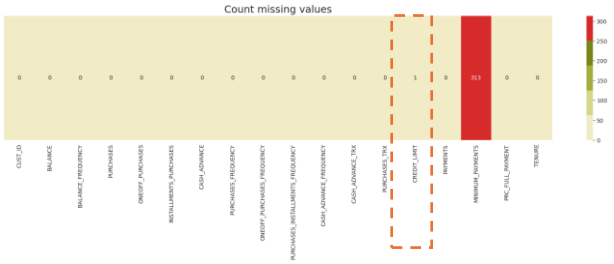
最后，检查数据的类型，以便接下来的代码编写。数据类型的输出如下：

```
The data type contains:    object --> 1    int64 --> 3    float64 --> 14
```

3.2 数据清洗

3.2.1 检查缺失值：

使用热图展示每列每行的缺失值数量。



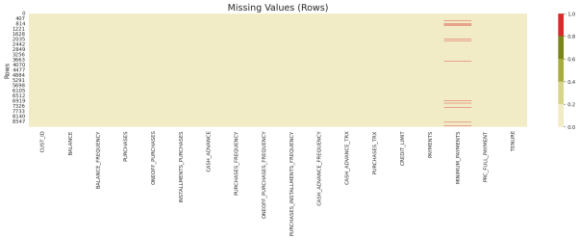
图表 2：列向的缺失值数量统计

3.2.2 处理缺失值

由以上可视化的结果我们知道，Credit_Limit 列和 Minimum_payments 列存在缺失值，先对于 Credit_Limit，我们采用以下操作来删除缺失值，并填充为 Ture。需要注意要修改则需要用到“reset”。

用 null_cl = df[df['CREDIT_LIMIT'].isnull()]检查数据框 df 中“CREDIT_LIMIT”列的空值。isnull()函数用于识别空值，然后根据这个序列筛选出包含空值的行，存储在 null_cl 变量中。使用 df = df.dropna(subset=['CREDIT_LIMIT']) 来删除“CREDIT_LIMIT”列中包含空值的行。用 df = df.reset_index(drop=True)重置数据框 df 的索引，drop=True 参数表示不将旧索引添加为数据框的列。

接下来我们对 Minimum_payments 做操作，先再次检查一下最低还款这一列的空值并画出图像如下，同时展示缺失值所在行。

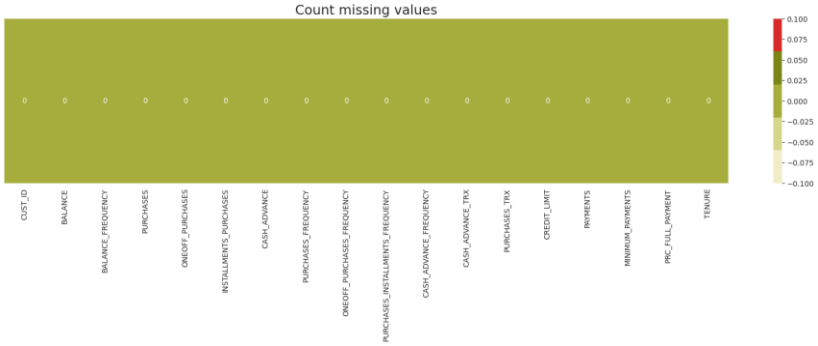


图表 3：包含行的缺失值数量统计

接着，用中位数填充缺失值。

```
# 用中位数填充'MINIMUM_PAYMENTS'列中的缺失值
df['MINIMUM_PAYMENTS'].fillna(df['MINIMUM_PAYMENTS'].median(), inplace=True)
# Print the shape of the DataFrame
print(df.shape)
```

