中北大学软件学院

实 验 报 告

专　　业：	软件工程
方　　向：	人工智能
课程名称：	机器学习实践
班　　级：	22130418
学　　号：	2213041835、2213041816
姓　　名：	吕炳荣、刘晓峰
辅导教师：	程晓鼎

2024 年 3 月制

成绩：_____

实验时间	2024 年 4 日 21 日 14 时至 18 时	学时数	4 学时

1. 实验名称

航空公司客户价值的 K-Means 分析

2. 实验目的

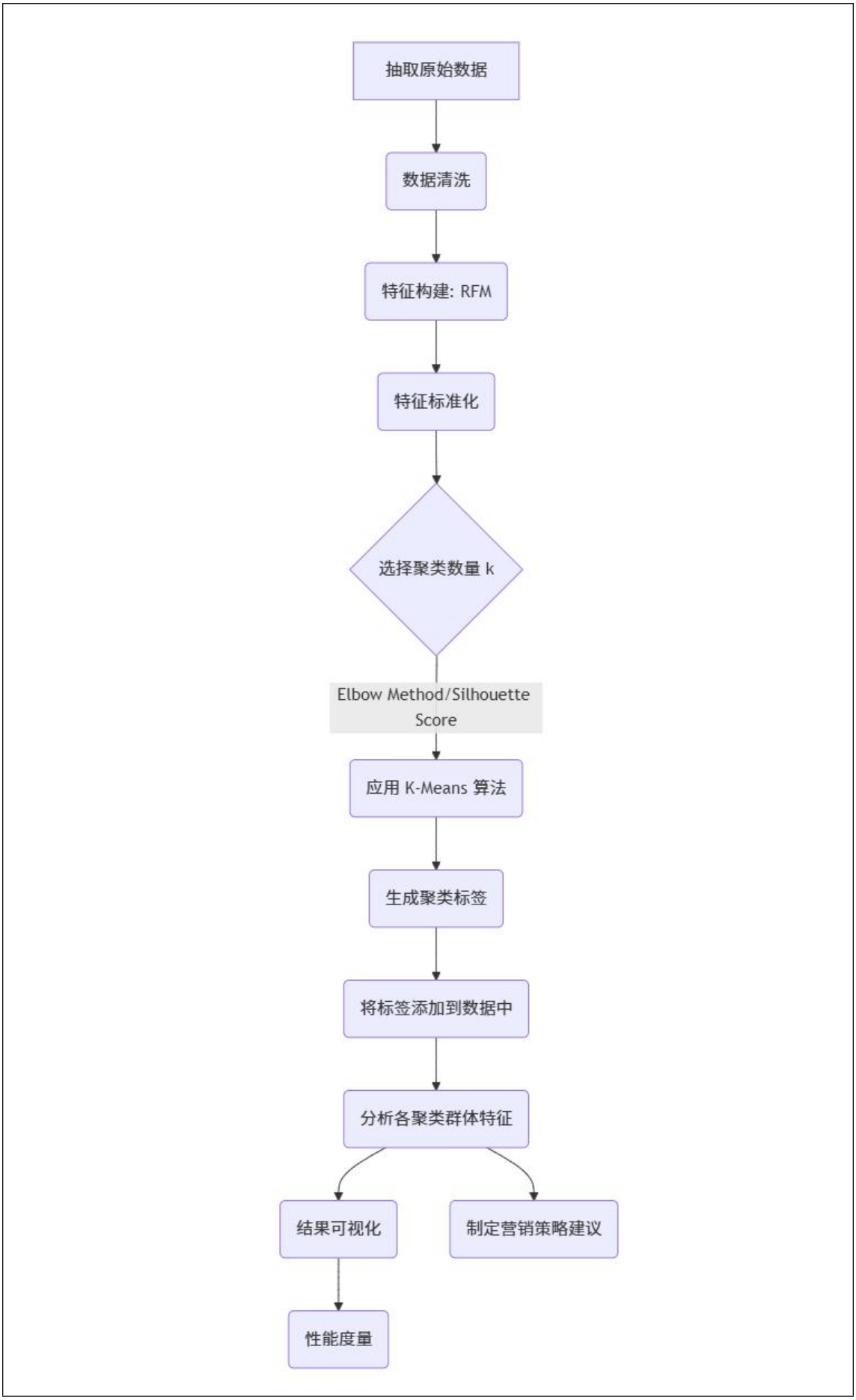
熟悉和掌握 K-Means 的聚类原理和过程，并会采用 K-Means 模型根据给定的数据进行实际应用，掌握 K-Means 模型的编码实现，能根据模型结果进行性能度量和结果分析。

3.实验内容

1、对提供的数据进行数据清洗、特征构建和标准化等操作。
2、使用 K-Means 算法进行客户分群，并进性能度量和结果分析。

4.实验原理或流程图

1、抽取数据。
2、对抽取的数据进行数据清洗、特征构建和标准化等操作。
3、基于 RFM 模型，使用 K-Means 算法进行客户分群。
4、针对模型结果得到不同价值的客户，进行营销策略分析。


抽取原始数据
数据清洗
特征构建: RFM
特征标准化
选择聚类数量 k
Elbow Method/Silhouette Score
应用 K-Means 算法
生成聚类标签
将标签添加到数据中
分析各聚类群体特征
结果可视化
制定营销策略建议
性能度量

5.实验过程或源代码

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
import datetime as dt

import warnings # 导入 warnings 模块来控制警告信息


# 忽略 KMeans n_init 的未来警告

warnings.filterwarnings("ignore", message="The default value of `n_init` will change from 10 to 'auto' in 1.4.")

# 忽略 seaborn palette 的未来警告

warnings.filterwarnings("ignore", message="Passing `palette` without assigning `hue` is deprecated")



# --- 设置字体以支持中文显示 ---

# 尝试使用多种中文字体，提高兼容性

plt.rcParams['font.sans-serif'] = ['SimHei', 'Arial Unicode MS', 'WenQuanYi Zen Hei', 'Microsoft YaHei']

plt.rcParams['axes.unicode_minus'] = False # 解决保存图像时

负号'-'显示为方块的问题


# 设置 matplotlib 和 seaborn 的显示风格，使图表更美观

plt.style.use('seaborn-v0_8-whitegrid')
sns.set_style('whitegrid')


# --- 辅助函数：用于进行聚类、分析和可视化的通用流程 ---

def perform_clustering_analysis(data, feature_cols, k, title_prefix=""):
    """
```

```
    执行标准化、K-Means聚类、结果分析和部分可视化。

    Args:
        data (pd.DataFrame): 包含原始数据的DataFrame。
        feature_cols (list): 用于聚类的特征列名列表。
        k (int): 聚类数量。
        title_prefix (str): 图表和输出信息的前缀标题。

    Returns:
        tuple: (kmeans_model, clustered_df, silhouette_score)
                    返回训练好的KMeans模型，带有Cluster标签的DataFrame，轮廓系数。
    """
    print(f"\n--- {title_prefix} ---")
    print(f"使用的聚类特征: {feature_cols}")
    print(f"用于聚类的数据量: {data.shape[0]} 行")

    # 提取特征列并进行拷贝，避免修改原始传入的data DataFrame
    features_df = data[feature_cols].copy()

    # 检查用于聚类的特征列是否有NaN（尽管清洗后应无，但再次确认，并直接在features_df上处理）
    if features_df.isnull().sum().sum() > 0:
        print(f"警告: 用于聚类的特征中仍存在NaN值，已自动删除对应行。")
        initial_feature_rows = features_df.shape[0]
        features_df.dropna(inplace=True)
```

```
            print(f"删除含NaN行后，剩余{features_df.shape[0]}行（删除{initial_feature_rows - features_df.shape[0]}行）")


    # 检查处理后的特征DataFrame是否为空
    if features_df.shape[0] == 0:
        print(f"错误：处理后的特征数据集为空，无法进行聚类。请检查数据清洗和特征提取步骤。")
        return None, data.copy(), -1 # 返回None模型和原始数据，轮廓系数为-1


    # 特征标准化
    scaler = StandardScaler()
    features_scaled = scaler.fit_transform(features_df)
    # 使用原始features_df的索引创建标准化后的DataFrame
    features_scaled_df = pd.DataFrame(features_scaled, columns=feature_cols, index=features_df.index)
    print("特征标准化完成.")


    # 应用K-Means算法
    print(f"正在使用 K-Means 进行聚类 (k={k})...")
    # 使用 StandardScaler 转换后的数据进行训练
    kmeans = KMeans(n_clusters=k, random_state=42, n_init='auto')
    kmeans.fit(features_scaled_df)
```

```
    # 将聚类标签添加到 DataFrame 中
    # 使用原始数据，但只保留 features_df 中存在的行（即 NaN 处理后的行）
    clustered_df = data.loc[features_df.index].copy()
    clustered_df['Cluster'] = kmeans.labels_

    # 分析各聚类群体特征
    cluster_means = clustered_df.groupby('Cluster')[feature_cols].mean()
    print(f"\n{title_prefix} 各聚类群体特征均值:")
    print(cluster_means)

    # 性能度量 (轮廓系数)
    silhouette_avg = -1 # 默认值，如果数据量过小或 k=1 则无法计算
    if k > 1 and features_scaled_df.shape[0] > k: # 轮廓系数要求样本数 > 簇数
        try:
            silhouette_avg = silhouette_score(features_scaled_df, kmeans.labels_)
            print(f"\n{title_prefix} 轮廓系数: {silhouette_avg:.4f}")
        except ValueError as e:
            print(f"\n{title_prefix} 计算轮廓系数出错: {e}")
            print("可能是数据量太小或簇分配问题。")
            silhouette_avg = -1
    elif k==1:
        print(f"\n{title_prefix} k=1 时轮廓系数无意义。")
    else:
```

```
            print(f"\n{title_prefix} 数据量不足或 k 值问题，无法计算轮廓系数。数据量: {features_scaled_df.shape[0]}, k: {k}")


    # 可视化
    print(f"绘制 {title_prefix} 聚类结果可视化图表...")
    plt.figure(figsize=(len(feature_cols) * 4, 5)) # 根据特征数量调整图宽度
    for i, col in enumerate(feature_cols):
        plt.subplot(1, len(feature_cols), i + 1)
        # 使用 hue 参数将颜色与 Cluster 关联，以避免 FutureWarning
        sns.boxplot(x='Cluster', y=col, data=clustered_df, palette='viridis', hue='Cluster', legend=False)
        plt.title(f'{title_prefix}\n{col} by Cluster')
        plt.xlabel('客户群体 (簇)')
        plt.ylabel(col)
    plt.tight_layout()
    plt.show()

    plt.figure(figsize=(8, 5))
    # 使用 hue 参数将颜色与 Cluster 关联，以避免 FutureWarning
    sns.countplot(x='Cluster', data=clustered_df, palette='viridis', hue='Cluster', legend=False)
    plt.title(f'{title_prefix} 各聚类群体的客户数量')
    plt.xlabel('客户群体 (簇)')
    plt.ylabel('客户数量')
    plt.show()

    return kmeans, clustered_df, silhouette_avg
```

```
# --- 加载原始数据 ---
print("--- 数据加载 ---")
file_path = 'air_data.csv'
try:
    # 尝试使用 utf-8 编码读取，如果失败再尝试 gbk 或 gb2312
    df_raw = pd.read_csv(file_path, encoding='utf-8')
except Exception as e:
    print(f"UTF-8 编码读取失败: {e}, 尝试使用 gbk...")
    try:
        df_raw = pd.read_csv(file_path, encoding='gbk')
    except Exception as e:
        print(f"GBK 编码读取失败: {e}, 尝试使用 gb2312...")
        try:
            df_raw = pd.read_csv(file_path, encoding='gb2312')
        except Exception as e:
            print(f"GB2312 编码读取失败: {e}, 请检查文件路径和编码.")
            exit() # 如果都失败，退出程序

print("数据加载成功，前 5 行数据:")
print(df_raw.head())
print("\n 数据基本信息:")
df_raw.info()
print(f"\n 原始数据共 {df_raw.shape[0]} 行.")