最后再次输出图像，检查缺失值的填充是否圆满完成。



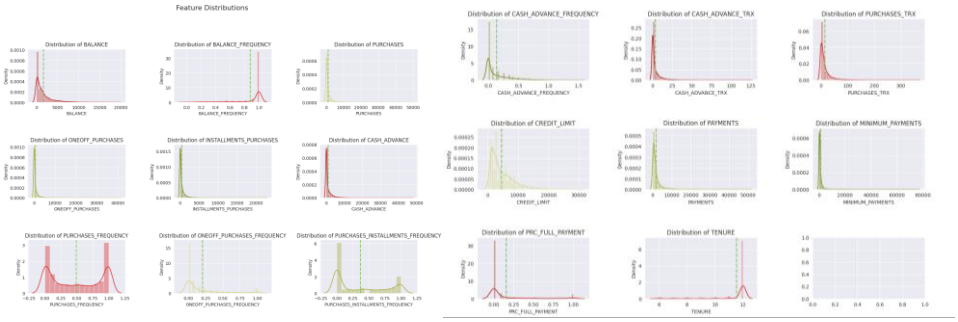
图表 4：处理缺失值后的可视化检查

由上图可知，缺失值已处理完毕。

3.3 离群值处理

3.3.1 监测离群值

首先，将每个特征的分布可视化，初步观察是否有极端值。



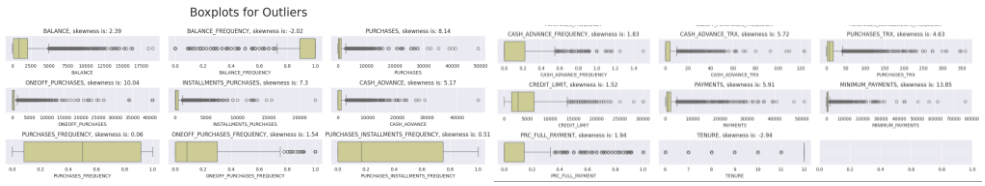
图表 5：特征分布图

由上图可知，特征基本为多峰分布，分布跨度较小。对于这个分布模式，我们可以在接下来采用箱线图进一步进行检测。

```
def boxplots_custom(dataset, columns_list, rows, cols, suptitle):
    fig, axs = plt.subplots(rows, cols, sharey=True, figsize=(15, 18))
    fig.suptitle(suptitle, y=1, size=25)
    axs = axs.flatten()
    for i, data in enumerate(columns_list):
        sns.boxplot(data=dataset[data], orient='h', color='#D5D6BA', ax=axs[i])
        axs[i].set_title(data + ', skewness is: ' + str(round(dataset[data].skew(axis=0, skipna=True), 2)))

boxplots_custom(dataset=df, columns_list=df.columns, rows=6, cols=3, suptitle='Boxplots for Outliers')
plt.tight_layout()
plt.show()
```

绘制箱线图如下，展示数据的四分位数范围、上下须范围和潜在的离群值。



图表 6：离群值箱线图

上图可知，BALANCE，CASH_ADVANCE 存在明显的高值离群点，PURCHASES 有较多高值异常点，MINIMUM_PAYMENTS 离群值非常多。MINIMUM_PAYMENTS 和 ONEOFF_PURCHASES 有较高正偏特征，BALANCE_FREQUENCY 和 TENURE 有负偏特征，PURCHASES_FREQUENCY 接近对称。

接下来使用 IQR 和 z-score 方法来更深入识别具体异常值，并统计每个特征的异常值数量。

Q1 是下四分位数，Q3 是上四分位数。

将上下界定义为 上界 $= Q3 + 1.5 \times IQR$ ； 下界 $= Q1 - 1.5 \times IQR$（IQR=Q3-Q1），数据值小于下界或大于上界的点是离群值。

```
Q1 = df.quantile(0.25)
Q3 = df.quantile(0.75)
IQR = Q3 - Q1
# 使用IQR方法识别异常值
outliers = ((df < (Q1 - 1.5 * IQR)) | (df > (Q3 + 1.5 * IQR)))
# 计算每个变量的异常值数量
num_outliers = outliers.sum()
```

计算，判断离群值，并将它们统计起来。

接着使用标准分数 z-score 表示每个数据点与其均值的偏离程度。z=（样本点-特征均值）/特征标准差。我们定义如果一个点的 z-score > 3，说明它偏离均值超过 3 倍标准差，是离群值。

最终由 z-score 计算得出的离群值有 1741 个。对于这些离群值，我们需要做进一步的处理。

3.3.2 替换离群值

如果值大于上须值（Q3 + 1.5 * IQR），则将其替换为上须值；如果值小于下须值（Q1 - 1.5 * IQR），则将其替换为下须值。

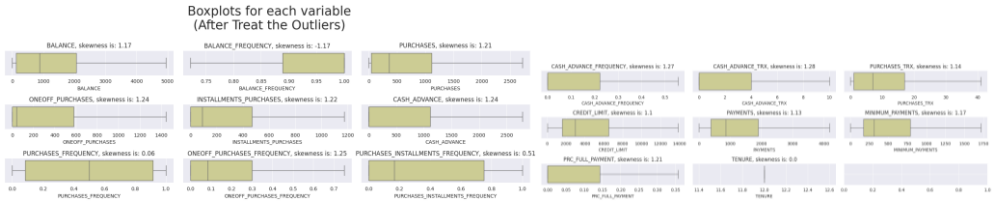
```
Outliers_Treatment = df.copy()
# 计算数据集中每列的IQR
Q1 = Outliers_Treatment.quantile(0.25)
Q3 = Outliers_Treatment.quantile(0.75)
IQR = Q3 - Q1
# 计算每列的上、下限值
whisker_width = 1.5
lower_whisker = Q1 - (whisker_width * IQR)
upper_whisker = Q3 + (whisker_width * IQR)
# 用每列的上部或下部品须值替换异常值
for col in Outliers_Treatment.columns:
    Outliers_Treatment[col] = np.where(Outliers_Treatment[col] > upper_whisker[col], upper_whisker[col],
                                       np.where(Outliers_Treatment[col] < lower_whisker[col], lower_whisker[col],
                                                Outliers_Treatment[col]))
```