# --- 原始实验：基于基本清洗和总 RFM 特征 ---
print("\n\n--- 原始实验: 基本清洗 + 总 RFM 特征 ---")
```

```
# 1. 原始清洗步骤 (基于删除)
df_basic_cleaned = df_raw.copy()
initial_rows_basic = df_basic_cleaned.shape[0]

# 处理 LAST_TO_END 中的异常标记，如'########'，并转换为数值
# errors='coerce' 会将无法转换的值设为 NaN
df_basic_cleaned['LAST_TO_END'] = pd.to_numeric(df_basic_cleaned['LAST_TO_END'], errors='coerce')

# 处理缺失值：删除关键 RFM 特征存在缺失值的行（包括上面转换后产生的 NaN）
basic_subset_cols = ['LAST_TO_END', 'FLIGHT_COUNT', 'Points_Sum']
df_basic_cleaned.dropna(subset=basic_subset_cols, inplace=True)

# 处理异常值：删除负值 (只对需要非负的特征进行)
df_basic_cleaned = df_basic_cleaned[(df_basic_cleaned['FLIGHT_COUNT'] >= 0) & (df_basic_cleaned['Points_Sum'] >= 0)]

print(f"原始实验清洗后，剩余 {df_basic_cleaned.shape[0]} 行 (删除 {initial_rows_basic - df_basic_cleaned.shape[0]} 行)")

# 2. 特征构建 (总 RFM: LAST_TO_END, FLIGHT_COUNT, Points_Sum)
rfm_cols_total = ['LAST_TO_END', 'FLIGHT_COUNT', 'Points_Sum']
```

```
# 3. 选择最佳聚类数量 k (在原始实验数据上进行 k 选择)

# 使用 Elbow 方法和 Silhouette Score

inertia_total = []
silhouette_scores_total = []

k_range = range(2, 11) # 尝试 2 到 10 个簇


print("\n 正在计算原始实验数据不同 k 值下的 Inertia 和 Silhouette Score...")

# 确保用于 k 选择的数据集是非空的

if df_basic_cleaned.shape[0] > 1: # 至少需要两个样本才能计算 silhouette score for k=2

    rfm_total_for_k_selection = df_basic_cleaned[rfm_cols_total].copy()
    scaler_k = StandardScaler()
    rfm_total_scaled_k = scaler_k.fit_transform(rfm_total_for_k_selection)

    for k in k_range:
        if rfm_total_scaled_k.shape[0] < k: # 如果数据量小于 k，跳过
            print(f"数据量({rfm_total_scaled_k.shape[0]}) 小于 k ({k})，跳过。")
            inertia_total.append(np.nan)
            silhouette_scores_total.append(np.nan)
            continue

        kmeans_k = KMeans(n_clusters=k, random_state=42, n_init='auto')
        kmeans_k.fit(rfm_total_scaled_k)
        inertia_total.append(kmeans_k.inertia_)
        if k > 1:
            try:
                score_k =
```

```
silhouette_score(rfm_total_scaled_k, kmeans_k.labels_)
                    silhouette_scores_total.append(score_k)
                except ValueError: # 避免少数情况下的计算错误
                    silhouette_scores_total.append(np.nan)
            else: # k=1
                silhouette_scores_total.append(np.nan)

    # 可视化Elbow方法和Silhouette Score结果 (原始实验)
    plt.figure(figsize=(14, 6))
    plt.subplot(1, 2, 1)
    plt.plot(k_range, inertia_total, marker='o')
    plt.xlabel('聚类数量 (k)')
    plt.ylabel('簇内平方和 (Inertia)')
    plt.title('原始实验: 使用 Elbow 方法选择最佳 k 值')
    plt.xticks(k_range)
    plt.grid(True)

    plt.subplot(1, 2, 2)
    # 注意：轮廓系数从 k=2 开始计算
    plt.plot(k_range[1:], silhouette_scores_total[1:], marker='o')
    plt.xlabel('聚类数量 (k)')
    plt.ylabel('轮廓系数 (Silhouette Score)')
    plt.title('原始实验: 使用轮廓系数选择最佳 k 值')
    plt.xticks(k_range[1:])
    plt.grid(True)

    plt.tight_layout()
    plt.show()

    # 根据图示选择最佳 k 值 (在此处手动设置，例如 k=4)
    optimal_k_original = 4
```

```
    print(f"\n根据原始实验数据分析，选择最佳聚类数量 k = {optimal_k_original}")
else:
    print("原始实验清洗后数据量不足，无法进行 K 值选择和聚类。请检查清洗规则。")
    optimal_k_original = None # 标记无法进行后续实验

# 4. 执行原始实验的聚类和分析 (仅在数据量足够时进行)
kmeans_original, df_clustered_original, silhouette_original = None, pd.DataFrame(), -1 # 初始化结果变量

if optimal_k_original is not None and df_basic_cleaned.shape[0] > 0:
    kmeans_original, df_clustered_original, silhouette_original = perform_clustering_analysis(
        df_basic_cleaned, rfm_cols_total, optimal_k_original, title_prefix="原始实验"
    )
else:
    print("\n跳过原始实验聚类，因为数据量不足或 K 值未确定。")


# --- 消融实验一：使用最近一年 RFM 特征 ---
print("\n\n--- 消融实验一：使用最近一年 RFM 特征 ---")
print("对比原始实验，这里更换了聚类特征，清洗方法保持与原始实验一致。")

# 使用与原始实验相同的清洗后的数据作为起点
```

```
# 确保原始实验成功且有数据
if df_basic_cleaned is not None and df_basic_cleaned.shape[0] > 0:
    df_l1y_experiment = df_basic_cleaned.copy()

    # 1. 特征构建 (最近一年 RFM: LAST_TO_END, L1Y_Flight_Count, L1Y_Points_Sum)
    rfm_cols_l1y = ['LAST_TO_END', 'L1Y_Flight_Count', 'L1Y_Points_Sum']

    # 检查新特征是否存在并进行安全处理（理论上原始清洗已处理 LAST_TO_END NaN）

    # 检查 L1Y_Flight_Count 和 L1Y_Points_Sum 中的缺失值或负值 (虽然 info 显示它们非空，但仍检查)
    l1y_check_cols = ['L1Y_Flight_Count', 'L1Y_Points_Sum']
    initial_rows_l1y_exp = df_l1y_experiment.shape[0]

    # 检查缺失值
    if df_l1y_experiment[rfm_cols_l1y].isnull().sum().sum() > 0:
        print("\n消融实验一：检查到新特征有缺失值，已删除对应行。")
        df_l1y_experiment.dropna(subset=rfm_cols_l1y, inplace=True)

    # 检查负值 (仅对可能非负的特征)
    df_l1y_experiment = df_l1y_experiment[(df_l1y_experiment[l1y_check_cols] >= 0).all(axis=1)]

    print(f"消融实验一清洗后，剩余
```

```
{df_l1y_experiment.shape[0]} 行 (删除 {initial_rows_l1y_exp - df_l1y_experiment.shape[0]} 行)")