完成这一步后，重新使用 IQR 方法检查数据当中是否还有异常值，并输出检查。

异常值数量 (每列):

	BALANCE	BALANCE_FREQUENCY	...	PRC_FULL_PAYMENT	TENURE
Outlier Count	0	0	...	0	0

再进一步输出箱线图结果：



图表 7：离群值处理后的箱线图结果

根据以上所有的输出结果我们可以知道所统计的数据已无异常值。

3.4 探索性数据分析

接下来通过单变量和多变量分析来深入了解数据的分布、特征间的关系以及潜在的模式。

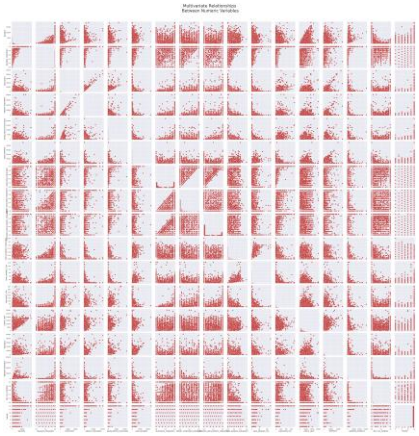
3.4.1 单变量分析

查看每个变量的分布、统计信息情况，再次检查统计值分布，将均值、中位数、最小值和最大值直接标注在图中。

3.4.2 多变量分析

首先我们绘制成对变量的散点图，查看特征之间的潜在关系和分布模式。

根据散点图，我们看出 PURCHASES 和 ONEOFF_PURCHASES 之间的关系较强，散点图显示出线性趋势。CASH_ADVANC 和 CASH_ADVANCE_TRX 也表现出相关性，散点密集呈现出正相关特性。

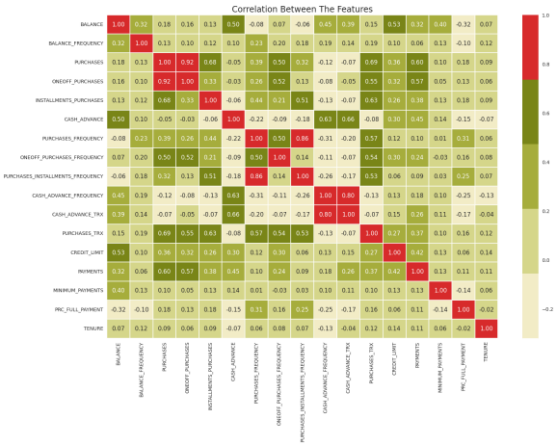


图表 8：变量散点图

接着，我们绘制相关性热图，通过热图，直观了解变量之间的相关性系数，做出更

精准的判断。

根据相关性热图，我们看到 PURCHASES 和 ONEOFF_PURCHASES 之间的相关系数为 0.92，说明一次性消费在总体消费中占有很大比重。PURCHASES_INSTALLMENTS_FREQUENCY 和 PURCHASES_FREQUENCY 的相关系数为 **0.86**，说明分期付款频率和总购买频率之间高度相关。CASH_ADVANCE 和 CASH_ADVANCE_TRX 的相关系数为 **0.66**，表明现金预支金额和相关交易次数密切相关。



图表 9：变量相关性热图

BALANCE_FREQUENCY 和大多数其他变量相关性较弱，例如与 PURCHASES 的相关系数为 0.18。PRC_FULL_PAYMENT 和其他变量的相关性普遍较低。

PURCHASES 和 ONEOFF_PURCHASES 的高度相关性可能导致多重共线性问题，在后续我们将会注意这方面问题。

3.5 存储数据

最后，我们将处理后的数据存储为 CSV 文件，命名为 processed_data

4. 数据降维：主成分分析（PCA）

下一步，我们通过主成分分析（PCA）减少数据维度，去除冗余信息，同时保留尽可能多的解释方差。

首先，我们先尝试保留 5 个主成分，根据原始数据拟合出主成分的方向。调用 sklearn 中的 PCA api，，指定`n_components=5`。

接着将原始数据投影到这 5 个主成分方向上，生成降维后的新数据。使用`fit_transform`方法对原始数据`df`进行 PCA 变换，得到降维后的数据存储在`df_pca`中。