    # 2. 执行聚类和分析 (使用原始实验确定的最佳 k 值，并确保数据量足够)
    kmeans_l1y, df_clustered_l1y, silhouette_l1y = None, pd.DataFrame(), -1 # 初始化
    if optimal_k_original is not None and df_l1y_experiment.shape[0] > 0:
        kmeans_l1y, df_clustered_l1y, silhouette_l1y = perform_clustering_analysis(
            df_l1y_experiment, rfm_cols_l1y, optimal_k_original, title_prefix="消融实验一 (L1Y Features)"
        )
    else:
        print("\n 跳过消融实验一聚类，因为数据量不足或 K 值未确定。")

else:
    print("\n 跳过消融实验一，因为原始实验数据量不足。")


# --- 消融实验二: 使用 IQR 方法处理异常值 (修改为 Capping) ---
print("\n\n--- 消融实验二: 使用 IQR 方法处理异常值 (修改为 Capping) ---")
print("对比原始实验，这里更改了数据清洗方法（IQR 异常值封顶），聚类特征保持与原始实验一致。")
```

```
# 从原始未清洗的数据开始，或至少是 LAST_TO_END NaN 处理后的数据
df_iqr_experiment = df_raw.copy()
initial_rows_iqr = df_iqr_experiment.shape[0]

# 处理 LAST_TO_END 中的异常标记（与原始实验相同）
df_iqr_experiment['LAST_TO_END'] = pd.to_numeric(df_iqr_experiment['LAST_TO_END'], errors='coerce')
df_iqr_experiment.dropna(subset=['LAST_TO_END'], inplace=True) # 删除转换后为 NaN 的行

print(f"消融实验二 (IQR 清洗) 前，处理 LAST_TO_END NaN 后，剩余 {df_iqr_experiment.shape[0]} 行")

# 1. IQR 清洗步骤 (修改为 Capping)
print("正在进行 IQR 异常值封顶处理...")
# 对 RFM 特征（LAST_TO_END, FLIGHT_COUNT, Points_Sum）进行 IQR 异常值检测和封顶
rfm_cols_for_iqr_capping = ['LAST_TO_END', 'FLIGHT_COUNT', 'Points_Sum']
df_iqr_cleaned = df_iqr_experiment.copy() # 从处理了 LAST_TO_END NaN 的数据开始

# 计算 IQR 并进行封顶
for col in rfm_cols_for_iqr_capping:
    if col in df_iqr_cleaned.columns: # 确保列存在
        # 检查列是否有足够的非 NaN 数据来计算分位数
        if df_iqr_cleaned[col].dropna().shape[0] < 2:
```

```
                print(f"警告：列 '{col}' 数据量不足
({df_iqr_cleaned[col].dropna().shape[0]} 个非 NaN 值)，无法
计算 IQR 和封顶，跳过。")
                continue

            Q1 = df_iqr_cleaned[col].quantile(0.25)
            Q3 = df_iqr_cleaned[col].quantile(0.75)
            IQR = Q3 - Q1

            lower_bound = Q1 - 1.5 * IQR
            upper_bound = Q3 + 1.5 * IQR

            # 封顶操作：将小于下界的设为下界，将大于上界
的设为上界
            # 注意：这里将下界设为 0，因为 FLIGHT_COUNT
和 Points_Sum 不能为负
            if col in ['FLIGHT_COUNT', 'Points_Sum']:
                lower_bound = max(0, lower_bound) # 确保
飞行次数和积分不为负

            # 使用 clip 方法进行封顶
            df_iqr_cleaned[col] =
df_iqr_cleaned[col].clip(lower=lower_bound,
upper=upper_bound)
            print(f"已对列 '{col}' 应用 IQR
({lower_bound:.2f}, {upper_bound:.2f}) 封顶。")
        else:
            print(f"警告：用于 IQR 处理的列 '{col}' 不存在。
")
```

```
print(f"消融实验二 (IQR清洗 - Capping) 后，剩余 {df_iqr_cleaned.shape[0]} 行 (此方法不删除行，只修改异常值)")
# 对比原始清洗删除的行数 (initial_rows_basic - df_basic_cleaned.shape[0])

# 可视化清洗后的数据分布（与原始清洗后的数据分布对比）
# 确保原始实验的数据集存在且非空，以便进行对比
if 'df_basic_cleaned' in locals() and df_basic_cleaned.shape[0] > 0:
    print("\n绘制清洗后RFM特征的箱线图，对比清洗效果...")

    plt.figure(figsize=(15, 8)) # 增加图高度，容纳更多信息

    plt.suptitle("清洗方法对比：基本清洗 (删除) vs IQR清洗 (封顶) (RFM特征)", y=1.02, fontsize=14)


    # --- 基本清洗后的箱线图 ---
    plt.subplot(2, 3, 1)
    sns.boxplot(y='LAST_TO_END', data=df_basic_cleaned, color='skyblue')
    plt.title('基本清洗 (删除)\nLAST_TO_END')

    plt.ylabel('') # 避免重复ylabel


    plt.subplot(2, 3, 2)
    sns.boxplot(y='FLIGHT_COUNT', data=df_basic_cleaned, color='skyblue')
    plt.title('基本清洗 (删除)\nFLIGHT_COUNT')
    plt.ylabel('')
```

```
        plt.subplot(2, 3, 3)
        sns.boxplot(y='Points_Sum',        data=df_basic_cleaned,
color='skyblue')
        plt.title('基本清洗 (删除)\nPoints_Sum')
        plt.ylabel('')


        # --- IQR 清洗 (Capping) 后的箱线图 ---
        plt.subplot(2, 3, 4)
        sns.boxplot(y='LAST_TO_END',      data=df_iqr_cleaned,
color='lightgreen')
        plt.title('IQR 清洗 (封顶)\nLAST_TO_END')
        plt.ylabel('值') # 加上一个通用 ylabel


        plt.subplot(2, 3, 5)
        sns.boxplot(y='FLIGHT_COUNT',    data=df_iqr_cleaned,
color='lightgreen')
        plt.title('IQR 清洗 (封顶)\nFLIGHT_COUNT')
        plt.ylabel('')


        plt.subplot(2, 3, 6)
        sns.boxplot(y='Points_Sum',          data=df_iqr_cleaned,
color='lightgreen')
        plt.title('IQR 清洗 (封顶)\nPoints_Sum')
        plt.ylabel('')


        plt.tight_layout(rect=[0, 0.03, 1, 0.95]) # 调整布局避免
标题重叠
        plt.show()
else:
         print("\n 跳过清洗效果对比箱线图绘制，因为原始实验
数据量不足。")
```

```
# 2. 执行聚类和分析 (使用原始实验确定的最佳 k 值，并确保数据量足够)
kmeans_iqr, df_clustered_iqr, silhouette_iqr = None, pd.DataFrame(), -1 # 初始化
if optimal_k_original is not None and df_iqr_cleaned.shape[0] > 0:
    kmeans_iqr, df_clustered_iqr, silhouette_iqr = perform_clustering_analysis(
        df_iqr_cleaned, rfm_cols_total, optimal_k_original, title_prefix="消融实验二 (IQR 清洗 - Capping)"
    )
else:
    print("\n 跳过消融实验二聚类，因为数据量不足或 K 值未确定。")


# --- 实验结果对比总结 ---
print("\n\n--- 实验结果对比总结 ---")

print(f"原始实验 (基本清洗 + 总 RFM):")
if df_clustered_original.shape[0] > 0:
    print(f"  使用数据量: {df_clustered_original.shape[0]} 行")
    print(f"  轮廓系数: {silhouette_original:.4f}")
    print("  各聚类群体 RFM 特征均值:")
    print(df_clustered_original.groupby('Cluster')[rfm_cols_total].mean())
else:
    print("  未成功执行聚类，数据量不足。")
```

```
print(f"\n 消融实验一 (基本清洗 + L1Y RFM):")
if df_clustered_l1y.shape[0] > 0:
    print(f"  使用数据量: {df_clustered_l1y.shape[0]} 行")
    print(f"  轮廓系数: {silhouette_l1y:.4f}")
    print("  各聚类群体 L1Y RFM 特征均值:")
    # 确保 rfm_cols_l1y 在 grouped df 中存在
    try:
print(df_clustered_l1y.groupby('Cluster')[rfm_cols_l1y].mean())
    except KeyError:
        print("  L1Y 特征列不存在于聚类结果中,请检查。")
else:
    print("  未成功执行聚类，数据量不足。")


print(f"\n 消融实验二 (IQR 清洗 - Capping + 总 RFM):")
if df_clustered_iqr.shape[0] > 0:
    print(f"  使用数据量: {df_clustered_iqr.shape[0]} 行")
    print(f"  轮廓系数: {silhouette_iqr:.4f}")
    print("  各聚类群体 RFM 特征均值:")
    try:
print(df_clustered_iqr.groupby('Cluster')[rfm_cols_total].mean())
    except KeyError:
        print("  总 RFM 特征列不存在于聚类结果中,请检查。")
```

```
else:
    print("  未成功执行聚类，数据量不足。")

print("\n--- 总结与讨论 ---")
print("对比不同实验的客户数量、轮廓系数以及各聚类群体的特征均值，可以评估不同特征集和不同清洗方法对聚类结果的影响。")
print("例如：")
print("- **清洗方法对比:** 原始实验使用简单删除异常值，消融实验二使用 IQR 封顶。对比两者使用的最终数据量和聚类结果（轮廓系数、群体特征），可以看出异常值对聚类稳定性和群体划分的影响。IQR 封顶通常保留更多数据，且可能受极端值影响更小，但也可能模糊掉一些真实的高价值客户特征（如果他们的极端值是真实的）。")
print("- **特征集对比:** 原始实验使用总 RFM 特征，消融实验一使用最近一年 RFM 特征。对比两者在相似数据量（如果清洗方法一致）下的聚类结果，可以评估哪种特征更能有效区分客户群体，例如最近一年特征可能更能识别活跃客户，而总特征更能识别累积价值客户。")
print("这些对比结果将作为实验报告中“消融实验”和“结果分析”部分的重要依据。")
```

```
--- 数据加载 ---
数据加载成功，前5行数据:
   MEMBER_NO   FFP_DATE  ... Ration_L1Y_BPS Point_NotFlight
0      54993  2006/11/2  ...       0.512777              50
1      28065  2007/2/19  ...       0.510708              33
2      55106   2007/2/1  ...       0.518530              26
3      21189  2008/8/22  ...       0.448275              12
4      39546  2009/4/10  ...       0.530943              39