然后输出每个主成分保留的数据方差占原始数据总方差的比例，这一步是为了分析每个主成分的重要性，决定是否需要进一步减少主成分的数量。

```
pca = PCA(n_components=5)  # 5个
df_pca = pca.fit_transform(df)
# 输出每个主成分的解释方差比例
print("Explained variance ratio:", pca.explained_variance_ratio_)
# 累计解释的方差比例
cumulative_variance = np.cumsum(pca.explained_variance_ratio_)
print("Cumulative explained variance:", cumulative_variance)
# 绘制解释方差的累计曲线
plt.figure(figsize=(8, 6))
plt.plot( *args: range(1, len(cumulative_variance) + 1), cumulative_variance, marker='o', linestyle='--')
plt.title("Cumulative Explained Variance by Principal Components")
plt.xlabel("Number of Principal Components")
plt.ylabel("Cumulative Explained Variance")
plt.grid(True)
plt.show()
```

由输出的结果我们可以知道，第一主成分：保留了 47.54% 的原始数据的信息，前两个主成分：累计保留了 65.31% 的信息，前五个主成分：累计保留了 95.47% 的信息。因此，我们可以只使用前 4-5 个主成分进行后续的分析，而不会显著丢失信息。

Principal Component 1	Principal Component 2	Principal Component 3	Principal Component 4	Principal Component 5

-4324.351008	915.734333	201.122729	-152.782832	-56.070543
4120.975546	-2437.938847	2361.236482	-3923.450851	-52.137671
1498.994555	-2001.750670	-2109.071892	1052.164413	303.906631
1321.563972	-1387.728898	-2765.268751	1263.158294	481.252569
-3741.083696	751.701480	529.244726	-227.577068	227.498468

图表 10：降维后的数据

5. 聚类分析：K-Means

使用 K-Means 聚类算法对降维后的数据进行客户细分。

5.1 确定最佳聚类数

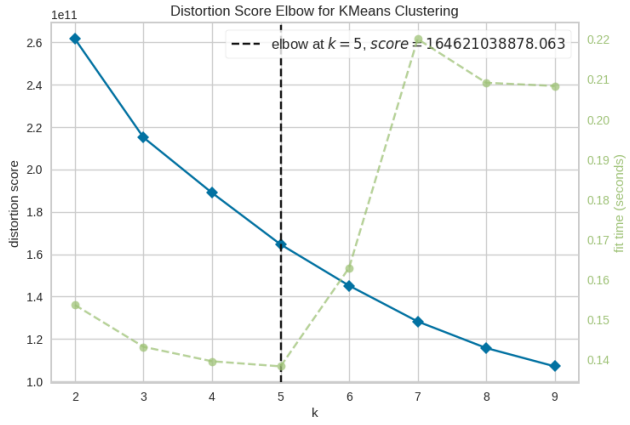
首先通过肘部法则（Elbow Method）选择最佳聚类数：先创建一个 KElbowVisualizer 对象，取 k 值的范围为 2 到 10，使用扭曲度作为评估聚类效果的指标。

先创建一个对象叫 KElbowVisualizer，传入了已经训练好的 K-Means 模型 kmeans_model。设定参数 k=(2, 10)指定要评估的聚类数范围，即从 2 到 10。用参数 metric='distortion'作为评估指标，失真度是聚类内所有点到其聚类中心的距离的总和，失真度越低，表示聚类效果越好。参数 timings=True 表示在可视化中显示每个 k 值的计算时间。接着，elbow_visualizer.fit(df_pca_df)将 PCA 降维后的数据 df_pca_df 拟合到肘部可视化对象中，计算不同 k 值下的失真度。

```
elbow_visualizer = KElbowVisualizer(kmeans_model, k=(2, 10), metric='distortion', timings=True)
elbow_visualizer.fit(df_pca_df)
elbow_visualizer.show()
```

查看肘部法则的结果，确定最佳的聚类数。画出以下图形，横轴表示聚类的数目，从 2 到 9，左侧纵轴表示聚类的扭曲度，即所有点到其最近聚类中心的距离的总和。扭曲度越低，表示聚类效果越好，因为数据点更接近其聚类中心。

我们要寻找扭曲度下降速度明显减缓的点，这个点就是肘部点。在这个点之前，增加聚类数目会显著提高聚类效果；而在这个点之后，增加聚类数目对聚类效果的提升变得不那么显著，但计算成本会增加。



图表 11：肘部法则可视化

观察图像可知，最佳聚类数为 5。

5.2 聚类模型与可视化

我们首先使用通过肘部法则确定的最佳聚类数 5 以及之前设置好的 kmeans 参数创建一个新的 KMeans 模型。然后使用 PCA 降维后的数据来拟合这个模型，并预测每个数据点的聚类标签。

```
# 使用最佳聚类数重新拟合 KMeans 模型
final_kmeans = KMeans(n_clusters=optimal_clusters, **kmeans_params)
df_pca_df['Cluster'] = final_kmeans.fit_predict(df_pca_df)
```

5.3 模型的评估

接着，使用 Silhouette 和 Calinski-Harabasz 来计算系数。

轮廓系数是无监督学习中用于评估聚类质量的度量指标，它衡量一个对象与其自身聚类的相似度与其他聚类相比的情况。轮廓系数的值介于-1 到 1 之间，值为 1 表示对象与其聚类非常匹配，与邻近聚类匹配度低；值为-1 则表示对象与其聚类匹配度低，与邻近聚类匹配度高。接近 0 的值意味着对象位于两个聚类的边界附近。

卡林斯基-哈拉巴斯系数是无监督学习中用于评估聚类质量的另一个度量指标，它衡量聚类间方差与聚类内方差的比率。较高的卡林斯基-哈拉巴斯系数表明聚类之间更加分离和不同。这意味着聚类模型生成了区分度高且独特的聚类，且对象被正确分配到了各自的聚类中。

silhouette_score 函数从 sklearn.metrics 模块调用，用于计算轮廓系数。df_pca_df 包含了主成分分析后的数据。计算轮廓数据之前要用 drop(labels='Cluster', axis=1)从数据框中删除名为'Cluster'的列，因为轮廓系数的计算不需要聚类标签。用 silhouette_avg 表示所有样本轮廓系数的平均值，保留四位小数。

接着计算 Calinski-Harabasz 分数，calinski_harabasz_score 函数同样来自 sklearn.metrics 模块，计算方法与轮廓系数类似，只是这里直接使用了包含聚类标签的数据框，因为 Calinski-Harabasz 分数的计算需要这些标签来确定聚类间的距离和聚类内的距离。calinski_harabasz 表示计算得到的 Calinski-Harabasz 分数，也是保留四位小数。

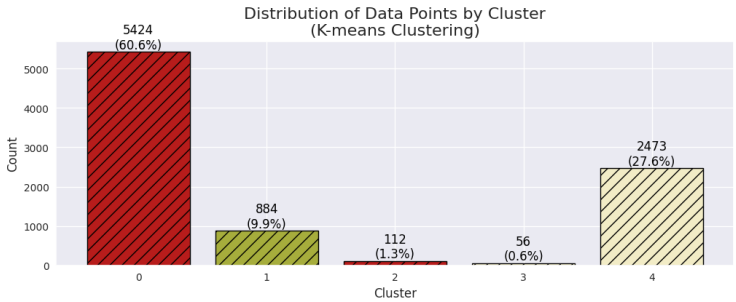
```
# 计算轮廓系数（Silhouette Score）
silhouette_avg = silhouette_score(df_pca_df.drop(labels='Cluster', axis=1), df_pca_df['Cluster'])
print(f"Silhouette Score: {silhouette_avg:.4f}")  # 输出轮廓系数

# 计算Calinski-Harabasz分数
calinski_harabasz = calinski_harabasz_score(df_pca_df.drop(labels='Cluster', axis=1), df_pca_df['Cluster'])
print(f"Calinski-Harabasz Score: {calinski_harabasz:.4f}")  # 输出Calinski-Harabasz分数
```

轮廓系数和卡林斯基-哈拉巴斯系数输出结果如下：

Silhouette Score: 0.5177

Calinski-Harabasz Score: 2862.2336

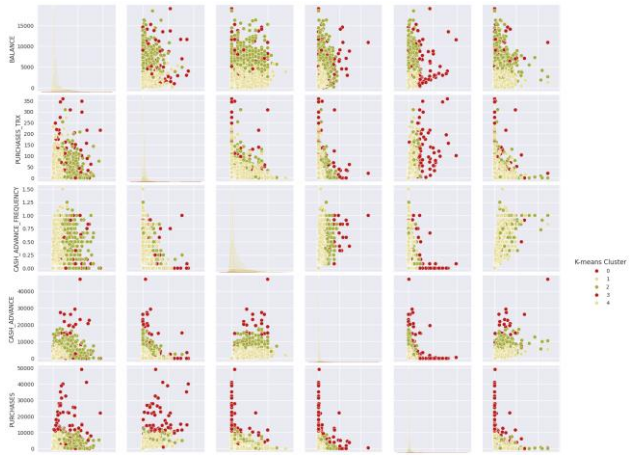


图表 10：聚类标签的分布

轮廓系数 0.5177 表示聚类效果中等。CH 分数 2862.2336 表示簇间分离度较好，但是否是高分需要结合数据规模评估。二者结合来看，聚类效果还可以，但可能存在一些边界样本的模糊分类。