[5 rows x 44 columns]
```

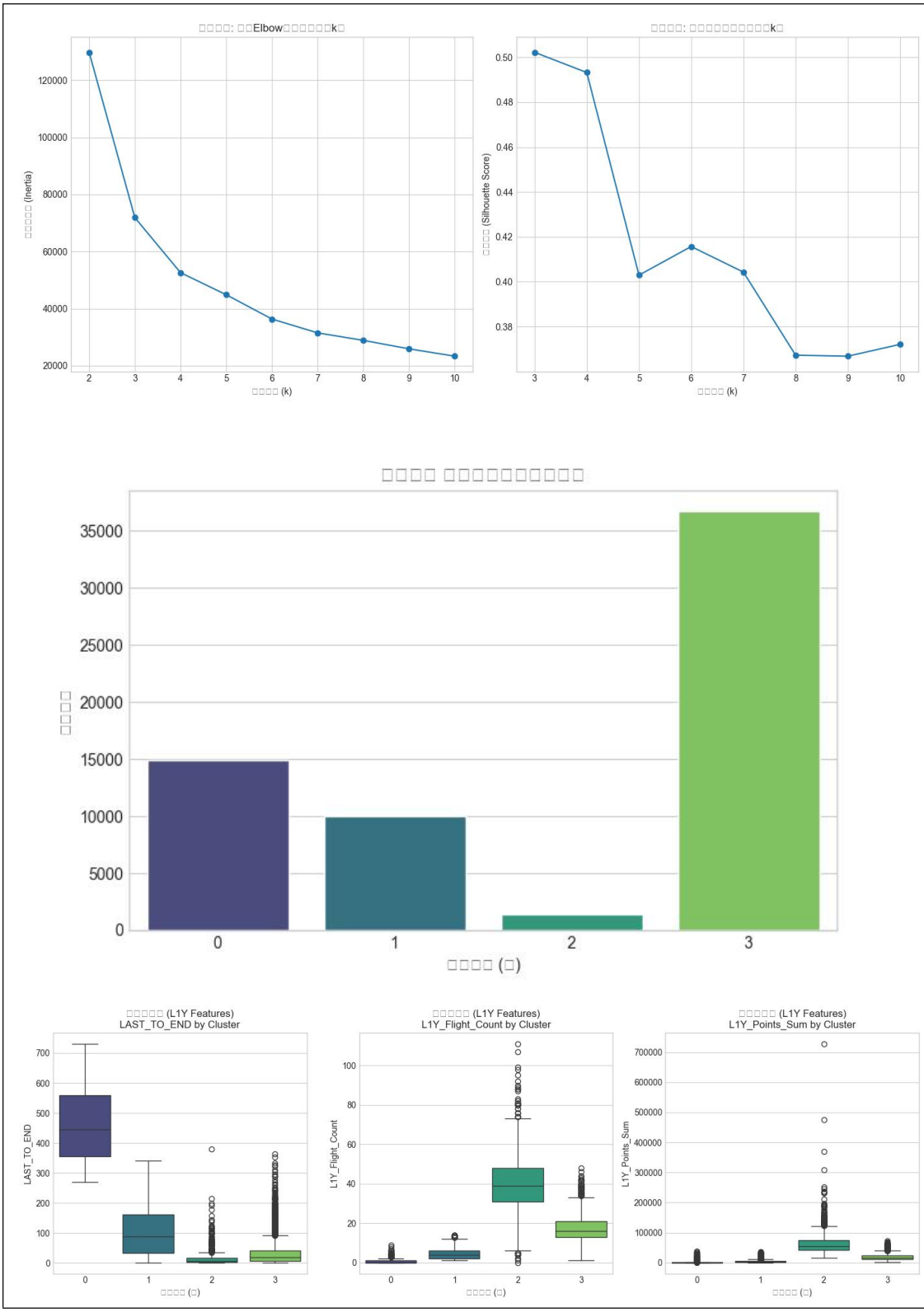
```
数据基本信息:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 62988 entries, 0 to 62987
Data columns (total 44 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   MEMBER_NO               62988 non-null  int64
 1   FFP_DATE                62988 non-null  object
 2   FIRST_FLIGHT_DATE       62988 non-null  object
 3   GENDER                  62985 non-null  object
 4   FFP_TIER                62988 non-null  int64
 5   WORK_CITY               60719 non-null  object
 6   WORK_PROVINCE           59740 non-null  object
 7   WORK_COUNTRY            62962 non-null  object
 8   AGE                     62568 non-null  float64
 9   LOAD_TIME               62988 non-null  object
 10  FLIGHT_COUNT            62988 non-null  int64
 11  BP_SUM                  62988 non-null  int64
 12  EP_SUM_YR_1             62988 non-null  int64
 13  EP_SUM_YR_2             62988 non-null  int64
 14  SUM_YR_1                62437 non-null  float64
 15  SUM_YR_2                62850 non-null  float64
 16  SEG_KM_SUM              62988 non-null  int64
 17  WEIGHTED_SEG_KM         62988 non-null  float64
 18  LAST_FLIGHT_DATE        62988 non-null  object
 19  AVG_FLIGHT_COUNT        62988 non-null  float64
```

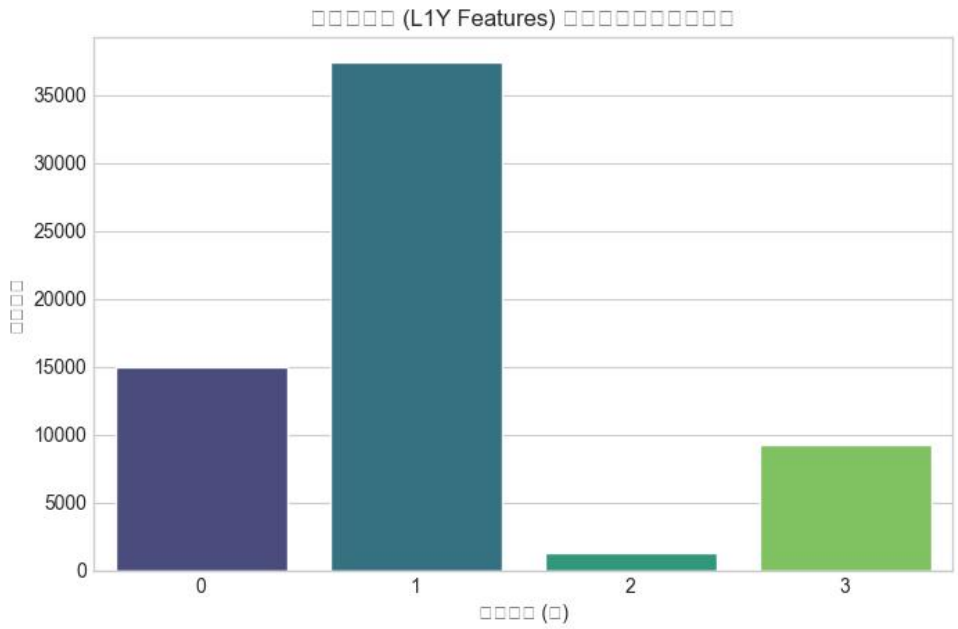
```
 27  EXCHANGE_COUNT           62988 non-null  int64
 28  avg_discount             62988 non-null  float64
 29  P1Y_Flight_Count         62988 non-null  int64
 30  L1Y_Flight_Count         62988 non-null  int64
 31  P1Y_BP_SUM               62988 non-null  int64
 32  L1Y_BP_SUM               62988 non-null  int64
 33  EP_SUM                   62988 non-null  int64
 34  ADD_Point_SUM            62988 non-null  int64
 35  Eli_Add_Point_Sum        62988 non-null  int64
 36  L1Y_ELi_Add_Points       62988 non-null  int64
 37  Points_Sum               62988 non-null  int64
 38  L1Y_Points_Sum           62988 non-null  int64
 39  Ration_L1Y_Flight_Count  62988 non-null  float64
 40  Ration_P1Y_Flight_Count  62988 non-null  float64
 41  Ration_P1Y_BPS           62988 non-null  float64
 42  Ration_L1Y_BPS           62988 non-null  float64
 43  Point_NotFlight          62988 non-null  int64
dtypes: float64(12), int64(24), object(8)
memory usage: 21.1+ MB

原始数据共 62988 行.