6. 聚类结果分析

可视化特征与聚类结果的关系如下：



图表 11：特征与聚类散点图

余额方面，群体 2 和 群体 3 的客户余额较高，集中在 4000 至 8000 范围内。群体 4 和群体 1 则显示余额普遍较低，表明这些客户可能是低消费群体或信用卡使用较少的用户。

消费交易次数方面，群体 3 明显高于其他群体，消费交易次数集中在 80 次以上，表明这一群体是高频消费客户。群体 0 和 群体 1 的消费交易次数则较为均匀分布在低频范围，可能代表偶尔使用信用卡进行消费的客户。

现金预支频率方面，群体 2 和 群体 3 展现出较高的现金预支频率，尤其是 群体 2，频率明显高于其他群体，可能代表偏爱使用现金预支功能的客户。群体 0 和 群体 4 在现金预支频率上非常低，表明这两类客户可能更倾向于直接消费，而非使用信用卡获取现金。

现金预支总额方面，群体 3 的现金预支总额显著高于其他群体，表明这一群体可能依赖信用卡预支较多。群体 4 和 群体 1 的现金预支几乎为零，进一步表明他们对这一功能需求较少。

总消费金额方面，群体 3 的总消费金额突出，高消费群体特征明显。群体 0 和 群体 1 的消费金额则更均匀分布，表明他们可能消费行为较为保守。

此外，按 K-means 分组，计算特质的均值，然后使用柱状图展示不同簇中各特征的均值，识别每簇的特性。

图表 12：聚类特征均值分析

群体 3 显示出最高的总消费金额、交易次数和现金预支总额，表明这一群体是活跃且高消费的客户。群体 2 在现金预支频率和交易次上表现突出，可能是偏好使用信用卡预支现金的客户。群体 0 和 群体 1 在所有特征上均表现中等或较低，属于消费行为普通或低活跃客户。群体 4 总体数值最低，可能代表低消费、低交易频率的保守型用户。

按聚类标签和租期对数据分组，统计每个租期的客户数量，分析每个聚类中客户租期的分布。输出的表格如下：

K-means Cluster	0		1							2						
TENURE	11	12	6	7	8	9	10	11	12	6	7	8	9	10	11	12
TENURE	5	51	25	18	24	24	47	92	2251	15	7	16	15	20	34	808

K-means Cluster	3					4						
TENURE	6	7	10	11	12	6	7	8	9	10	11	12
TENURE	1	1	1	3	106	162	164	156	136	168	231	4368

表格展示了不同聚类中成为客户的天数的分布情况，反映了客户使用信用卡的时长特点。客户天数的分布可以帮助识别客户忠诚度和活跃度的差异。群体 4 中客户天数为 12 个月的客户占比最大，达到了 4368 人，说明该群体稳定性高。而 群体 0 和 群体 3 的客户数量相对较少，且分布较为分散。

结论或体会

结论：

群体 0: 均衡型中度活跃客户

1. 特征表现
    1) 在账户余额和消费金额等指标上，群体 0 客户处于中间水平，既不偏低也不特别高。
    2) 这些客户现金预支金额现金交易频率也处于适中水平，表明他们偶尔使用信用卡的现金功能。
    3) 购买交易次数和消费频率显示出他们有一定的日常信用卡消费习惯。
2. 总结

群体 0 的客户是相对均衡的消费群体，既不完全依赖现金交易，也没有特别高频的信用卡使用。他们可能是普通的信用卡用户，主要将信用卡作为日常消费的一部分。

群体 1: 低消费/高全额还款客户

1. 特征表现
    1) 群体 1 客户的账户余额和消费金额处于最低水平，表明他们消费金额较低。
    2) 然而，他们在全额还款比例上的表现较为突出，可能说明他们倾向于一次性还清账单。
    3) 现金预支和购买频率也较低，这表明他们较少使用信用卡的增值功能。
2. 总结

群体 1 的客户使用信用卡的频率和金额都较低，且多数情况下会选择全额还款。他们可能只是将信用卡作为支付手段，而非金融工具。

群体 2: 现金依赖型客户

1. 特征表现
    1) 群体 2 客户在现金预支金额现金预支频率上显著高于其他群体。
    2) 他们的账户余额和消费总金额等指标则相对中等，说明这些客户更依赖于信用卡的现金功能，而不是消费功能。
    3) 购买频率和分期购买频率偏低，表明他们较少使用信用卡进行日常消费。
2. 总结：

群体 2 的客户更倾向于将信用卡作为短期融资工具（如应急现金预支），而不是日常消费工具。他们的现金预支行为可能与短期资金周转需求有关。

群体 3: 高消费/高活跃客户

1. 特征表现
   1) 群体 3 消费金额购买交易次数远高于其他群体，显示出极高的消费活跃度。
   2) 他们的账户余额和现金预支金额也较高，表明这些客户可能具有较强的消费能力和资金需求。
   3) 该群体账户时长分布较广，说明这些客户可能包含新用户和老用户的混合体。
2. 总结