--- 原始实验: 基本清洗 + 总RFM特征 ---
原始实验清洗后，剩余 62988 行 (删除 0 行)
```

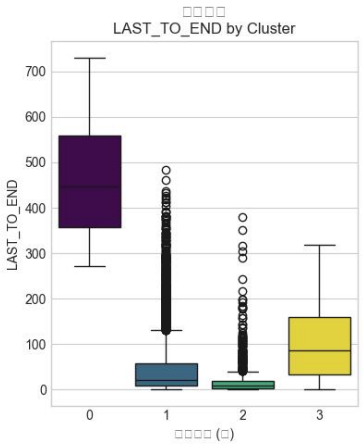
```
根据原始实验数据分析，选择最佳聚类数量 k = 4

--- 原始实验 ---
使用的聚类特征: ['LAST_TO_END', 'FLIGHT_COUNT', 'Points_Sum']
用于聚类的数据量: 62988 行
特征标准化完成.
正在使用 K-Means 进行聚类 (k=4)...

原始实验 各聚类群体特征均值:
         LAST_TO_END  FLIGHT_COUNT     Points_Sum
Cluster
0         463.382361      4.014017    4133.749698
1          42.114613     29.873244   30690.861401
2          18.018310     68.246479  104162.486620
3         101.902382      7.939309    7491.271761

原始实验 轮廓系数: 0.4932
绘制 原始实验 聚类结果可视化图表...
```

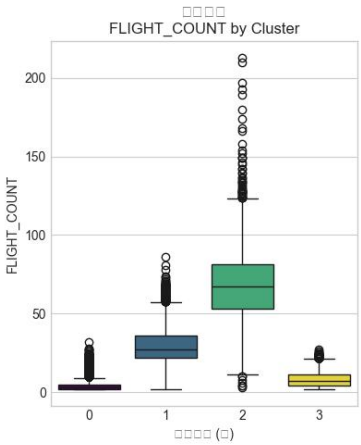
```
--- 消融实验一：使用最近一年RFM特征 ---
对比原始实验，这里更换了聚类特征，清洗方法保持与原始实验一致。
消融实验一清洗后，剩余 62988 行（删除 0 行）

--- 消融实验一（L1Y Features）---
使用的聚类特征：['LAST_TO_END', 'L1Y_Flight_Count', 'L1Y_Points_Sum']
用于聚类的数据量：62988 行
特征标准化完成。
正在使用 K-Means 进行聚类（k=4）...

消融实验一（L1Y Features）各聚类群体特征均值：
         LAST_TO_END  L1Y_Flight_Count  L1Y_Points_Sum
Cluster
0         462.541328          0.468471      568.170377
1         102.674012          4.381986     4097.517356
2          14.010614         39.840030    64884.876422
3          31.978432         17.157231    18422.465329

消融实验一（L1Y Features）轮廓系数：0.5103
绘制 消融实验一（L1Y Features）聚类结果可视化图表...
```

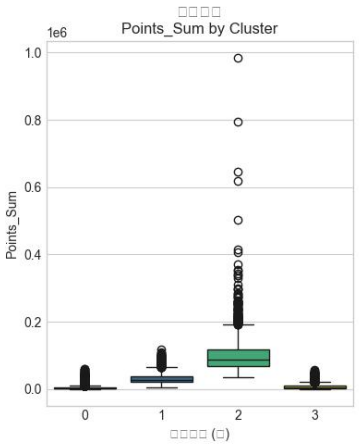
```
--- 消融实验二：使用IQR方法处理异常值（修改为Capping）---
对比原始实验，这里更改了数据清洗方法（IQR异常值封顶），聚类特征保持与原始实验一致。
消融实验二（IQR清洗）前，处理LAST_TO_END NaN后，剩余 62988 行
正在进行IQR异常值封顶处理...
已对列 'LAST_TO_END' 应用 IQR (-329.50, 626.50) 封顶。
已对列 'FLIGHT_COUNT' 应用 IQR (0.00, 33.00) 封顶。
已对列 'Points_Sum' 应用 IQR (0.00, 31593.75) 封顶。
消融实验二（IQR清洗 - Capping）后，剩余 62988 行（此方法不删除行，只修改异常值）