群体 3 代表银行的高价值客户群，他们频繁使用信用卡进行大额消费，并可能偶尔利用现金预支功能。这些客户是信用卡服务的主要目标群体，具有较高的利润潜力。

群体 4: 低活跃/保守型客户

1. 特征表现
   1) 群体 4 的账户余额和消费金额处于最低水平，表明这些客户很少使用信用卡进行消费。
   2) 他们的购买交易次数和现金预支金额也极低，显示他们几乎不依赖信用卡进行交易或现金周转。
   3) 在现金预支频率和消费频率上的表现同样较低，进一步证实了他们的低活跃度。
   4) 账户时长较长，表明这些客户可能已经持卡多年，但使用率持续较低。

体会：

在清洗数据的时候，要特别注意不要在原表格上直接进行修改。最好在原始数据的基础上创建备份，以防在清洗过程中不小心出现错误，破坏数据。保留原始数据能够随时恢复到初始状态。

其次，在每一个清洗数据的步骤后，都最好重新输出一下结果，检查清洗是否到位。实验中，在清除并填补空白值的时候，我漏掉了一列的空白值没有填补，导致后续的步骤输出的结果出现问题，后来一步步找回去才发现是清洗数据的时候没有清洗完整，浪费了一些时间。

K-means 算法的概念在理论上相对直观，实际应用中虽然可以直接调用 API，但是操作也可能会遇到很多问题，比如调整 K-means 算法参数的时候需要一些耐心，最好进行多次尝试。K-means 参数是一个迭代的过程，需要多次调试和评估。通过实验不同的参数组合，我也更好地理解了它们如何影响聚类结果。并且最后要通过肘部法则和其他方法共同确定最佳 K 值，而不是只用肘部法则或者只用 Calinski-Harabasz 等指数进行确定，如果选择的 K 值不是结合多个方法且权衡下最好的 K 值，那么不合适的分类还会影响后续各种评估。

在使用多个指标之前，我对指标评估的结果是否一致有些担心。所幸在本实验的实际评估过程中，指标指向的最优 k 值是一样的。当我们有多个评估指标，且每个指标对于最佳的 k 值有不同的建议时，决定使用哪个指标来选择聚类模型的聚类数量可能会很有挑战性。一方面可以查看每个指标建议的 k 值对应的聚类结果的特征，并将其与领域知识或业务需求进行比较，可以识别出能够产生最有意义或最有用的聚类的 k 值。L 另一方面可以使用一个综合指标，将多个评估指标合并为一个单一的分数。例如，可以使用轮廓系数和 Calinski-Harabasz 分数的几何平均值来对 k 值进行排名，并选择具有最高综合分数的 k 值。这种方法可以平衡不同评估指标的优缺点，并减少对单一指标过度拟合的风险。

思考题

在金融或商业领域的实际应用中，机器学习系统的构建流程是怎么样的？以及机器学习系统在训练和部署会遇到什么样的阻碍？

机器学习系统的构建流程：

1. 需求分析

我们想要解决什么问题？这是一个分类、回归还是聚类问题？我们如何衡量成功？

2. 数据采集

在数据收集阶段，可以通过以下几种方式获取数据：利用机构内部数据源、编写网络爬虫从互联网抓取数据、从企业数据库中提取数据，以及使用 UCI 机器学习库和 Kaggle datasets 等开源数据集。

3. 数据清洗

在数据预处理阶段，主要关注点包括处理缺失值、异常值处理以及格式统一。处理缺失值时，可以选择删除含有缺失值的记录或者用统计方法如均值、中位数或众数来填充。异常值处理涉及识别那些明显偏离正常范围的数据点，并决定是删除还是进行修正。最后，确保所有数据格式一致是必要的，这包括日期、数值和分类数据的格式，以便数据可以被模型正确读取和处理。

4. 数据分析与可视化

在机器学习项目中，数据分析与可视化是理解数据的关键步骤。探索性数据分析（EDA）涉及对数据进行深入的统计分析和可视化，这有助于揭示数据的基本特征和潜在的问题点。紧接着，统计绘图通过使用柱状图、散点图等图形工具，进一步直观地展示数据的分布情况和变量之间的关系，为后续的特征工程和模型训练提供直观的数据理解。