绘制清洗后RFM特征的箱线图，对比清洗效果...
```

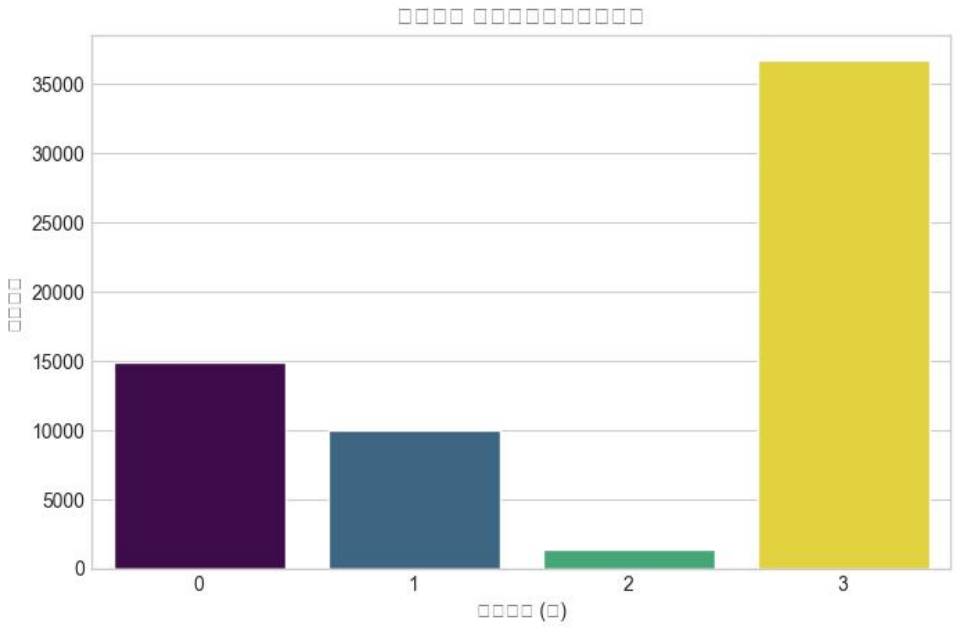

□□□□: □□Elbow□□□□□□k□
□□□□ (Inertia)
□□□□ (k)
□□□□: □□□□□□□□□□k□
□□□□ (Silhouette Score)
□□□□ (k)
□□□□ □□□□□□□□□□
□□□□
□□□□ (□)
□□□□□ (L1Y Features)
LAST_TO_END by Cluster
LAST_TO_END
□□□□ (□)
□□□□□ (L1Y Features)
L1Y_Flight_Count by Cluster
L1Y_Flight_Count
□□□□ (□)
□□□□□ (L1Y Features)
L1Y_Points_Sum by Cluster
L1Y_Points_Sum
□□□□ (□)

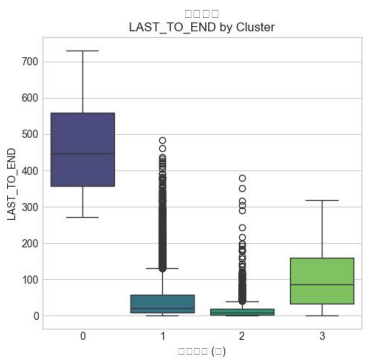

□□□□□ (L1Y Features) □□□□□□□□□□□
□□□□
35000
30000
25000
20000
15000
10000
5000
0
0
1
2
3
□□□□ (□)

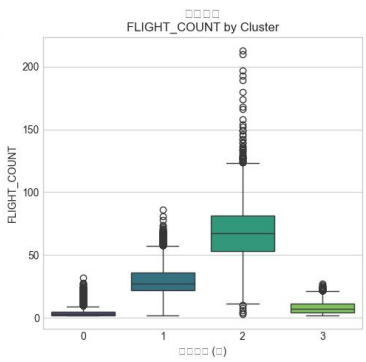

□□□□
LAST_TO_END by Cluster
LAST_TO_END
700
600
500
400
300
200
100
0
0
1
2
3
□□□□ (□)

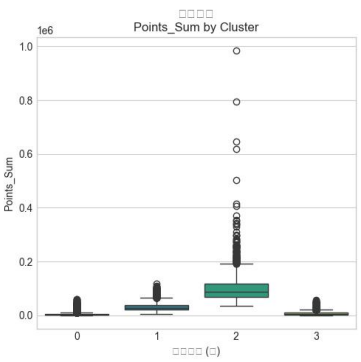

□□□□
FLIGHT_COUNT by Cluster
FLIGHT_COUNT
200
150
100
50
0
0
1
2
3
□□□□ (□)


□□□□
Points_Sum by Cluster
1e6
Points_Sum
1.0
0.8
0.6
0.4
0.2
0.0
0
1
2
3
□□□□ (□)









6.实验心得

--- 实验结果对比总结 ---

原始实验 (基本清洗 + 总 RFM):

使用数据量: 62988 行

轮廓系数: 0.4932

各聚类群体 RFM 特征均值:

LAST_TO_END FLIGHT_COUNT Points_Sum

Cluster

0	463.382361	4.014017	4133.749698
1	42.114613	29.873244	30690.861401
2	18.018310	68.246479	104162.486620
3	101.902382	7.939309	7491.271761

消融实验一 (基本清洗 + L1Y RFM):
使用数据量: 62988 行
轮廓系数: 0.5103
各聚类群体 L1Y RFM 特征均值:

Cluster	LAST_TO_END	L1Y_Flight_Count	L1Y_Points_Sum
0	462.541328	0.468471	568.170377
1	102.674012	4.381986	4097.517356
2	14.010614	39.840030	64884.876422
3	31.978432	17.157231	18422.465329

消融实验二 (IQR 清洗 - Capping + 总 RFM):
使用数据量: 62988 行
轮廓系数: 0.4416
各聚类群体 RFM 特征均值:

Cluster	LAST_TO_END	FLIGHT_COUNT	Points_Sum
0	467.768032	3.779672	3752.151834
1	77.522685	14.800503	15003.199659
2	36.159605	29.202533	28315.778189
3	117.468844	5.479083	4567.824565

--- 总结与讨论 ---
对比不同实验的客户数量、轮廓系数以及各聚类群体的特征均值，可以评估不同特征集和不同清洗方法对聚类结果的影响。
例如：
- 清洗方法对比: 原始实验使用简单删除异常值，消融实验二使用 IQR 封顶。对比两者使用的最终数据量和聚类结果（轮廓系数、群体特征），可以看出异常值对聚类稳定性和群体划分的影响。IQR 封顶通常保留更多数据，且可能受极端值影响更小，但也可能模糊掉一些真实的高价值客户特征（如果他们的极端值是真实的）。
- 特征集对比: 原始实验使用总 RFM 特征，消融实验一使用最近一年 RFM 特征。对比两者在相似数据量（如果清洗方法一致）下的聚类结果，可以评估哪种特征更能有效区分客户群体，例如最近一年特征可能更能识别活跃客户，而总特征更能识别累积价值客户。
这些对比结果将作为实验报告中“消融实验”和“结果分析”部分的重要依据。
通过本次实验，我深刻体会到了数据清洗和特征选择对聚类分析结果的重要性。实验中，我对比了不同清洗方法和特征集对客户群体划分的影响，具体心得如下：

1. **数据清洗的影响**：在消融实验二中，通过使用 IQR 封顶方法代替简单的异常值删除，我观察到数据清洗方法对最终聚类结果有显著影响。尽管 IQR 封顶保留了更多数据，

但轮廓系数有所下降，表明聚类的稳定性受到了影响。这说明在处理异常值时，需要权衡数据的完整性和聚类结果的准确性。

2. **特征集选择的重要性**：通过比较原始实验和消融实验一，我发现使用最近一年的 RFM 特征（L1Y RFM）相较于总 RFM 特征，能够更好地识别活跃客户。这表明在某些情况下，关注最近的行为可能比长期累积的数据更能反映客户的当前价值和行为模式。
3. **轮廓系数的参考价值**：轮廓系数作为衡量聚类效果的一个重要指标，在本次实验中起到了关键作用。轮廓系数较高的实验表明聚类效果更好，群体内部的相似度更高，群体间的差异也更明显。
4. **综合分析的必要性**：实验结果表明，单一的清洗方法或特征集选择并不能保证最佳的聚类效果。综合考虑数据清洗和特征选择，结合业务背景和实际需求，才能得到更有价值的聚类结果。

总体而言，本次实验不仅加深了我对数据预处理和特征工程的理解，还让我认识到在实际应用中，需要根据具体问题灵活选择合适的方法和工具。通过不断尝试和比较，才能找到最适合当前数据和业务需求的解决方案。