5. 特征工程

特征工程是机器学习中至关重要的一步，它直接影响模型的性能。特征选择是从所有可用特征中挑选出与目标变量最相关的特征，以减少模型的复杂性和提高预测的准确性。特征构建则涉及创建新的特征或对现有特征进行转换，这可以增强模型捕捉数据中复杂关系的能力。此外，特征编码如独热编码等处理，是将分类变量转换为模型可以处理的数值型数据，这对于处理分类数据尤为重要。

6. 机器学习建模与调优

在机器学习建模与调优阶段，首先需要根据问题的具体类型选择合适的机器学习算法。一旦模型选定，接下来的任务是使用训练数据集对模型进行训练，这一过程中模型会学习数据中的模式和规律。随后，超参数调优成为提升模型性能的关键，通过网格搜索、随机搜索或贝叶斯优化等方法调整模型参数，以找到最佳的模型配置。

7. 模型评估

模型评估可以验证模型性能。交叉验证，尤其是 K 折交叉验证，是一种常用的方法，它通过将数据集分成多个子集并对每个子集进行训练和验证，来评估模型的稳定性和准确性。此外，性能指标的选择也至关重要，根据不同的问题类型，如分类或回归，会选择合适的评估指标，例如准确率、召回率、F1 分数等，来量化模型的性能。

8. 模型融合

模型融合是一种提高模型性能的技术，它通过结合多个模型的预测结果来实现。常见的融合方法包括投票、堆叠等，这些方法可以帮助我们减少模型的方差或偏差，从而提高模型的整体性能。

9. 模型部署

模型部署是将训练好的模型应用到实际生产环境中的过程。这不仅涉及到模型的上线，还包括性能监控，即持续跟踪模型在实际应用中的表现，并根据反馈进行必要的调整，以确保模型的长期稳定性和有效性。

10. 模型维护与更新

模型的维护与更新是机器学习项目的持续阶段。模型重训练是指定期使用新收集的数据对模型进行更新，以保持模型的准确性和时效性，这对于应对数据漂移和环境变化至关重要。同时，模型优化是一个动态过程，需要根据业务发展和数据变化不断调整和改进模型结构和参数，以适应新的数据分布和业务需求。

训练阶段的阻碍：

1. 训练数据的数量不足

机器学习算法通常需要大量数据才能正常工作，尤其是对于复杂问题如图像或语音识别。足够的数据可以使不同机器学习算法的表现几乎一致，强调了数据比算法更重要。

2. 训练数据不具代表性

训练数据需要对将要泛化的新示例有代表性，无论是基于实例的学习还是基于模型的学习。采样偏差和无反应偏差可能导致训练集非代表性，影响模型的泛化能力。

3. 低质量数据

训练集中的错误、异常值和噪声会使得系统难以检测到底层模式，影响模型表现。

4. 无关特征

只有包含足够多的相关特征和较少的无关特征，系统才能完成学习。

5. 过拟合训练数据

过拟合是指模型在训练数据上表现良好，但在新数据上泛化能力差。虽然诸如深度神经网络这类的复杂模型可以检测到数据中的微小模式，但是如果训练集本身是有噪声的，或者数据集太小，那么很可能会导致模型检测噪声本身的模式。当模型相对于训练数据的数量和噪度都过于复杂时，会发生过拟合。

6. 欠拟合训练数据

欠拟合通常发生在模型相对于数据结构过于简单时，例如，用线性模型来描述生活满意度就属于欠拟合。现实情况远比模型复杂得多，所以即便是对于用来训练的示例，该模型产生的预测都一定是不准确的。

部署阶段的阻碍：

1. 数据科学语言管理

机器学习应用程序常由不同编程语言编写，这些语言间的交互存在障碍。需要将模型从原型语言（如 R 或 Python）移植到生产环境常用的语言（如 C++或 Java），这可能降低模型性能。R 语言在新版本软件发布时可能崩溃，且运行速度慢，不适合处理大数据。需要机器学习生命周期工具自动跟踪依赖，并与模型捆绑部署。

2. 算力和 GPU

现代神经网络深度大，训练和推理需要大量算力，GPU 资源稀缺且昂贵。

3. 可移植性

软件组件移植到不同主机环境的能力不足，限制了模型的部署。

4. 可扩展性

模型需能够应对数据量和性能需求的增加，需要多种工具监管性能和可扩展性。

5. 在极限状态下进行机器学习计算

API 调用量可能在短时间内急剧增加，扩大和缩减部署规模是挑战。

6. 提升鲁棒性使模型可以运行

需要大量工作将原型模型准备就绪，可能需要重新编码以适应现有架构。并且在生产环境下访问数据困难，数据仓库存在技术和组织上的负担。

7. 测试和验证的问题

模型不断演进，每次变更都需要重新验证性能。其次，使用相同的测试集和验证集评价模型，保证可比性。而且，性能提升可能带来预测时间增长，需要基准对比测试和负载和压力测试。